Approaching Single-Episode Survival Reinforcement Learning with Safety-Threshold Q-Learning

Filipo Studzinski Perotto, Melvine Nargeot, Aymane Ouahbi

Approaching Single-Episode Survival Reinforcement Learning with Safety-Threshold Q-Learning *

Filipo Studzinski Perotto, Melvine Nargeot, and Aymane Ouahbi

ONERA, DTIS, Toulouse, France
filipo.perotto@onera.fr

Abstract. *Survival Reinforcement Learning* is a specific type of RL problem constrained by a *risk of ruin*. The underlying stochastic sequential decision process with which the agent interacts includes a budget that evolves over time with the received rewards and must remain positive throughout its entire lifetime. The goal is to find a good trade-off between *exploration*, *exploitation*, and *safety* during a single learning episode, maximizing rewards while managing the available budget to minimize the probability of ruin. Existing approaches do not provide satisfactory solutions to this problem. This paper introduces the *safety-threshold heuristic*, which is used to extend the standard *Q-Learning* method. A simulated grid environment is used to evaluate its performance.

Keywords: Safe Reinforcement Learning · Budgeted Stochastic Process · Lifelong Learning · Safety-Threshold Q-Learning · Single-Episode RL

1 Introduction: Survival RL Problem

Reinforcement Learning (RL) and particularly *Deep* RL algorithms have achieved great success in a wide range of applications in recent years. RL techniques enable an agent to improve its behavior while interacting with an unknown environment, only guided by reward signals. However, the baseline RL methods are not suitable to be applied directly on expensive or safety-critical real-world systems due to the lack of integrity guarantees during the learning process. This constitutes a major difficulty in applying RL techniques in non-simulated environments since the exploratory behavior of these algorithms can potentially lead the system to catastrophic states or can cause unsustainable losses. In some cases, even an apparently good initial policy learned from simulation cannot provide enough guarantees to be correctly transferred to the real world. In other cases, reliable simulators are simply not available.

Safe RL is the subdomain of RL focused on systems that can learn from experience while being robust to strong disturbances, avoiding dangerous side-effects of trial-and-error and satisfying safety constraints. *Survival* RL is a

* This work is supported by a French government grant managed by the National Research Agency under the France 2030 program with the reference ANR-23-DEGR-0001 – The DEEPGREEN Project.

particular case of *Safe* RL in which the agent is constrained by a budget that represents a vital, limited, and expendable resource, but that can be recharged during running with the received rewards. The process terminates in failure if the budget is over, requiring the agent to mitigate that *risk of ruin* during a single run (i.e. without reset). Most RL methods rely on the episodic nature of simulated environments, in which reaching a catastrophic state can be penalized with a significantly bad reward, helping the agent to learn about what should be avoided in subsequent episodes. That assumption is particularly important for model-free methods, which suffer from high sample inefficiency. However, in non-episodic scenarios, the agent must learn to avoid catastrophic states without ever experiencing them.

In a *Survival* RL problem, the agent deals with a risk of ruin while facing a single-life process. The objective is to find a strategy that prevents from being ruined throughout that unique lifetime while still trying to learn an optimal policy with minimal regret. Some practical applications corresponding to this scenario include automated trading agents that want to learn better strategies while effectively operating on the stock market with a limited bankroll, or drones, rovers, and satellites, that want to learn how to better accomplish their mission, but paying attention to the charge of their batteries. In fact, several real-world problems as well as many bio-inspired mechanisms involve this kind of survival concerns: robots and software agents must learn to improve their performance while managing a finite amount of resources, avoiding ruin in the same way that organisms in the nature act to escape from death.

The Survival RL problem, as defined in this paper, is illustrated in Figure 1 and can be formally stated as $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, P, R, S_0, B_0, h, \gamma\}$, where:

$$\begin{cases} \mathcal{S} = \{s_1, \cdots, s_n\} & \text{is a finite set of } n \text{ discrete states;} \\ \mathcal{A} = \{a_1, \cdots, a_m\} & \text{is a finite set of } m \text{ discrete actions;} \\ P = \mathbb{P}(s' \mid s, a) & \text{is a stationary distribution defining the probability of} \\ & \text{state transition from } s \text{ to } s' \text{ with action } a; \\ R = \mathbb{P}(r \mid s, a, s') & \text{is a stationary distribution defining the probability of} \\ & \text{reward } r \in \mathbb{R} \text{ after a transition from } s \text{ to } s' \text{ with } a; \\ S_0 = \mathbb{P}(s_0) & \text{is the distribution for the initial state } s_0 \in \mathcal{S}; \\ B_0 = \mathbb{P}(b_0) & \text{is the distribution for the initial budget } b_0 \in \mathbb{R}^+; \\ h \in \mathbb{N}^+ \cup \{\infty\} & \text{is the maximal time-horizon of the process;} \\ \gamma \in [0, 1] & \text{is a constant discount factor to the cumulated rewards.} \end{cases}$$

The process starts at round $t = 0$, in which the agent observes the initial state $s_0 \sim S_0$ and the initial budget $b_0 \sim B_0$. The process evolves round by round. At each successive round t, the agent observes the current state s_t and chooses an action a_t to perform, which provokes a state transition to s_{t+1} and returns a reward r_{t+1}. The budget changes according to the received rewards, so as :

$$b_{t+1} = b_t + r_{t+1} \tag{1}$$

where b_t is the budget at round t and r_{t+1} is the reward received at round $t+1$. If the budget is depleted during running, the agent is ruined and the process stops. Otherwise, the process stops at the maximal time-horizon h, with $b_h = b_0 + \sum_{t=1}^{h} r_t$. The agent disposes of an initial policy π_0 that is improved with the experiences by a learning function ψ, producing a new policy π_t at each time step, after observing s_t, r_t, and b_t. The evolving process is illustrated in the Figure 1. An optimal policy π^* maximizes the expected γ-discounted sum of rewards over a given (potentially infinite) time-horizon h, for any initial state s_0:

$$\forall s_0 \in \mathcal{S}, \forall \pi \in \mathit{\Pi} \ : \ \mathbb{E}\left[\sum_{t=1}^{h} \gamma^{t-1} r_t \mid \pi^*\right] \ \geq \ \mathbb{E}\left[\sum_{t=1}^{h} \gamma^{t-1} r_t \mid \pi\right]. \tag{2}$$

