Distributed Q-learning with Gittins Prioritization

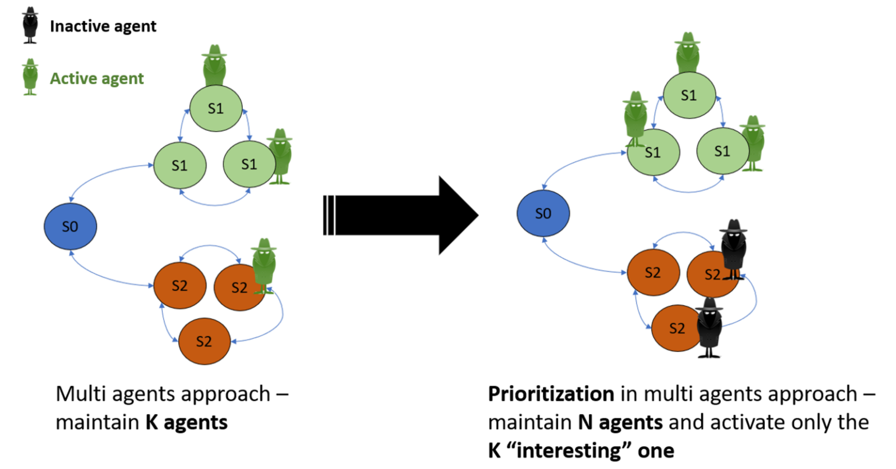
JHONATHAN OSIN, NAAMA PEARL, TOM ZAHAVY, CHEN TESSLER, SHIE MANNOR

YONI.OSIN@GMAIL.COM, NAAMA.PEARL@GMAIL.COM, TOMZAHAVY@CAMPUS.TECHNION.AC.IL, CHEN.TESSLER@CAMPUS.TECHNION.AC.IL

1. Introduction

Main Contribution

- Distributed RL framework:
  - N agents share the same policy
  - At each step, $K < N$ agents are selected to act
  - A prioritization mechanism selects between the agents
- Sampling resources are used more efficiently!



Related Work

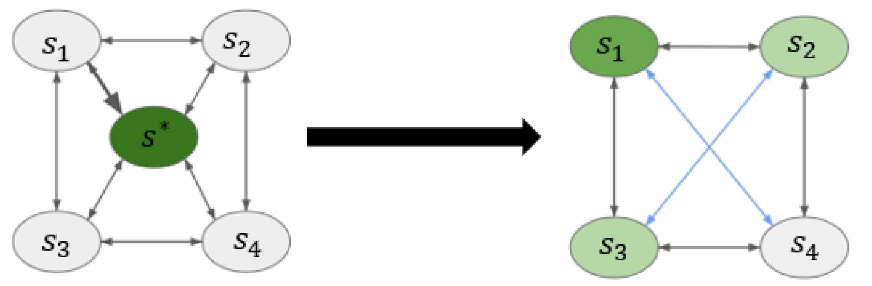
[1] **Prioritized Sweeping:** Off policy algorithm which samples state-action pairs according to some index policy.
David Andre et al. Generalized prioritized sweeping. NeurIPS, 1998.
[2] **Prioritized experience replay:** Experience collected is prioritized based on the magnitude of the error.
Schaul et al. Prioritized experience replay. ICLR, 2016.
[3] **Rainbow DQN:** An ablation test shows that prioritization [2] is the lead reason for performance boost.
Hessel, et al. Rainbow: Combining Improvements in Deep Reinforcement Learning AAAI, 2018.
[4] **Agent Parallelism:** Multiple agents are run in parallel, each with a different exploration scheme.
Horgan, et al. Distributed Prioritized Experience Replay. ICLR, 2018.

2. Gittins Index

Gittins Index

$$\nu^i(s) = \sup_{\tau>0} \frac{\mathbb{E}[\sum_{t=0}^{\tau} \gamma^t r_i(s_{i,t}) | s_0 = s]}{\mathbb{E}\sum_{t=0}^{\tau} \gamma^t}$$

