Distributed Q-learning with Gittins Prioritization

Jhonathan Osin*
Department of Electrical Engineering
Technion Israel Institute of Technology
Haifa, Israel 3200003
yoni.osin@gmail.com

Naama Pearl*
Department of Electrical Engineering
Technion Israel Institute of Technology
Haifa, Israel 3200003
naama.pearl@gmail.com

Tom Zahavy
Department of Electrical Engineering
Technion Israel Institute of Technology
Haifa, Israel 3200003
tomzahavy@campus.technion.ac.il

Chen Tessler
Department of Electrical Engineering
Technion Israel Institute of Technology
Haifa, Israel 3200003
chen.tessler@campus.technion.ac.il

Shie Mannor
Department of Electrical Engineering
Technion Israel Institute of Technology
Haifa, Israel 3200003

Abstract

We consider a distributed reinforcement learning framework where multiple agents interact with the environment in parallel, while sharing experience, in order to find the optimal policy. At each time step, only a sub set of the agents is selected to interact with the environment. We explore several mechanisms for selecting which agents to prioritize based on the reward and the TD-error, and analyze their effect on the learning process. When the model is known, the optimal prioritization policy is the Gittins index. We propose an algorithm for learning the Gittins index from demonstrations and show that it yields an $\epsilon-$optimal Gittins policy. Simulations in tabular MDPs show that prioritization significantly improves the sample complexity.

Keywords: Gittins Index

*equal contribution

1 Introduction

Stochastic Gradient Descent (SGD) is a common optimization scheme which is widely used when training on large amounts of data. While common, SGD requires that the training data is independent and identically distributed. In reinforcement learning (RL) this assumption does not hold when the data is collected serially from a single policy. The DQN algorithm [Mnih et al., 2015] highlighted this issue, and showed that breaking the correlation is crucial in order to attain reasonable performance, this was performed using a replay memory.

There are two methods for overcoming the issue of correlated data in RL: (1) in off-policy learning [Mnih et al., 2015], the transitions observed during training are stored in an *experience replay* and randomly sampled during training. When the memory is large enough, these samples can be assumed to be iid. (2) In on-policy learning, the training is performed using on-line data which is collected by multiple agents acting in *parallel*. Similarly, in this approach, as the number of agents grows the correlation within the batches is removed.

Distributed data collection and off-policy learning using an experience replay are orthogonal approaches and can be combined [Horgan et al., 2018]. In this work, we focus on the distributed setting and propose a framework in which there are multiple agents N in the environment, yet only a sub-set K of them can be played. This framework encompasses both the standard single agent approach $N = 1$ [Mnih et al., 2015] and the pure-parallelism approach $K = N$ [Horgan et al., 2018], and introduces an additional challenge - selecting the optimal K agents out of the N available.

While a naive approach would be to randomly select this sub-set, we propose an alternative approach - to select the agents based on a prioritization scheme, either based on the reward or the TD-error. The benefit of this approach is highlighted in the concept of 'prioritized sweeping' [Andre et al., 1998, Schaul et al., 2015, Hessel et al., 2018], which showed that selecting which transitions to learn from is superior to random sampling. In our framework, as the number of agents N increases, the sub-set selection problem can be seen as a similar approach to that of Andre et al. [1998].

Our main contribution is a prioritization scheme based on the Gittins Index (we refer the reader to Gittins et al. [2011] for a survey of existing methods). Gittins et al. considered the scenario in which there are N parallel procedures, yet only a single procedure may be run at once. They presented the Gittins Index, a prioritization scheme which provides each procedure with an index, such that selecting the procedure with the maximal index is the optimal approach. Intuitively, selecting the agent based on the Gittins Index results in selecting the agent with the highest rate of return.

Our extensive experiments show that prioritization has a positive impact on the underlying performance and outperforms the random baseline, which is equivalent to the existing approaches. In particular, we observe that TD-error prioritization, using the Gittins Index, out-performed the other methods.

2 Gittins Index

The Gittins Index is a mathematical model for optimizing, in a sequential manner, the allocation of effort between several competing projects. The goal of the problem is to maximize the expected discounted cumulative reward. Index policies, and the Gittins Index within them, choose between the projects by computing an index for each project. The index is computed independently for each project, and the policy is to choose the project with the highest current index at each decision step. Computing the indices independently dramatically reduces the complexity of the problem.