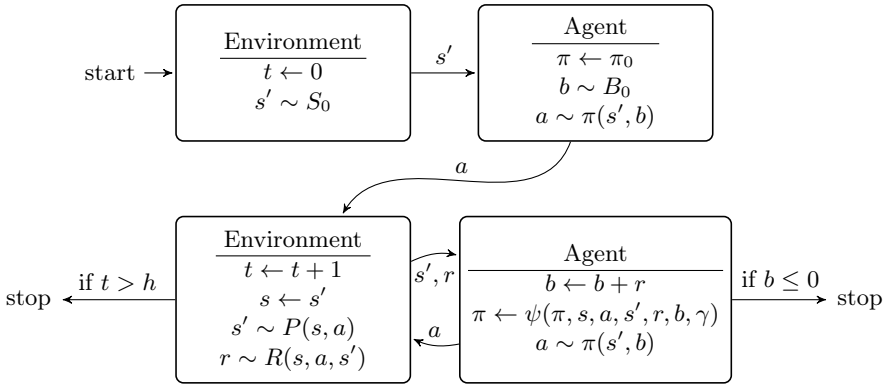


Fig. 1: In a *Survival* RL scenario, the agent-environment cycle includes a budget that should remain positive throughout the entire process execution.

In an online context, with no previous separated training phase, the overall performance of an RL algorithm is impacted by trial-and-error actions, increasing a *total regret*, defined as the difference between the sum of actually perceived rewards, and the expected sum of rewards the agent could have earned if following an optimal policy π^* from the beginning. In a classic episodic RL approach, the budget could be assimilated to an additional observation variable, and being ruined penalized with a strong negative reward. Since the objective is to learn and survive during a single episode, budget and ruin must be used as additional information at round by round decision-making time.

The contribution presented in this paper is an original heuristic to deal with survival RL problems. The insight is to extend standard algorithms by making the agent operate in two different modes, depending on the remaining budget: a *survival mode*, which is activated when the budget is too low, making the agent behave conservatively in order to recharge its budget, and a *normal mode*, to which the agent switches back when a comfortable budget level is reached

again. The background concepts related to RL, and more specifically to the classic Q-Learning algorithm, are presented in Section 2. An overview on the *Safe* RL state-of-the-art is done in Section 3. The contribution of the paper, the *Safety-Threshold* (ST) strategy used to modify the *Q-Learning* method giving rise to *ST-Q-Learning*, is explained in Section 4. Preliminary experimental results are described in Section 5, showing that the use of the proposed heuristic increases the survival expectancy of the agent, and even reduces the regret due to the maintain of a persistent but cautiousness exploratory behavior. Section 6 concludes the paper and suggests some future work.

2 Background: RL and Q-Learning

Without considering safety or survival issues, an RL agent must learn from its observations, exploring the environment and exploiting its knowledge to increase its gain, eventually converging to a policy of actions that maximizes the expected discounted sum of future rewards (an optimal policy). Classically, the RL loop is characterized by an interleaved interaction between an agent that perceives a state and executes an action, and an environment that returns a new state and a reward [48]. The environment can be described as a stochastic *Markovian Decision Process* (MDP) which evolves discretely over time [8, 46, 55].

When the "model" is known (i.e. when the functions R and P are given), an MDP can be exactly solved with polynomial complexity using *linear* or *dynamic programming* (DP) techniques [26]. Reinforcement Learning methods are necessary when a model of the world is not available in advance, which is a very common situation in practice. Model-based RL strategies try to estimate the functions R and P from the observed samples, during interaction, then extracting a policy of actions from them [28, 44], for example, using adaptive dynamic programming [32]. In contrast, model-free RL algorithms have been more effective [51] by trying to learn a policy directly from the experience, generally relying on local approximations of the state-action utilities (Q-values), updated after each observation.

Let $\bar{R}(s,a,s')$ be the mean or expected reward of function R for s,a,s'. The Bellman's equation [6] defines the value for a policy π from state s as:

$$V_\pi(s) = \sum_{s'} \Big[\mathbb{P}\big(s' \mid s, \pi(s)\big) \big(\bar{R}(s,\pi(s),s') + \gamma V_\pi(s') \big) \Big] . \tag{3}$$

The value of a given action a executed from state s, followed by policy π, is:

$$Q_\pi(s,a) = \sum_{s'} \Big[\mathbb{P}\big(s' \mid s, a\big) \big(\bar{R}(s,a,s') + \gamma V_\pi(s') \big) \Big] . \tag{4}$$

An optimal policy π^* corresponds to:

$$\forall s \in \mathcal{S} \ : \ \pi^*(s) = \arg\max_a \Big\{ \sum_{s'} \Big[\mathbb{P}\big(s' \mid s, a\big) \big(\bar{R}(s,a,s') + \gamma V_{\pi^*}(s') \big) \Big] \Big\} . \tag{5}$$

Q-learning [54] is a traditional model-free algorithm that implements an off-policy immediate temporal-difference (TD) strategy for learning a policy on environments with delayed rewards. The estimated Q function is stored in memory as a table in the form $S \times A \to \mathbb{R}$, where the entries represent the estimated utility of each state-action pair at the given time, denoted by $\hat{Q}(s, a)$. The stored Q-table is updated after each transition through a simple *value-iteration* step, based on the equation 3, using the α-weighted average of the old and the new values, where s, a, and s' are the observed transition, r is the received reward, and α is the learning rate:

$$\hat{Q}'(s,a) \leftarrow \hat{Q}(s,a) + \alpha\big(r + \gamma \max_{a'}[\hat{Q}(s',a')] - \hat{Q}(s,a)\big) \tag{6}$$

Since what can be learned depends on the agent's behavior, a necessary trade-off must be found between *exploration* (trying different actions) and *exploitation* (choosing the action with best expected utility based on the current knowledge). In Q-Learning, the ε-greedy technique is used to induce undirected exploration by introducing some randomness to the decision-making process [48]. At each time step, the agent either executes a random action with probability ε, or follows the current estimated optimal policy with probability $1 - \varepsilon$. To avoid paying a linear regret due to a constant exploration, a time-dependent exploration parameter ε_t can be used, and gradually decreased while t evolves [4].

A different approach to solve the exploration-exploitation dilemma is the *optimism in the face of uncertainty*. The idea is to explore potentially good but insufficiently frequented state-action pairs [36] (directed exploration). In tabular methods, like Q-Learning, it can be done by simply initializing the Q-values optimistically, and then following a completely greedy strategy. The more a state-action pair will be tried, the closer its estimated utility will approach its true value. Most exploration-exploitation trade-off strategies found in the literature lean on the *upper-confidence bound* principle [10, 29, 5, 45].