- **The problem**: given K Markov Reward Process (MRP), in each timestep one process should be activated while others remain frozen. The goal is to **maximize** the accumulated reward.
- The action space is choosing which process to activate $\mathcal{A} = \{1, ..., K\}$
- The state space is the combination of all process' state $\bar{s} = (s_1, .., s_K)$. After choosing an action i, the i_{th} process is promoted, and the state $\bar{s}$ changes. $\pi^*(\bar{s}) = \arg\max_\pi \mathbb{E}\sum_{t=0}^{\infty}[\gamma^t r(\bar{s}_t, \pi(\bar{s}_t))|\bar{s}_0 = \bar{s}]$
- The problem of choosing which process to activate seems to have exponential complexity.
- The policy that maximizes the value function is the one that maximize the index of each process $\pi^*(\bar{s}) = \arg\max_{i\in[1,...,K]} \nu^i(s_i)$
- Thus, it is calculated **independently** for each process, and therefore reducing the problem to polynomial complexity.
- Calculation is done in iterations - continue until state space is of size 1:
  1. let $s^* = \arg\max_{s\in\mathbb{S}} r(s, \pi(s))$, denote it's index by $\nu(s^*) = r(s^*, \pi(s^*))$
  2. remove s^*, and recalculate $\tilde{p}(s, s')$, $\tilde{r}(s)$ for every $s \in \mathbb{S} \setminus s^*$



Gittins Index in our framework

- Under a **specific policy**, a Gittins index can be calculated for each state of the MDP.
- Considering each agent as a process, choosing the agent in the state with the highest index will maximize the future accumulative reward.

Gittins Index in approximate model

- Gittins Index theorem is the optimal policy in a planning problem, when the model is known.
- In our framework, the model is unknown. The learning process aims to aprroximate the problem.
- Define the MDP $\hat{M}$ as an $\alpha - approximation$ of the MDP M if: $||R_M - R_{\hat{M}}||_\infty \le \alpha$, $||P_M - P_{\hat{M}}||_\infty \le \alpha$.
- We show that given an α-approximation of the model, the Gittins index policy yielded from the approximation is $\epsilon(\alpha)$ optimal.

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 based on their Gittins Index.*

3. Method

Framework

- N **agents** interact with a **single unknown MDP**.
- At each timestep a **subset of** k **agents** are prioritized to advance. Other remain frozen.
- A **global policy**, is learned using **Q-learning** based on all agents observation.

Prioritization Schemes

During the learning process the **score of each state** is periodically calculated, based on either:

1. *reward* - $r(s, \pi(s))$
2. *TD error* - $r(s, \pi(s)) + \gamma \cdot \arg\max_a Q(s', a) - Q(s, \pi(s))$

Above scores yielded **4 prioritization schemes**:

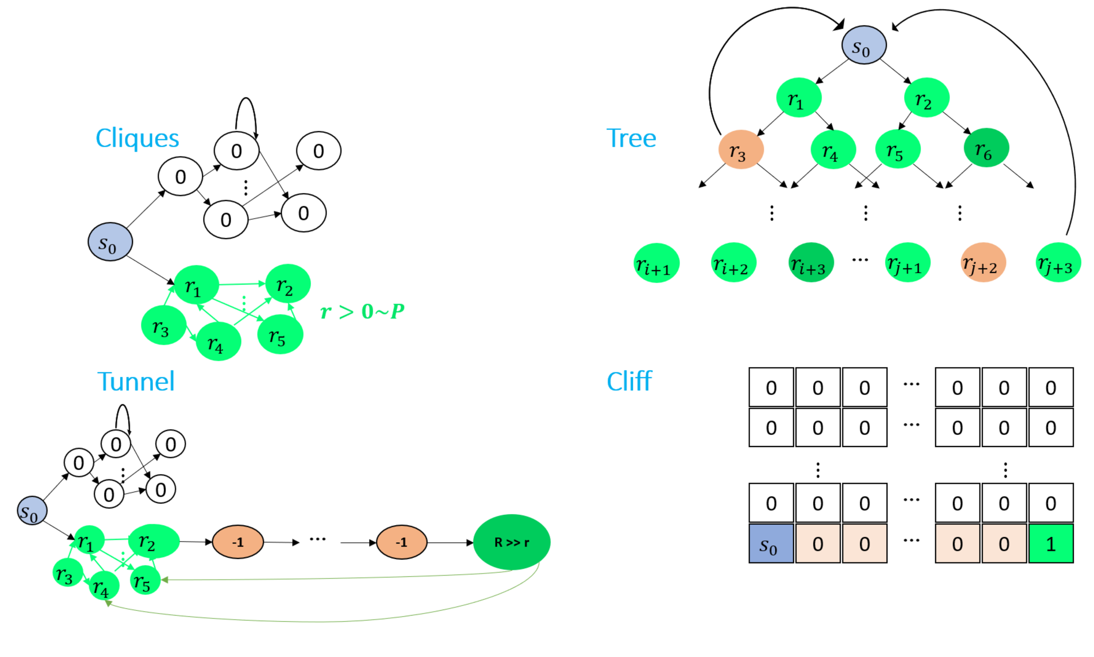
- greedy reward
- greedy TD-error
- Gittins reward
- Gittins TD-error