Specifically, when each process is an MDP that is associated with a policy (a Markov Reward Process (MRP)), the Gittins Index theorem shows that the optimal allocation policy is to select the process with the highest Gittins index.

Denote the dynamics of the $i-$th MRP by π_i, the reward at state s by $r_i(s)$, and $\gamma \in [0, 1)$ is the discount factor. The Gittins Index of the $i-$th MRP is given by $\nu^{\pi_i}(s) = \sup_{\tau>0} \frac{E^{\pi_i}[\sum_{t=1}^{\tau} \gamma^t r_i(s_t)|s_0=s]}{\sum_{t=1}^{\tau} \gamma^t}$.

In this work we focus on a specific setup where there are N identical processes. Each process represents an MRP induced from the MDP M by the current policy π. Definition 1 formalizes this approach. Selecting which agent to prioritize is equivalent to the Gittins allocation problem, and the Gittins Index is therefore the optimal solution.

Definition 1. *We define the N agent MDP $\bar{M}$ as a composition of N duplicates of the MDP M. We define the state by $\bar{s} = (s^1, ..., s^N)$, where s^i is the location of the i^{th} agent. The actions $\{\bar{a}_1, ..., \bar{a}_N\}$ correlate to selecting the i^{th} agent. Selecting action a_i will transition the i^{th} agent from state s^i to s'^i based on the original transition kernel $P : S \times S \times A \to [0, 1]$, while the other agents remain static, and produces a reward $r(s^i)$.*

One way to **calculate the Gittins Index** is based on the proof by induction and interchange argument as described in Tsitsiklis [1994]. The method assumes a finite state space and a known dynamics model of the environment. The calculation is done by induction. In each step a state s^* is picked such that $r(s^*) = \max_s r(s)$ and its Gittins index is denoted by $\nu(s^*) = r(s^*)$. s^* is then removed from the state space and the problem is reduced to a smaller problem. $\tilde{p}(s, s')$ for each $s, s' \neq s^*$ is recalculated to represent the new transition probability between the states, including the

probability of a bandit to move from s to s^* and just then arrives to s'. In addition, $\tilde{r}(s)$ represents the discounted expected reward during the time steps between a bandit moves to s^* and until it returns back to s or transitions to any other state. The result of these calculations is shown to provide the Gittins index.

2.1 Calculating the Gittins Index in an Approximate Model

The Gittins Index theorem assumes full knowledge of the environment (dynamics and rewards), which are not available for RL algorithms. In this Section, we analyze what happens when the Index is calculated using an approximate model (Definition 2). The approximate model can be estimated using the agents interaction with the environment. Alternatively, it can represent prior knowledge on the domain. Theorem 1 suggests that computing the index in the approximate model yields indices that are close to those of the true model, given that the approximate model is close to the correct one.

Definition 2. *Let M and $\hat{M}$ be Markov decision processes over the same state space. Then we say that $\hat{M}$ is an ϵ-approximation of M if: $||R_M - R_{\hat{M}}||_\infty \leq \epsilon$, $||P_M - P_{\hat{M}}||_\infty \leq \epsilon$.*

The following theorem shows that calculating the Gittins index on an $\epsilon-$approximate model yields an $\epsilon-$ optimal policy in the true model.

Theorem 1. *The optimal policy for choosing K agents within all N possible operating in an MDP, considering an approximate model of the environment is to greedily select the best agents by their Gittins Index.*

Before we prove the Theorem, we cite a classic result from [Kearns and Singh, 1998].

Lemma 1 (Simulation Lemma [Kearns and Singh, 1998])**.** *Let M be a Markov decision process with $|S|$ states. Let $T \geq (1/(1-\gamma))\log(R_{\max}/(\epsilon(1-\gamma)))$, and let $\hat{M}$ be an $O(\epsilon/(|S|TG_{\max}^T)^2)$-approximation of M. Then for any policy π, $||V_M^\pi - V_{\hat{M}}^\pi||_\infty \leq \epsilon$.*