3 State of the Art: Safe, Risk-Averse, and Lifelong RL

The baseline RL methods are not suitable to be applied directly on expensive or safety-critical real-world systems due to the absence of integrity guarantees during the learning process [18, 7, 14]. Particularly in the case of complex physically embodied systems, even if the agent is able to previously learn a policy of actions, either through offline RL using a database of experiences, or through online RL into a simulated environment, in the real scenario the agent will eventually face unexpected events, perhaps disastrous. Successful examples like the drone racing agents [47] are extensively trained in simulation, coupled with physically informed constraints, and fine-tuned with real-world data using off-line RL, before facing the wild reality. For that reason, *Safe RL* is the object of an increasing number of publications [33, 30, 17, 25, 49, 53, 19, 35, 23, 27, 52, 50, 57, 34, 21, 2, 14, 15, 16, 24], attracting the attention of the AI community and giving rise to dedicated workshops in prestigious conferences.

Safe RL is concerned with safety during the learning process, and is synonym of "Risk-Averse" RL. Two distinct approaches appear in the literature do deal with risk [24]: (a) modifying the optimality criterion (the classic expected discounted cumulative reward) with a safety factor, or (b) incorporating external knowledge during the exploration process. In the first approach, the agent takes into account the expected variability on the utility of state-action pairs in order to identify (and avoid) less predictable (thus considered risky) actions. The notion of safety corresponds to reward and transition stability. The risk-reward trade-off is then addressed by *mean-variance* metrics or *value-at-risk* metrics, based on the lower quantiles of the estimated utility distribution for each state-action pair [35, 12, 15]. However, no notion of budget is considered in that approach, making it incompatible with the survival RL problem proposed here. The risk is understood as variability or unpredictability, and does not account for a risk of ruin.

In the second *Safe* RL approach, the idea is to use a standard RL mechanism, but monitoring and eventually interfering in decision making whenever needed in order to ensure safety. For example, a *shield* module can be introduced within the sensorimotor loop of the agent, endowed with previous knowledge to prevent the RL mechanism from choosing unsafe actions [2]. Another strategy is to gradually improve an initial predefined baseline policy, which is assumed to be stable and safe but suboptimal. The idea is to promote a controlled exploration in which the agent deviates from the baseline behavior smoothly [7, 16, 23]. The baseline policy works by delimiting a safe region, which is gradually extended by the learning mechanism. In a similar approach called *Conservative* RL, the goal is to perform at least as well as an existing baseline policy during the learning process [22].

In another framework called *Budgeted* MDP [11, 9, 56], the goal is to maximize the rewards constrained by a total budget, but with separated reward and cost functions. The budget is consumed by the costs, which means that it can only decrease at every round, until eventually being depleted. Solving this problem involves searching for a good reward-cost ratio. The same strategy is not feasible in the survival setting due to the fact that there is a single reward function returning positive and negative values in a process that can possibly run infinitely.

In this paper we are interested in a class of problems that we call *Survival* RL, which cannot be reduced to any of the previously cited *Safe* RL approaches, and can be seen as the multi-state version of *Survival Multi-Armed Bandits* [43, 42, 41]. A similar problem found in the literature is called *Single-Life* or *Lifelong* RL [13], but it is more related to *transfer learning* techniques, where an agent must adapt to a new scenario based on what was previously learned in other scenarios, for example, by shaping rewards to stimulate staying within the known region of the state space.

In the proposed problem, the agent aims to learn an optimal policy constrained by a *budget* that can be increased or decreased at each round with the received rewards, and which must remain positive all along the process. Since rewards can be positive and negative, the agent can either increase the probability of running the process indefinitely, becoming infinitely rich, or inversely, can increase the

probability of ruin, until eventually getting broke. More generally, the underlying *Survival* RL problem can be seen as a *Constrained* MDP [3, 19, 37, 1, 38], where the rewards are to be maximized, subject to maintaining a positive budget along the entire process lifetime. Another possible definition is a multi-objective optimization problem, where the rewards are to be maximized, and the probability of ruin is to be minimized.

4 Contribution: Safety-Threshold Q-Learning

In this section, the *Safety-Threshold Q-Learning* (STQ) method is introduced (Algorithm 1), as the main contribution of the paper. It is designed to tackle *Survival* RL problems using the original *Safety-Threshold* heuristic. The insight is the following: when the budget is low, the risk of ruin is high, then the agent should be pragmatic by exploiting its current knowledge to recharge the budget; in contrast, when the budget is high, the risk of ruin is low, then the agent can continue with exploratory actions. STQ distinguishes between a high and a low budget based on two hand-tuned hyperparameters, w_q and w_k.

In addition, STQ extends the classic Q-Learning by storing a second state-action value table, called K-table, with the same dimensions than the Q-table. Both tables are updated in the same way (Eq. 6) and at same time, after every round. The only difference lies on their initialization: the Q-table is initialized with zeros (neutral), i.e. $\forall s, a : Q_0(s, a) = 0$, while the K-table is initialized with a positive value $k_0 > 0$, in order to create an optimistic behavior, so as $\forall s, a : K_0(s, a) = k_0$, where k_0 is also a hand-tuned hyperparameter. In this way, when the actions are chosen based on the K-table, the principle of *optimism in the face of uncertainty* holds, and the agent will execute a directed exploration, tending to navigate to promising few observed transitions.

In this way, beyond the *exploration rate* (ε), the *learning rate* (α), and the *discount factor* (γ), STQ must be tuned with 3 additional hyperparameters: the *safety threshold* (w_q), the *exploration threshold* (w_k), and the initial k-value (k_0). During running, the agent switches between two behavior modes: *normal mode* and *survival mode*. When the budget falls under the *safety threshold* ($b_t < w_q$), the agent enters in *survival mode*, becoming greedy (i.e. following the best estimated action for the observed states, and canceling any random exploration), choosing the actions based on the "neutral" Q-table, in order to recharge its budget. In contrast, when the budget overpasses the *exploration threshold* ($b_t > w_k$), the agent returns to *normal mode*, free to conduct exploration due to both the undirected approach resulting from the use of ε-greedy action choice, and the directed approach resulting from the use of the "optimistic" K-table for decision making. The insight is that the *safety threshold* helps to keep the agent far from ruin by making it conservative (greedy) in order to prioritize budget recharging by following the estimated best actions according to the current experience.

