G-DICE re-implemented

Caus Danu

14.12.2019

1 Re-implementation of discrete G-DICE

The aim of this short technical report is to implement a flavour of the baseline G-DICE algorithm and verify that it works correctly on the decentralized tiger problem. G-DICE stands for graph-based discrete cross-entropy policy search and is an evolutionary algorithm for finding agent policies in a discrete action space and discrete observation space. This report is made as preparation for the master thesis project to follow, that will build on top of the G-DICE algorithm to consider also continuous observations, and very high-dimensional observations.

The implementation makes use of deterministic actions that are executed in a node, unlike [1] where the authors use stochastic policies. We also define deterministic next nodes for each individual graph node, whereas [1] implement a stochastic next node transition. In this sense, we have a basic version of G-DICE that draws deterministic policies as deterministic actions and deterministic next nodes for each given current node. These deterministic policies are then simulated many times to calculate an average reward. A more detailed description of this flavour of G-DICE can be seen in Algorithm 1

Algorithm 1 G-DICE with deterministic policies

Input: Number of nodes (N_n), Number of iterations (N_k), Number of samples (N_s), Number of best samples (N_b), Number of evaluation simulations (N_{sim}), Learning rate (α), Discount factor (γ), Number of lookahead steps (h)

Output: $bestPolicy$, $bestValue$

$bestPolicy \leftarrow \emptyset$
$V_w, V_b \leftarrow -Infinity$ ▷ Worst and best values are very small initially
for $i \leftarrow 0, ..., numAgents$ **do**
    $\theta_0^{(i)(a|q)} \leftarrow UniformDistribution(), \forall q$
    $\theta_0^{(i)(q^{'}|q,z)} \leftarrow UniformDistribution(), \forall q, z$ ▷ q' means next node, z is the observation
end for
for $k = 0, ..., N_k - 1$ **do**
    $storageList \leftarrow \emptyset$
    $bestPolicies \leftarrow \emptyset$
    $bestValues \leftarrow \emptyset$
    for $sample = 0, ..., N_s - 1$ **do**
        $deterministicAction \leftarrow Sample(actionDistribution), \forall q$
        $deterministicNextNode \leftarrow Sample(nextNodeDistribution), \forall q, z$
        $reward = Evaluate(h, N_{sim},\ \gamma)$
        if $reward > V_w$ **then**
            $storageList \leftarrow tuple(deterministicAction, reward, deterministicNextNode)$
        end if
        if $reward > V_b$ **then**
            $bestPolicy \leftarrow tuple(deterministicAction, deterministicNextNode)$
            $V_b \leftarrow reward$
        end if
    end for
    $bestPolicies = Sort(storageList)[0 : N_b]$ ▷ Sort and take best N_b elements
    $bestValues \leftarrow getRewards(bestPolicies)$
    $V_w \leftarrow \min(bestValues)$
    for $i \leftarrow 0, ..., numAgents$ **do**
        $\theta_{k+1}^{(i)(a|q)} \leftarrow MLE(bestPolicies)\ \forall q$
        $\theta_{k+1}^{(i)(a|q)} \leftarrow \alpha * \theta_{k+1}^{(i)(a|q)} + (1-\alpha) * \theta_k^{(i)(a|q)}\ \forall q$ ▷ Smoothing
        $\theta_{k+1}^{(i)(q^{'}|q,z)} \leftarrow MLE(bestPolicies)\ \forall q^{'}, z$
        $\theta_{k+1}^{(i)(q^{'}|q,z)} \leftarrow \alpha * \theta_{k+1}^{(i)(q^{'}|q,z)} + (1-\alpha) * \theta_k^{(i)(q^{'}|q,z)}\ \forall q^{'}, z$
    end for
end for
return $bestPolicy$, V_b

2 Validation experiments

The purpose of these experiments was to validate if the implementation of the G-DICE algorithm behaves as expected. The established algorithm giving the ground truth data is provided by [2]. The problem to solve is the canonical Decentralized Tiger problem.

Dec-Tiger problem The problem chosen to verify the algorithm on is the Dec-Tiger problem (see [3]). The setting is as follows (see Figure 1) : there are 2 agents standing in front of two doors. Behind one of the doors there is a large treasure, whereas the other door hides a tiger.

The agents do not know which door locks the treasure, however they can make observations to maximize the probability of opening the desired door. The observations are basically audio signals. An agent can either hear the tiger to its left or right side. However there is a 15% chance of getting the wrong observation.