Performance, compared to **random prioritization** baseline, was evaluated using:

- Online regret
- Periodic offline evaluation of the learned policy

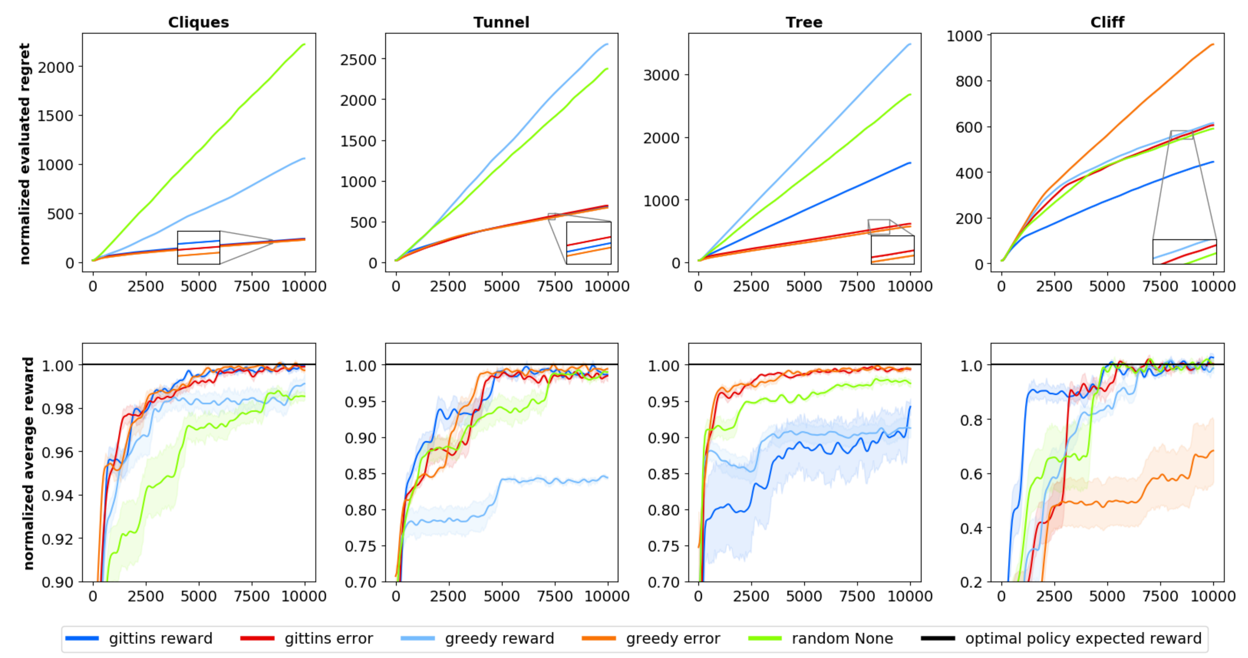
4. Experiments

MDP's



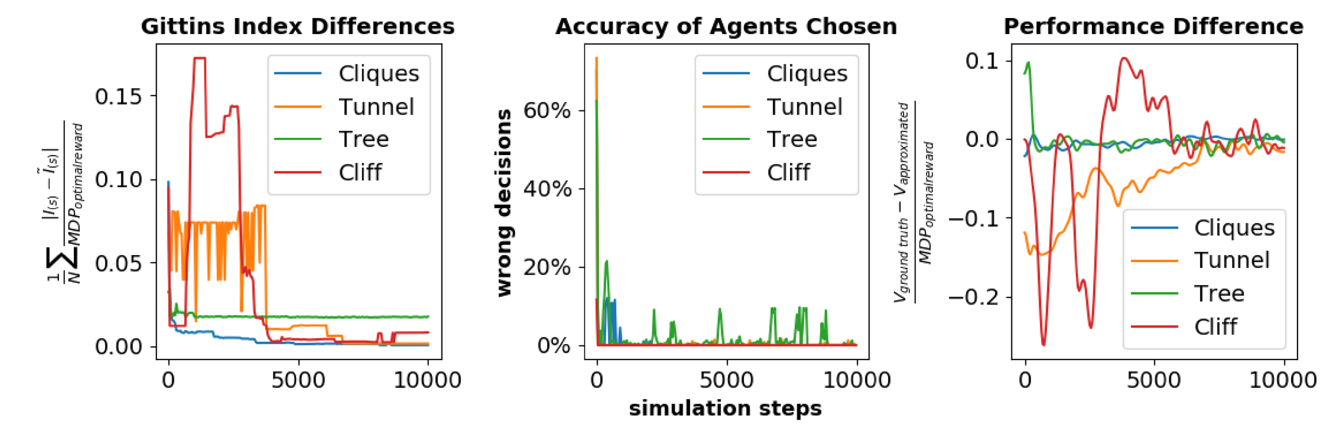
Performance analysing: Regret and average reward