To ensure a comfortable budget level for exploration, two thresholds are necessary. Once activated, the *survival mode* is kept until the budget exceeds a superior *exploration threshold*, then the agent reverts to its *normal mode*, at least

while it does not fall below the *safety threshold* again. The use of two separated thresholds, like a thermostat, allows to avoid quick inefficient switches between the two modes. For illustrating it, in the experience shown in the Figure 2, a better exploration rate is obtained, for the same number of steps, when the agent is using separated safety and exploration thresholds, than when it is using a unique threshold. The STQ method is described in Algorithm 1.

Algorithm 1 - Safety Threshold Q-Learning (STQ)

Input: b_0 {the initial budget}, s_0 {the initial state}, w_q {the safety threshold}, w_k {the exploration threshold}, k_0 {the initial optimistic mean value}, ε {the random exploration rate}, γ {the discount factor}, α {the learning rate}, h {the maximum time-horizon}.

$t \leftarrow 0,\ \ s \leftarrow s_0,\ \ b \leftarrow b_0$
$\forall s, a\ :\ \ Q(s,a) \leftarrow 0,\ \ \ K(s,a) \leftarrow k_0$
$survival \leftarrow$ `false` } *initialize*

while $b > 0$ **and** $t \leq h$ **do** } *loop*

  $survival \leftarrow \begin{cases} \texttt{true} & \textbf{if } b_t < w_q \\ \texttt{false} & \textbf{if } b_t > w_k \end{cases}$ } *update mode*

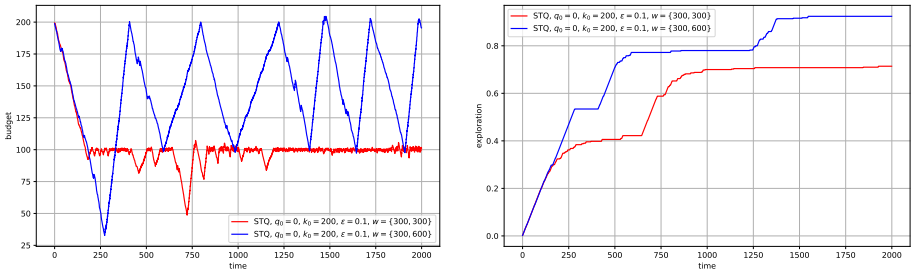
  if *survival* **is** `true` **then**
    $a_t \leftarrow \arg\max_a Q(s,a)$ *#greedy Q*
  else
    either with probability ε
      $a_t \leftarrow \arg\max_a K(s,a)$ *#greedy K*
    or with probability $1-\varepsilon$
      $a_t \leftarrow \mathbb{U}(\mathcal{A})$ *#random action*
  end if } *choose action*

  execute a_t
  observe $s_{t+1},\ r_{t+1}$
  $b_{t+1} \leftarrow b_t + r_{t+1}$ } *conclude cycle*

  update $Q(s,a)$ and $K(s,a)$ using Eq. 6 } *learn*

end while

In classic RL, an intuition involving the exploration-exploitation dilemma is that the agent should have an initial exploratory tendency that is gradually changed by a greedy behavior at the limit when time goes to infinity, and the error on the estimated utilities is supposed to approach zero. In the survival problem, however, that shift should be based on the budget instead of time [41]. For this reason, when an STQ agent has enough budget to explore, it will base its decisions on the optimistic K-table. However, if it is in *survival mode*, it will use the neutral Q-table. As the update formula is the same for both tables,



(a) Evolution of the budget over time.

(b) Exploration rate over time.

Fig. 2: Example of a typical execution with horizon $h = 2000$, and initial budget $b_0 = 200$. STQ using the same value ($w_q = w_k = 100$) for both safety and exploration thresholds explores the environment less than the one using separate thresholds ($w_q = 100$, $w_k = 200$).

they should both converge to similar values if the agent can dispose of enough experience. Thus, after a sufficiently large number of time steps, whatever the agent's mode, it will finally make decisions following an exploitation logic.

5 Experimental Results on a Grid World

A survival reinforcement learning problem is non-episodic, forcing the agent to use its knowledge and exploit non-optimal rewards to keep its budget strictly positive (i.e. to survive). To evaluate the proposed method in a survival context, a 2D grid problem is defined. The agent is positioned into a corner of a grid map, corresponding to a matrix with size $X \times Y$, where $X = 25$ is the number of columns and $Y = 5$ the number of rows in the grid. The reward distribution R is defined in function of the cost of movement, the influence of reward spots into a set $\mathcal{R}$, and a variance factor σ^2. A big reward spot ($r_{big} = +10$) is positioned at the opposite diagonal corner in relation to the agent's initial position. Two small positive reward spots ($r_{small} = +1$ and $r_{mid} = +2$) are placed in the diagonal between the agent and the big reward, around 1/3 and 2/3 of the total distance, respectively. A spreading factor $\eta \in [0, 1]$ make adjacent cells inherit part of the rewards, exponentially decreasing with the distance to the respective spots. That reward function, represented as a relief on the grid, is shown in the Figure 3.

The agent can choose between 4 actions: *north*, *south*, *east*, or *west*, with deterministic effect of moving the agent to the corresponding direction. But moving is costly, and the cost becomes bigger in the region near to the big reward. In fact, a linear variation is applied in function of the distance to the opposite corner in relation to the agent's initial position, from $r_{cost}(s) = -0.5$ when coordinate $x_s = 1$, to $r_{cost}(s) = -1.0$, when $x_s = 25$. If the agent tries to go outside of the map, it remains in the same state, receiving the corresponding reward. Let $\mathcal{N}$ be a Gaussian distribution with mean μ and variance σ^2, and d

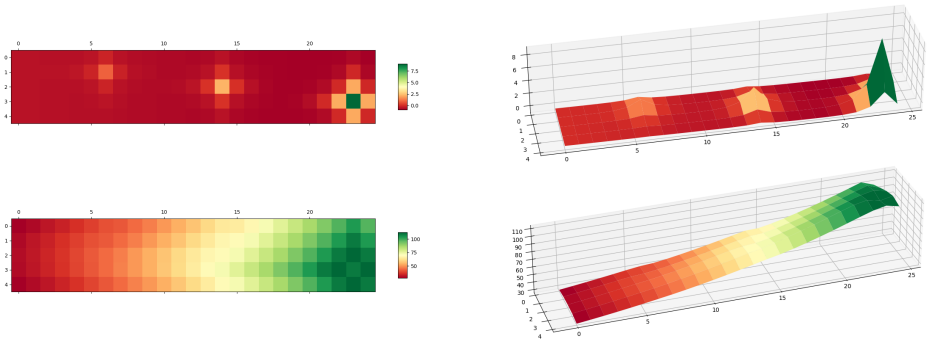


Fig. 3: At the top, the reward map for the survival experience with rectangular map of size 25×5, $\gamma = 0.95$, one reward spot of $+10$ and two reward spots of $+1$ and $+2$, combined with costs varying from -0.5 to -1.0, from one side to another, and a reward spreading $\eta = 0.2$. At the bottom, the true Q-Values for the designed survival experience, obtained through dynamic programming.