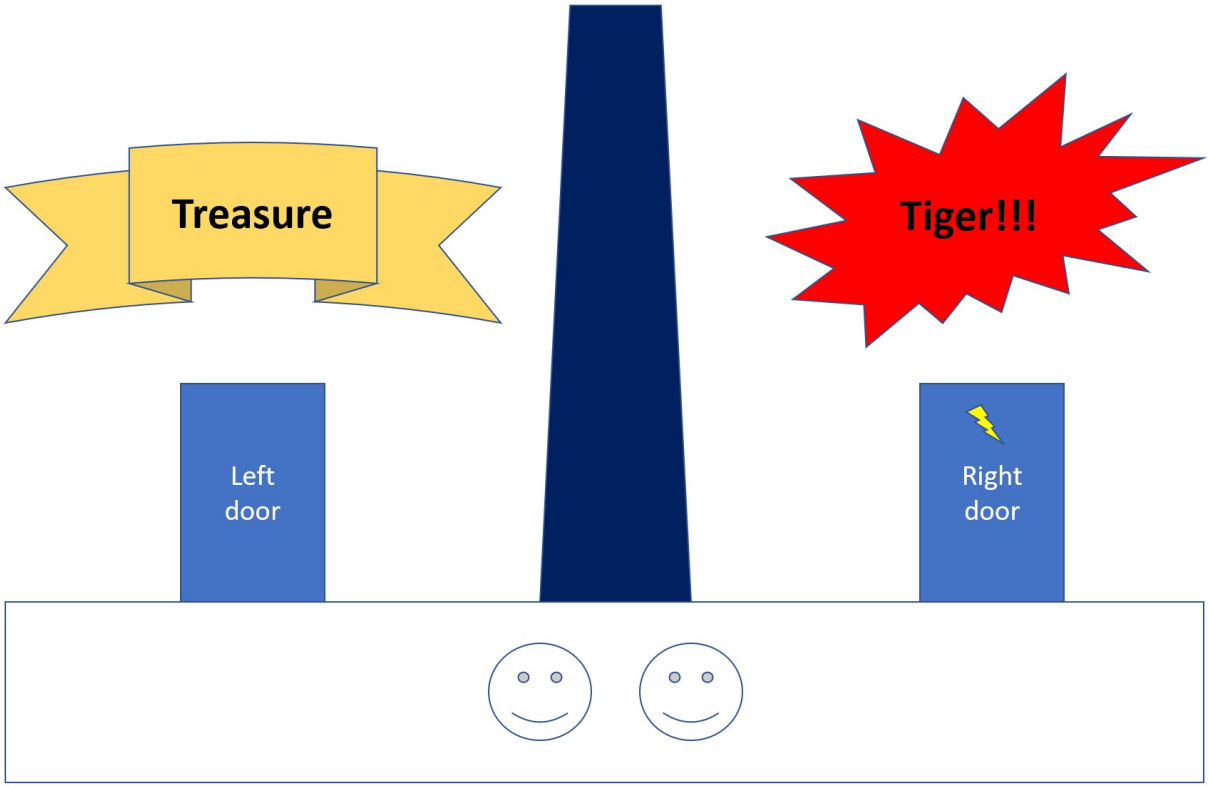


Figure 1: Dec-Tiger problem setting. Two agents need to jointly decide which of the doors to open based on what they hear in a hallway. Opening the wrong door has fatal consequences. Opening the correct door will result in high rewards.