Proof of Theorem 1. The problem of choosing one of the N agents in the MDP $\bar{M}$ is in fact a bandits process, in which each agent is an MRP, and at each step the decision maker should decide which agent to activate. In this case, the Gittins Index theory claims that the maximum expected accumulated reward, $V_{\bar{M}}^{\pi^*}$ will be attained by choosing the agent with the greatest Gittins Index. Therefore, in the composite MDP the Gittins Index will also be the optimal policy. For $\hat{\bar{M}}$ as an $alpha(\epsilon)$-approximation of the original MDP $\bar{M}$ for $\alpha(\epsilon) = O(\epsilon/(|S|TG_{\max}^T)^2)$, it exists by the Simulation Lemma that for any policy the value function will be $epsilon$-optimal. In particular, the Gittins Index yields an $epsilon$-approximation of the optimal value function. □

3 Method

Setup: Our framework consists of N agents that interact with a single environment. At each time-step the decision maker selects a subset of K agents. Once an agent is selected it advances a single time-step, while the other agents remain frozen. The agents share the same policy, which is updated using data that is collected from all the agents. This data is used to train a single Q-learning based policy, which is shared across all agents.

Prioritization: In Section 2 we showed not only that the Gittins Index is the optimal solution given that the model is known, it also provides solutions that are close to optimal when we have a good approximation for the model. Next, we describe explicitly the prioritization mechanisms that we consider in this work.

During the learning process the score of each state is periodically calculated, based on the *reward* $r(s, \pi(s))$ or the *TD error* $r(s, \pi(s)) + \gamma \cdot \arg\max_a Q(s', a) - Q(s, \pi(s))$. The priority of each agent is calculated using (1) *greedy prioritization*, a naive method, in which the priority for each state is simply its score, or (2) a *Gittins Index based prioritization*, in which the state's priority is determined by the Gittins value, as described in Section 2. This approach generates four prioritization schemes, greedy reward, greedy TD-error, Gittins reward and Gittins TD-error. In addition to these prioritization schemes, we also compare to *random selection*, in which at each round the agents are selected at random. We evaluate these methods both based on the online regret and a periodic offline evaluation of the learned policy.

4 Experiments

We validate our approach on several MDP types in order to highlight the difference between the various prioritization schemes. We explain each MDP below, what goal it serves, and the conclusions from it.

Cliques MDP: This MDP is characterized by a single initial state which randomly leads to one of two cliques, regardless of the action the agent performs. While one clique yields positive rewards, the other provides a constant zero reward. During the simulation all prioritization method put effort in the positive clique, while the random selection approach activates the agents in both cliques equally. This results in low offline evaluation value and clear difference in the regret and shows the importance of using the prioritization in the learning process.

Tunnel MDP: This MDP is an extension of the Cliques MDP, in which the positive reward clique contains a short sequence of negative rewards. In this scenario, the greedy reward method prioritizes agents in the zero-reward clique over agents residing in the negative part of the positive clique. As it ignores the accumulative reward and only considers the immediate outcom, it exhibits this sub-optimal behavior.

Tree MDP: The structure of this MDP resembles a binary decision tree. At each state the agent has two actions, either to go left or right. Upon taking action a, with probability p the agent will instead perform a random action. Certain states are termination states, from which the agent returns to the root of the tree - the only way to move backwards in this MDP. In this MDP, both reward-based prioritization schemes fail, as they prioritize exploitation over proper exploration. On the other hand, the TD-error schemes prioritize exploration over the tree, resulting in fast convergence to the real values.

Cliff Walker: A variant of the classic cliff-walker [Sutton and Barto, 2018], where, at each step there is a probability in which the agent will perform a random action. As the randomness is concentrated around the cliff, the greedy TD-error approach prioritizes agents near the cliff over agents which are located in safer regions, as the Gittins considers the accumulated error, it prioritizes states further away from the edge as the risk of termination is lower and thus it would visit more states in the process.