be the distance (the number of necessary steps) between the cell represented by state s at coordinates (x_s, y_s) and the spot position at coordinates (x_{spot}, y_{spot}), defined as $d_{spot}(s) = |x_s - x_{spot}| + |y_s - y_{spot}|$. The effective reward distribution when the agent steps into a cell is:

$$R(s) = \mathcal{N}\big(\mu(s), \sigma^2\big) \quad \text{with} \quad \mu(s) = r_{cost}(s) + \sum_{spot \in \mathcal{R}} \eta^{d_{spot}(s)} r_{spot} \tag{7}$$

The consequence is that, for the cells of the grid far from the reward spots, any movement executed by the agent will return a negative reward, reducing its budget. A good strategy should be using the small reward spots to recharge the budget, and then be able to explore the environment until reaching the big reward, without being ruined before. This "reward desert" makes exploration difficult, and to make things worse, the moving costs are smaller near to the initial position, which can potentially inhibits exploration in the opposite side of the grid, where the big reward is placed. Whereas a conventional agent would take the risk of dying by exploring the environment without taking the budget into account, or, inversely, would accept to survive using a sub-optimal strategy, lying on the small positive rewards, STQ explores the environment while managing its budget, exploiting positively rewarded actions, although sub-optimal, in order to ensure a sufficiently frequent budget recharging, then continuing exploration to discover an optimal policy. The true Q-values of the problem are presented in the Figure 3 (bottom).

Different scenarios with initial budgets $\{100, 150, 200, 250, 300, 350, 400\}$ have been tested into a 25×5 grid-world, with discount factor $\gamma = 0.95$. Each simulation ran until a maximal time-horizon $h = 5000$ steps, and was repeated 500 times to allow some statistical analysis. Three different variations of the classical Q-Learning method have been tested: a greedy one, another that explores based

on the *optimism in the face of the uncertainty*, with Q-tables initialized with value $q_0 = 200$, and ε-greedy with exploration rate fixed to 0.1. Those classical Q-Learning instances have been compared to Safety-Threshold Q-Learning, with Q-table values initialized to 0 and K-table values initialized to $k_0 = 200$. Different combinations of safety and exploration thresholds have been tested: $\{100, 200\}, \{200, 400\}, \{50, 800\}$. The parameter α (learning rate) was set to 0.5 for all algorithms. The experimental results have been produced into a Intel core i7 CPU. All the algorithms have been implemented in Python from scratch.

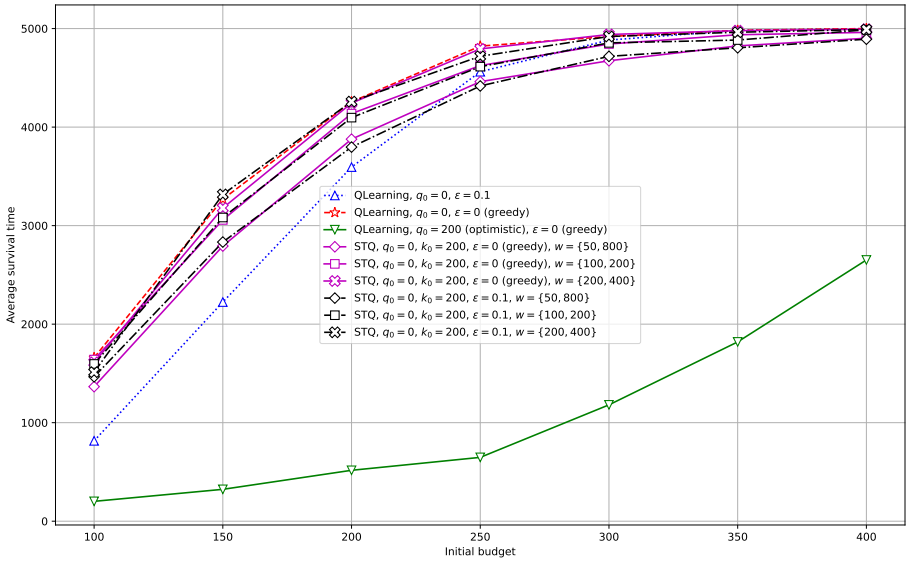


Fig. 4: Average survival time as a function of the initial budget b_0 from 100 to 400, with maximal time-horizon $h = 5000$. Simulation repeated 500 times in a grid map with dimensions 25×5.

The obtained results indicate that STQ highly improves the performance of classic Q-learning in terms of exploration (Figure 5). The agent is persistently exploratory when the budget is greater than the exploration threshold, allowing a very good approximation between the learned Q-function and the true Q-function for the entire state-action space. It can be surprising that, in terms of survival rates, the proposed heuristic is not significantly better than classic greedy or ε-greedy (Figure 4). It is because those methods are able to survive using the small rewards, which allows positive rewarded trajectories, even if not-optimal. But STQ presented a better average reward return in the long-term (Figures 6 and 7) thanks to the systematic exploration that allows to discover the big reward earlier. Figures $4 - 7$ present the average survival time, average exploration rate, average final budgets, and the average budget evolution, for Greedy ($\varepsilon = 0$),