- In most scenarios, **prioritization is better** than random selection
- **Gittins approach based on the TD-error** has the most consistent positive effect across all domains.



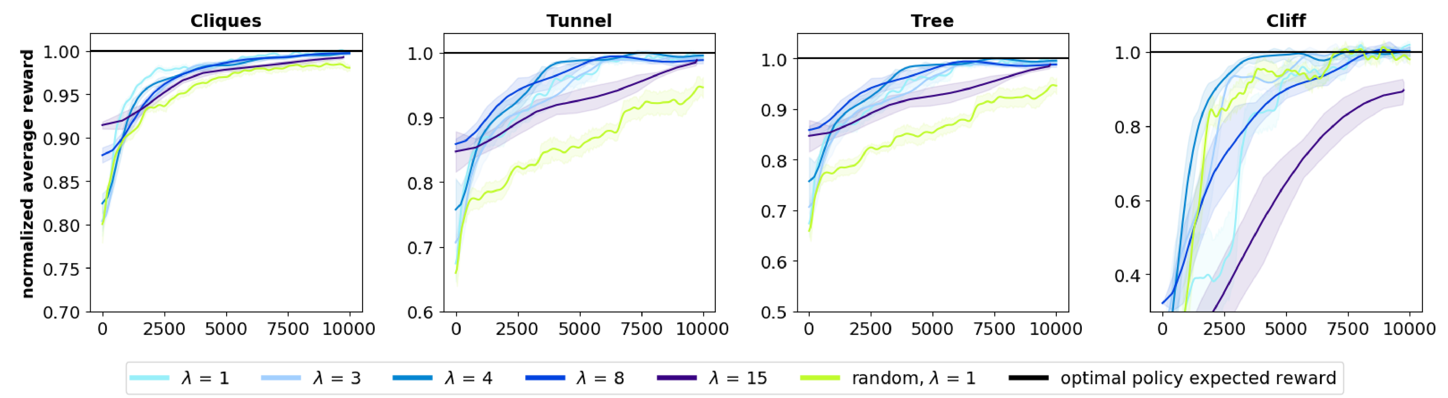
Gittins Indices Accuracy

Gittins Index calculated in the **approximate model** were compared to those in the **real model**:



Temporal Extension Implication

- Exploring the implication of selecting the agents prioritized each $\lambda > 1$ timesteps (rather then every timestep).
- Using the temporal extension can improve performance for small values of λ.
- Almost in all the MDPs, adding temporal extension still resulted in improvement over the random baseline.



5. Summary

1. Prioritization based on the TD-error is the best approach
2. Prioritizing based on the Gittins Index is robust to temporal extensions
3. Using an approximate model to estimate the Gittins Index results in near-optimal performance when compared to using the exact model