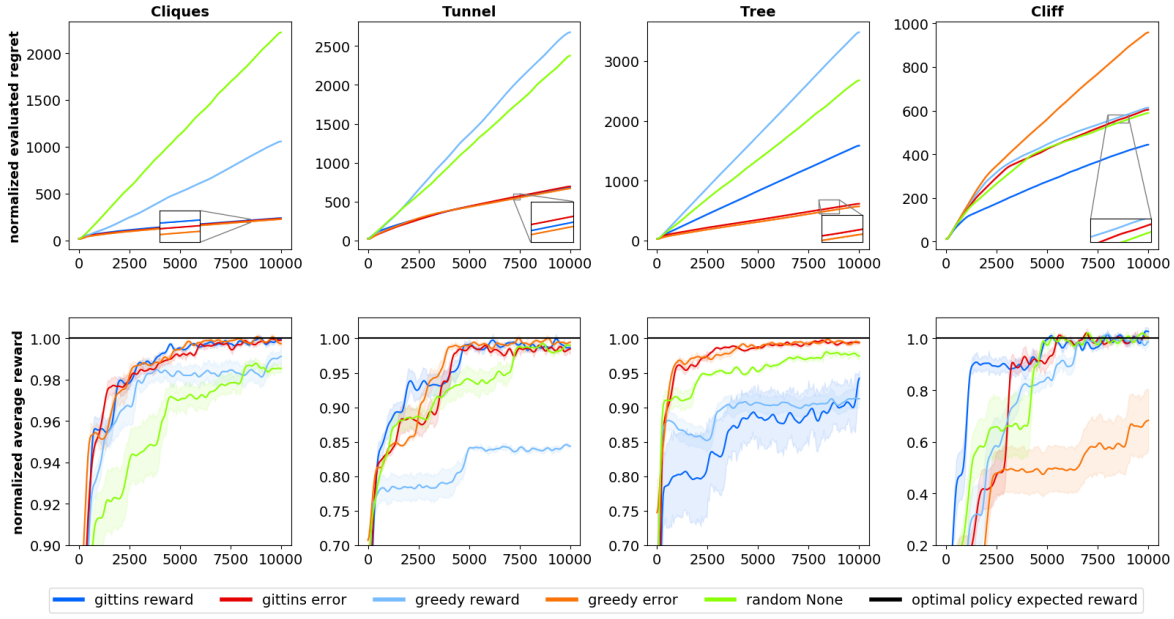

Figure 1: Performance analysis using all prioritizing methods, on MDP types

Our results, depicted in Figure 1, suggest that in most scenarios, prioritization is better than random selection, both in terms of regret and in terms of periodic evaluation of the policy. In addition, while each method fails in one MDP or the other, the Gittins approach based on the TD-error provides the most consistent behavior across all scenarios.

4.1 Temporal Extension

While so far we focused on methods that select the agents to prioritize at each time-step, in many scenarios, we would like to make a decision once every λ steps. In this section, we explore the implications of using a temporal extension ($\lambda > 1$). We focus in the Gittins TD-error prioritization scheme and compare it with the random selection scheme. Figure 2 presents our results. The results show that in using the temporal extension can improve performance for small values of λ, but for larger values of λ (e.g., $\lambda = 15$), performance deteriorates. Nevertheless, almost in all the MDPs, adding temporal extension still resulted in improvement over the random baseline.

4.2 Gittins Calculation in Approximate Model

Lastly, we test the accuracy of the Gittins Index calculation in the approximate model with respect to the exact model. In this experiment we consider computing the Gittins Index based on the reward signal. We evaluate the calculation based on three criteria (1) Difference in the Gittins Index values, (2) Number of incorrect decisions, i.e., the times a wrong agent is selected due to approximation errors, and (3) The performance difference between prioritizing based on the true