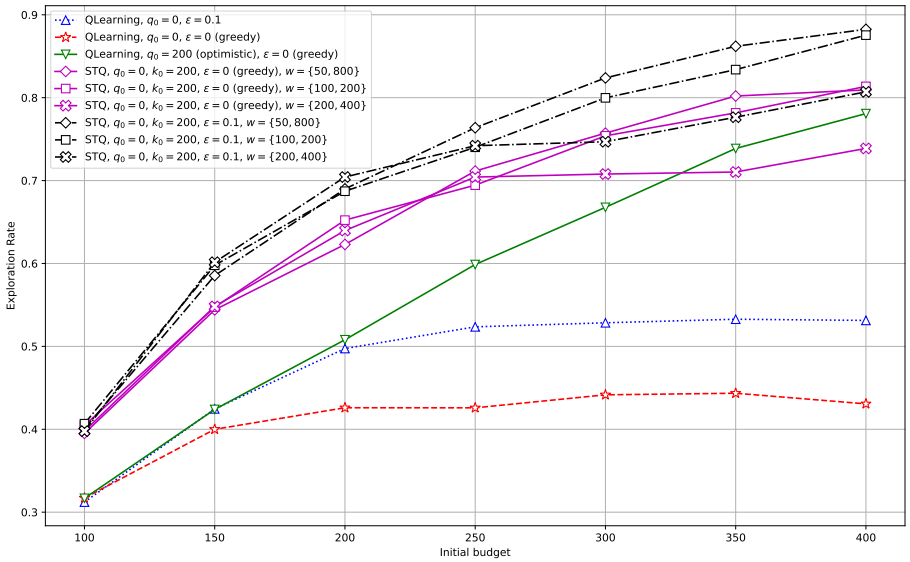


Fig. 5: Exploration rate (visited state-action pairs) as a function of the initial budget b_0 from 100 to 400, with maximal time-horizon $h = 5000$. Simulation repeated 500 times in a grid map with dimensions 25×5.

Classic Q-learning ($\varepsilon = 0.1$), Optimistic-Greedy ($\varepsilon = 0$, $q_0 = 200$), and the combinations of STQ with $\varepsilon = 0.1$ or greedy, thresholds $\{100, 200\}$, $\{200, 400\}$ and $\{50, 800\}$, with $k_0 = 200$, $\alpha = 0.5$, and $\gamma = 0.95$.

6 Discussion, Conclusion, and Future Work

Survival RL is still a class of understudied problems, with diverse open questions. In this paper, we propose a first heuristic approach to deal with Survival RL. Safety-Threshold Q-Learning introduces three new meta-parameters: w_q, w_k, and k_0, and allows the agent to change between two behaviors: in *survival mode* it uses a *neutral* Q-table, which has been initialized with 0, and follows a greedy policy, trying to exploit the most positive decisions given its current knowledge; in *normal mode*, the agent follows the policy suggested by a second, *optimistic*, K-table ("K" for knowledge), which had been initialized with highly positive values, defined by the parameter k_0. The double threshold works like a thermostat: when the budget goes under w_q, the agent enters in survival mode and becomes greedy, trying to make the budget increase again using its best current policy, even if not optimal. Then, when the budget becomes sufficiently high, greater than w_k, it comes back to the normal mode, which is very explorative at the beginning, converging to the optimal policy with high probability when the K-table and Q-table approximate the true Q-values.

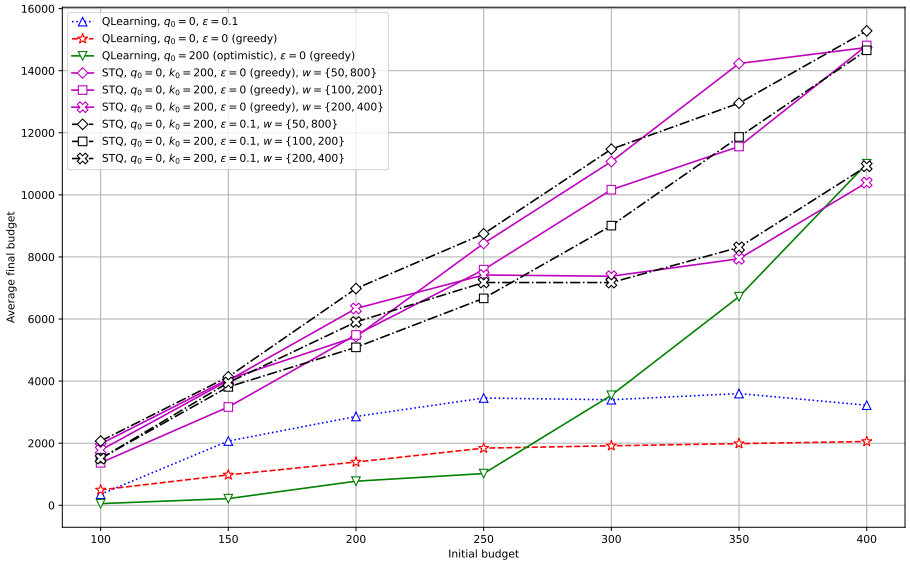


Fig. 6: Average final budgets as a function of the initial budget b_0 from 100 to 400, with maximal time-horizon $h = 5000$. Simulation repeated 500 times in a grid map with dimensions 25×5.

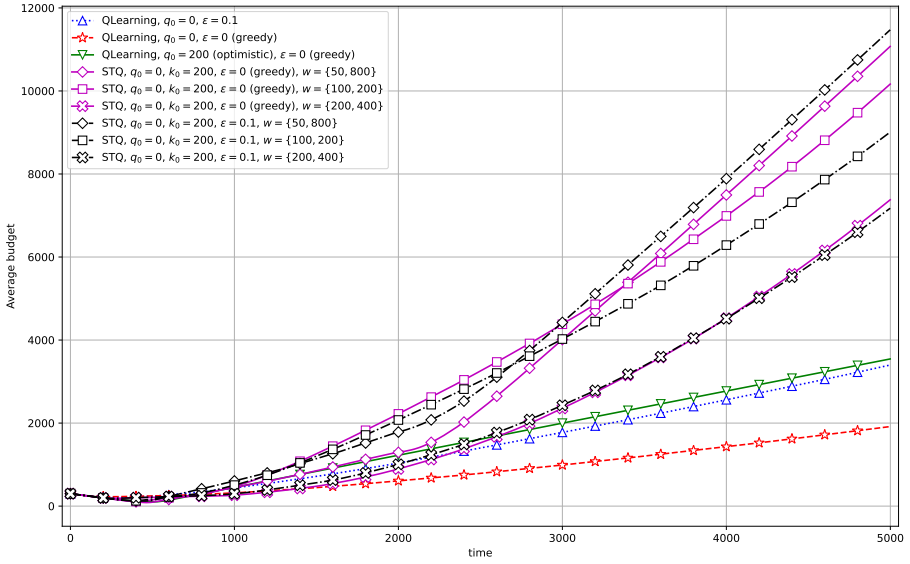


Fig. 7: Average budget evolution over time, until horizon $h = 5000$. Simulation repeated 500 times in a grid map with dimensions 25×5 with $b_0 = 300$.

Experiments in a grid world show that STQ executes a kind of systematic exploration on the first rounds of the simulation, eventually finding a small reward, to which it returns when necessary to recharge the budget. The random exploration actions used by the classic Q-Learning makes more difficult to go to the other side of the board. At the same time, since Q-Learning is not aware about the risk of ruin, it depletes the budget more often.