6. Work in progress

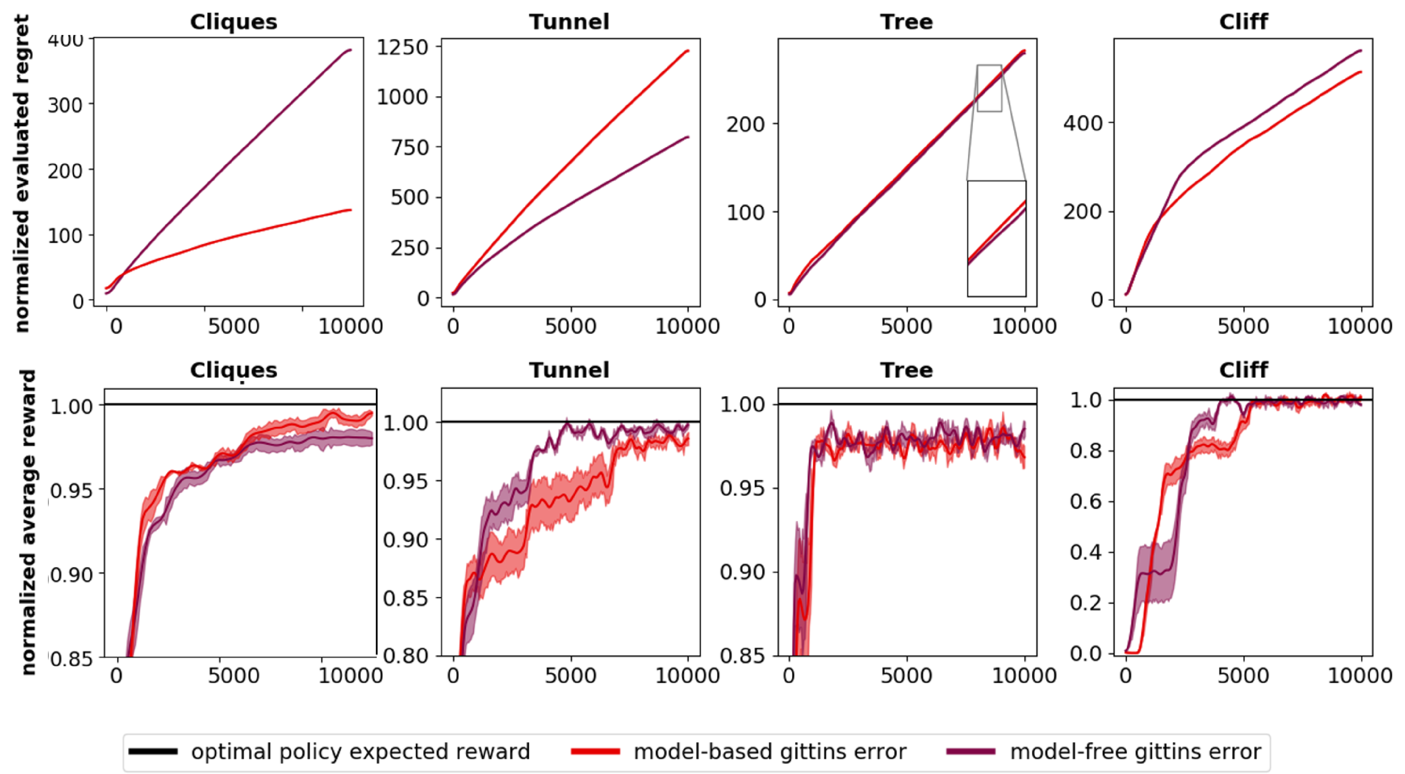
Model free Gittins index

- In order to test our method in more complex domains, we propose an estimate of the Gittins Index in a model-free setting.
- The empirical value of the index is calculated in 2 phases:
  1. An estimation based on weighted average of discounted accumulated reward is calculated for every trajectory length in $\tau \in [0, T]$, where $T \propto \frac{1}{1-\gamma}$, is the effective horizon of the problem.
  2. The final index is the maximal estimation from those calculated in step 1.

$$\hat{\nu}^\pi(s) = \max_{\tau\in[0,T]} \frac{1}{m} \frac{\sum_{t=1}^{\tau}[\gamma^t r_i(s_t)|s_0 = s]}{\sum_{t=1}^{\tau}\gamma^t}$$

Model Free Approach

We show empirical results comparing our methods in a model free setting:



Deep Reinforcement Learning Approach

- We trained a DNN which predicts the empirical Gittins index of a state, for a set of predefined trajectory lengths, denoted as $\{t_1, ..., t_k\}$.
- Let $F_\theta^\tau(s)$ be the network's prediction for the empirical Gittins index for a trajectory of length τ, where θ is the network's weights.
  Thus, the index is given by:
  $$\hat{\nu}^\pi(s) = \max_{\tau\in\{t_1,...,t_k\}} F_\theta^\tau(s)$$
- We combined the above DNN with the A2C algorithm to investigate the effect of our approach in more complex domains.
- Initial results show that A2C does not improve via prioritization. Our next step is to explore Q-learning based methods, such as [4], an approach which is highly correlated to our tabular results.

Future Plans

- Testing our approach on other, more complex domains.
- Integrating an approach which facilitates exploration via advancing agents with random actions.