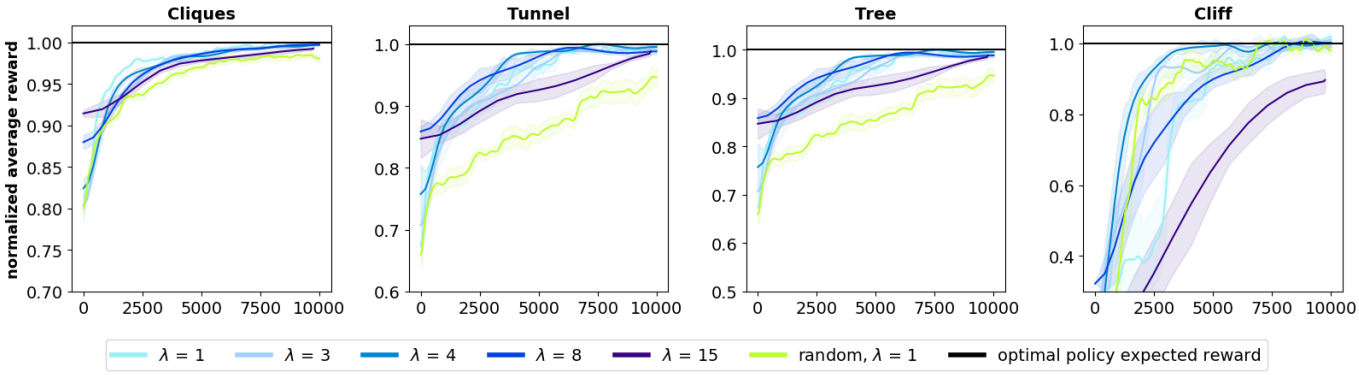

Figure 2: Temporal Extension effect on performance, using Gittins error and random on all MDP

and the approximate models. Figure 3 shows that when training begins and the approximation errors are large, there is a large difference in all of the three criteria. However, quite early in the training process, the approximated Gittins becomes quite similar to the true Gittins and prioritizes the agents well. These results provide an empirical proof to Theorem 1 and confirms that the Gittins Index using an approximate model is an efficient method for prioritization during learning.

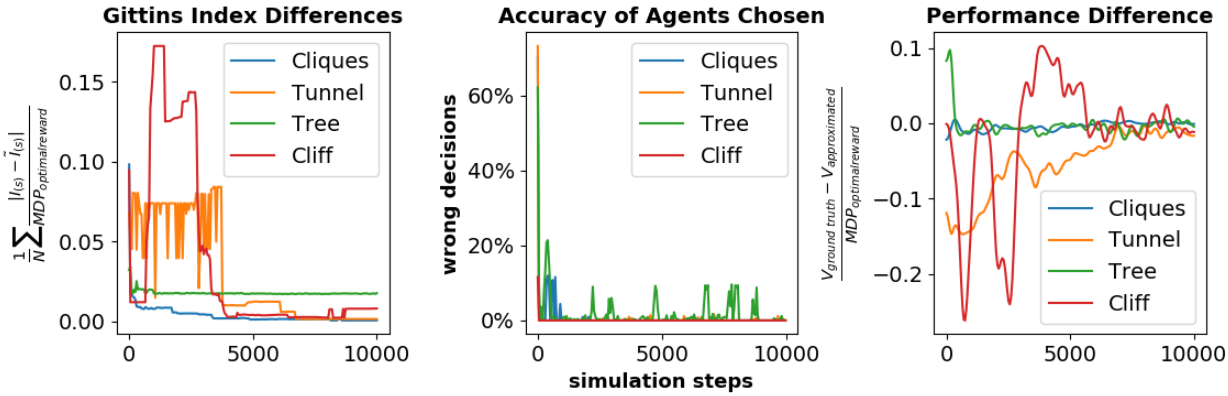

Figure 3: Comparison between the approximate and exact model for reward prioritization using the Gittins index.

5 Discussion

In this work we proposed a new framework for distributed Q-learning. We presented two metrics for prioritization (reward and TD-error) and two prioritization schemes (greedy and using the Gittins Index). Our results show that (1) prioritization based on the TD-error is the best approach, a result also seen in Andre et al. [1998], Schaul et al. [2015], Hessel et al. [2018], (2) prioritizing based on the Gittins Index is robust to temporal extensions and (3) using an approximate model to estimate the Gittins Index results in near-optimal performance when compared to using the exact model. In future work we plan to extend our framework and use it in the setting of Deep RL. We anticipate that this will require developing model free methods for learning the Gittins Index.

References

David Andre et al. Generalized prioritized sweeping. In *NeurIPS*, 1998.

John Gittins et al. *Multi-armed bandit allocation indices*. John Wiley & Sons, 2011.

Matteo Hessel et al. Rainbow: Combining improvements in deep reinforcement learning. In *AAAI*, 2018.

Dan Horgan et al. Distributed prioritized experience replay. *arXiv*, 2018.

Kearns and Singh. Near-optimal reinforcement learning in polynomial time. *Machine learning*, 1998.

Volodymyr Mnih et al. Human-level control through deep reinforcement learning. *Nature*, 2015.

Tom Schaul et al. Prioritized experience replay. *arXiv*, 2015.

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

Tsitsiklis. A short proof of the gittins index theorem. *The Annals of Applied Probability*, 1994.