Finally, even if most of the classic (tabular) RL methods have proven interesting theoretical guarantees, the need of storing a Q-table limits their practical application to complex real-world problems. The combinatorial nature of an exhaustive enumeration of states would make impossible to maintain a Q-table in memory. In addition, the discrete representation of states and actions can be inappropriate to domains that could be naturally represented by continuous dimensions. In recent years, the rise of *Deep Neural Networks* (DNNs) provided new possibilities for function approximation and representation learning [31]. Deep RL combines DNNs with classic RL methods, allowing to improve scalability and to solve the complexity issues faced by tabular methods, thus enabling decision making and learning in high-dimensional state and action spaces [20]. When available, gradients provide a strong learning signal that can be estimated through sampling, offering great generalization capacities. Powerful open-source frameworks are available today allowing to implement and test new Deep RL algorithms easily, like *PyTorch*, *TensorFlow*, *Keras*, and *AIDGE*.

The next steps of this research involve the extension of the proposed heuristic to other methods, including Deep RL algorithms. The algorithm *Deep Q-Networks* (DQN) [40, 39], for example, is a deep version of Q-Learning and relies on approximating the state-action values Q^* using DNNs. The problem is that STQ implemented optimism by initializing a K-table optimistically. This kind of initialization is not possible in DNN, since the parameters of the network impact multiple states at once, and the relation between neurons and states changes during learning. A second improvement can be replacing the fixed thresholds by some adaptive function, making the agent changing between survival and normal mode smarter and smoothly.

References

1. Achiam, J., Held, D., Tamar, A., Abbeel, P.: Constrained policy optimization. In: Proc. of the 34th ICML. p. 22–31. PMLR (2017)
2. Alshiekh, M., Bloem, R., Ehlers, R., Könighofer, B., Niekum, S., Topcu, U.: Safe reinforcement learning via shielding. In: Proc. of 32nd AAAI. pp. 2669–2678 (2018)
3. Altman, E.: Constrained Markov Decision Processes. Chapman & Hall (1999)
4. Auer, P., Cesa-Bianchi, N., Fischer, P.: Finite-time analysis of the multiarmed bandit problem. Mach. Learn. **47**(2-3), 235–256 (2002)
5. Auer, P., Ortner, R.: Logarithmic online regret bounds for undiscounted reinforcement learning. In: Advances in Neural Information Processing Systems 19, Proceedings of the 20th NeurIPS (2006). pp. 49–56. MIT Press (2007)
6. Bellman, R.: On the theory of dynamic programming. Proc. Natl. Acad. Sci. USA **38**(8), 716–719 (1952)

7. Berkenkamp, F., Turchetta, M., Schoellig, A., Krause, A.: Safe model-based reinforcement learning with stability guarantees. In: Proc. of the 30th NeurIPS (2016). pp. 908–918. Curran (2017)
8. Bertsekas, D.: Reinforcement Learning and Optimal Control. Athena (2019)
9. Boutilier, C., Lu, T.: Budget allocation using weakly coupled, constrained markov decision processes. In: Proc of 32nd UAI. p. 52–61. AUAI Press (2016)
10. Brafman, R., Tennenholtz, M.: R-max - a general polynomial time algorithm for near-optimal reinforcement learning. J. Mach. Learn. Res. **3**, 213–231 (2002)
11. Caramanis, C., Dimitrov, N., Morton, D.: Efficient algorithms for budget-constrained markov decision processes. IEEE Trans. Automat. Contr. **59**(10), 2813–2817 (2014)
12. Carpin, S., Chow, Y., Pavone, M.: Risk aversion in finite markov decision processes using total cost criteria and average value at risk. In: Proc. of ICRA. pp. 335–342. IEEE (2016)
13. Chen, A.S., Sharma, A., Levine, S., Finn, C.: You only live once: Single-life reinforcement learning via learned reward shaping. In: Decision Awareness in Reinforcement Learning Workshop at ICML 2022 (2022)
14. Cheng, R., Orosz, G., Murray, R., Burdick, J.: End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks. In: Proc. of 33rd AAAI. pp. 3387–3395 (2019)
15. Chow, Y., Ghavamzadeh, M., Janson, L., Pavone, M.: Risk-constrained reinforcement learning with percentile risk criteria. JMLR **18**(167), 1–51 (2018)
16. Chow, Y., Nachum, O., Duenez-Guzman, E., Ghavamzadeh, M.: A lyapunov-based approach to safe reinforcement learning. In: Proc. of NeurIPS. v.31. pages 8103–8112. Curran (2018)
17. Du, Y., Wang, S., Huang, L.: Provably efficient risk-sensitive reinforcement learning: Iterated cvar and worst path. In: The 11th Int. Conf. on Learning Representations, ICLR 2023 (2023)
18. Dulac-Arnold, G., Levine, N., Mankowitz, D.: Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. Mach Learn **110**, 2419–2468 (2021)
19. Efroni, Y., Mannor, S., Pirotta, M.: Exploration-exploitation in constrained mdps. ArXiv **abs/2003.02189** (2020)
20. François-Lavet, V., Henderson, P., Islam, R., Bellemare, M.G., Pineau, J.: An introduction to deep reinforcement learning. Foundations and Trends in Machine Learning **11**(3-4), 219–354 (2018)
21. Fulton, N., Platzer, A.: Safe reinforcement learning via formal methods: Toward safe control through proof and learning. In: Proceedings of the AAAI Conference on Artificial Intelligence (2018)
22. Garcelon, E., Ghavamzadeh, M., Lazaric, A., Pirotta, M.: Conservative exploration in reinforcement learning. In: Proceedings of the 23rd Int. Conf. on Artificial Intelligence and Statistics (AISTATS). Proceedings of Machine Learning Research, vol. 108, pp. 1431–1441. PMLR (2020)
23. García, J., Shafie, D.: Teaching a humanoid robot to walk faster through safe reinforcement learning. Engineering Applications of Artif. Intel. **88** (2020)
24. García, J., Fernández, F.: A comprehensive survey on safe reinforcement learning. JMLR **16**, 1437–1480 (2015)
25. Gu, S., Yang, L., Du, Y., Chen, G., Walter, F., Wang, J., Yang, Y., Knoll, A.: A review of safe reinforcement learning: Methods, theory and applications (2023)
26. Howard, R.: Dynamic Programming and Markov Processes. MIT Press, Cambridge, MA (1960)

27. Jansen, N., Könighofer, B., Junges, J., Serban, A., Bloem, R.: Safe reinforcement learning using probabilistic shields. In: 31st Int. Conf. on Concurrency Theory (CONCUR 2020). LIPICS: Schloss Dagstuhl (2020)
28. Kaiser, L., Babaeizadeh, M., Milos, P., Osinski, B., Campbell, R.H., Czechowski, K., Erhan, D., Finn, C., Kozakowski, P., Levine, S., Mohiuddin, A., Sepassi, R., Tucker, G., Michalewski, H.: Model based reinforcement learning for atari. In: 8th Int. Conf. on Learning Representations, ICLR 2020. OpenReview.net (2020)
29. Kearns, M., Singh, S.: Near-optimal reinforcement learning in polynomial time. Mach. Learn. **49**(2-3), 209–232 (2002)
30. Lam, T., Verma, A., Low, B.K.H., Jaillet, P.: Risk-aware reinforcement learning with coherent risk measures and non-linear function approximation. In: The 11th Int. Conf. on Learning Representations, ICLR 2023. OpenReview.net (2023)
31. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**, 436–444 (2015)
32. Liu, D., Wei, Q., Wang, D., Yang, X., Li, H.: Overview of Adaptive Dynamic Programming, pp. 1–33. Springer, Cham (2017)
33. Liu, Z., Guo, Z., Cen, Z., Zhang, H., Tan, J., Li, B., Zhao, D.: On the robustness of safe reinforcement learning under observational perturbations. In: The 11th Int. Conf. on Learning Representations, ICLR 2023. OpenReview.net (2023)
34. Lütjens, B., Everett, M., How, J.P.: Safe reinforcement learning with model uncertainty estimates. In: 2019 Int. Conf. on Robotics and Automation (ICRA). pp. 8662–8668. IEEE (2019)
35. Majumdar, A., Pavone, M.: How should a robot assess risk? towards an axiomatic theory of risk in robotics. Robotics Research **10**, 75–84 (2020)
36. Meuleau, N., Bourgine, P.: Exploration of multi-state environments: Local measures and back-propagation of uncertainty. Mach. Learn. **35**(2), 117–154 (1999)
37. Miryoosefi, S., Brantley, K., Daume, H., Dudik, M., Schapire, R.: Reinforcement learning with convex constraints. In: Proc. of NeurIPS. v.32. pages 14093–14102. Curran (2019)
38. Miryoosefi, S., Jin, C.: A simple reward-free approach to constrained reinforcement learning. In: Int. Conf. on Machine Learning (ICML). vol. 162, pp. 15666–15698 (2022)
39. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.: Human-level control through deep reinforcement learning. Nature **518**, 529–533 (2015)
40. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.: Playing atari with deep reinforcement learning. CoRR **abs/1312.5602** (2013)
41. Perotto, F.S., Pucel, X., Farges, J.: Time is budget: A heuristic for reducing the risk of ruin in multi-armed gambler bandits. In: Bramer, M., Stahl, F. (eds.) Artificial Intelligence XXXIX - 42nd SGAI Int. Conf. on Artificial Intelligence, AI 2022, Proceedings. LNCS, vol. 13652, pp. 346–352. Springer (2022)
42. Perotto, F.S., Vakili, S., Gajane, P., Faghan, Y., Bourgais, M.: Gambler bandits and the regret of being ruined. In: Dignum, F., Lomuscio, A., Endriss, U., Nowé, A. (eds.) AAMAS '21: 20th Int. Conf. on Autonomous Agents and Multiagent Systems. pp. 1664–1667. ACM (2021)
43. Perotto, F., Bourgais, M., Silva, B., Vercouter, L.: Open problem: Risk of ruin in multiarmed bandits. In: Proc. of COLT. pp. 3194–3197 (2019)
44. Polydoros, A.S., Nalpantidis, L.: Survey of model-based reinforcement learning: Applications on robotics. J. Intell. Robotic Syst. **86**(2), 153–173 (2017)

45. Poupart, P., Vlassis, N., Hoey, J., Regan, K.: An analytic solution to discrete bayesian reinforcement learning. In: Proc. of the 23rd ICML. pp. 697–704. ACM (2006)
46. Puterman, M., Patrick, J.: Dynamic programming. In: Encyclopedia of Machine Learning, pp. 298–308. Springer (2010)
47. Song, Y., Steinweg, M., Kaufmann, E., Scaramuzza, D.: Autonomous drone racing with deep reinforcement learning. In: IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS, 2021. pp. 1205–1212. IEEE (2021)
48. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, 2 edn. (2018)
49. Thomas, G., Luo, Y., Ma, T.: Safe reinforcement learning by imagining the near future. Advances in Neural Information Processing Systems **34**, 13859–13869 (2021)
50. Turchetta, M., Kolobov, A., Shah, S., Krause, A., Agarwal, A.: Safe reinforcement learning via curriculum induction. Advances in Neural Information Processing Systems **33**, 12151–12162 (2020)
51. Valencia, D., Jia, J., Li, R., Hayashi, A., Lecchi, M., Terezakis, R., Gee, T., Liarokapis, M., MacDonald, B.A., Williams, H.: Comparison of model-based and model-free reinforcement learning for real-world dexterous robotic manipulation tasks. In: 2023 IEEE Int. Conf. on Robotics and Automation (ICRA). pp. 871–878 (2023)
52. Wachi, A., Sui, Y.: Safe reinforcement learning in constrained markov decision processes. In: Int. Conf. on Machine Learning. pp. 9797–9806. PMLR (2020)
53. Wagener, N.C., Boots, B., Cheng, C.A.: Safe reinforcement learning using advantage-based intervention. In: Int. Conf. on Machine Learning. pp. 10630–10640. PMLR (2021)
54. Watkins, C.J.C.H., Dayan, P.: Q-learning. Machine Learning **8**(3), 279–292 (1992)
55. Wiering, M., Otterlo, M.: Reinforcement learning and markov decision processes. In: Reinforcement Learning: State-of-the-Art, pp. 3–42. Springer (2012)
56. Wu, D., Chen, X., Yang, X., Wang, H., Tan, Q., Zhang, X., Xu, J., Gai, K.: Budget constrained bidding by model-free reinforcement learning in display advertising. In: Proc. of 27^{th} CIKM. p. 1443–1451. ACM (2018)
57. Yang, Y., Vamvoudakis, K.G., Modares, H.: Safe reinforcement learning for dynamical games. Int. Journal of Robust and Nonlinear Control **30**(9), 3706–3726 (2020)