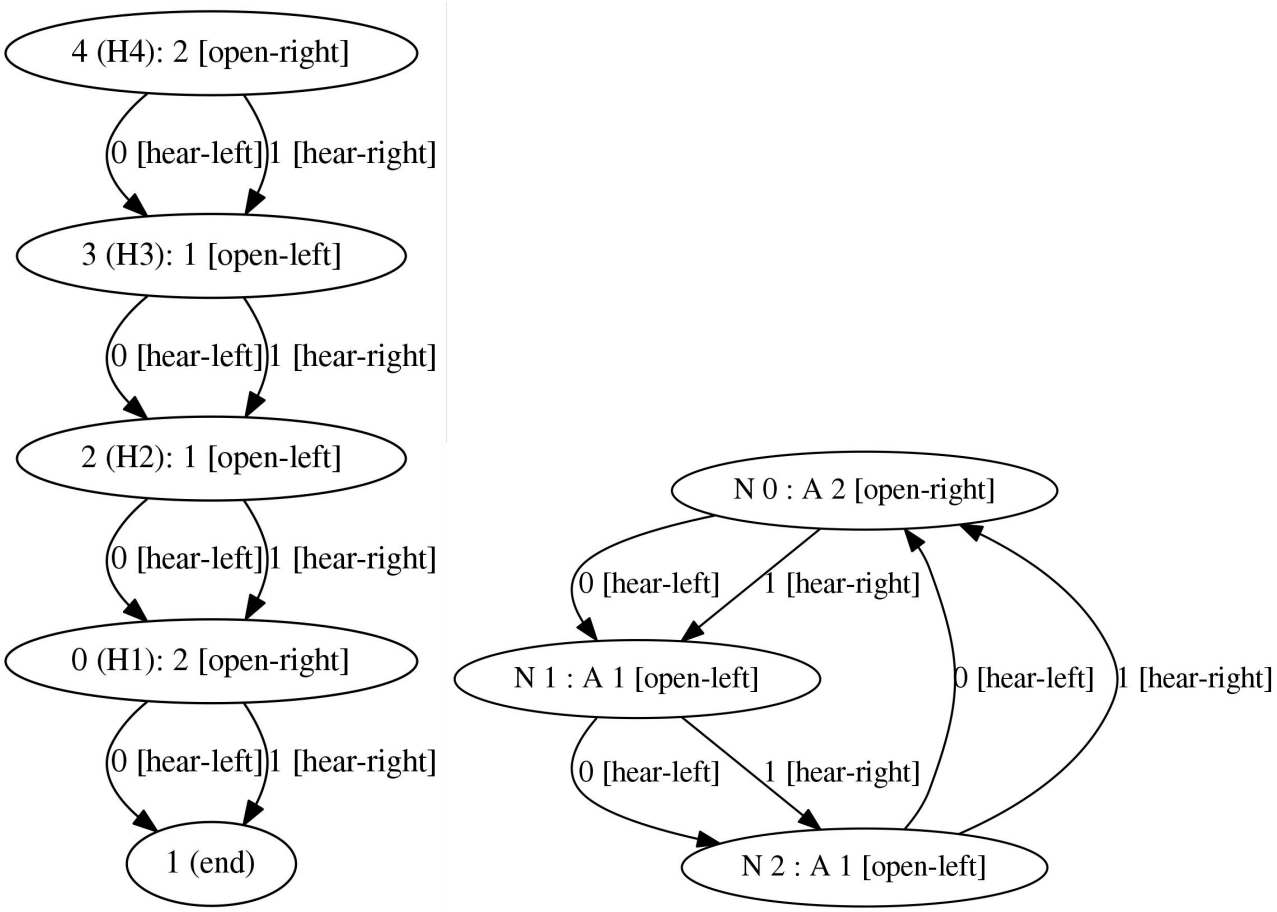
The rewards are shown in Table 1. Note that the highest reward is given when the agents open the correct door and they both decided to open it. A smaller positive reward is given when they open the correct door, but one of them actually decided to passively listen. The worst case is when the wrong door is opened, but yet one of the agents was willing to still listen. Such a case is even worse than if both decided to open the wrong door. This intuitively makes sense because the purpose is to encourage consensus between the agents.

Table 1: Rewards of the Dec-Tiger problem. $State_{left}/State_{right}$ means that the tiger is behind the left/right door respectively.

$Action$	$State_{left}$	$State_{right}$
$\langle listen, listen \rangle$	-2	-2
$\langle listen, open\ left \rangle$	-101	9
$\langle listen, open\ right \rangle$	9	-101
$\langle open\ left, listen \rangle$	-101	9
$\langle open\ left, open\ left \rangle$	-50	20
$\langle open\ left, open\ right \rangle$	-100	-100
$\langle open\ right, listen \rangle$	9	-101
$\langle open\ right, open\ left \rangle$	-100	-100
$\langle open\ right, open\ right \rangle$	20	-50

Figures 2 and 3 present two equivalent policies: one designed for the G-DICE framework, and the other for the policy graph improvement algorithm (or PGI) by [2]. The purpose was to test that the new implementation of G-DICE calculates the correct policy value, by comparing it to an already established algorithm. The results are indeed correct and they asymptotically get closer to the true policy value the more rollout simulations are performed. At most 10000 simulations were run with an absolute error of 0.1, which is a small and satisfying error.

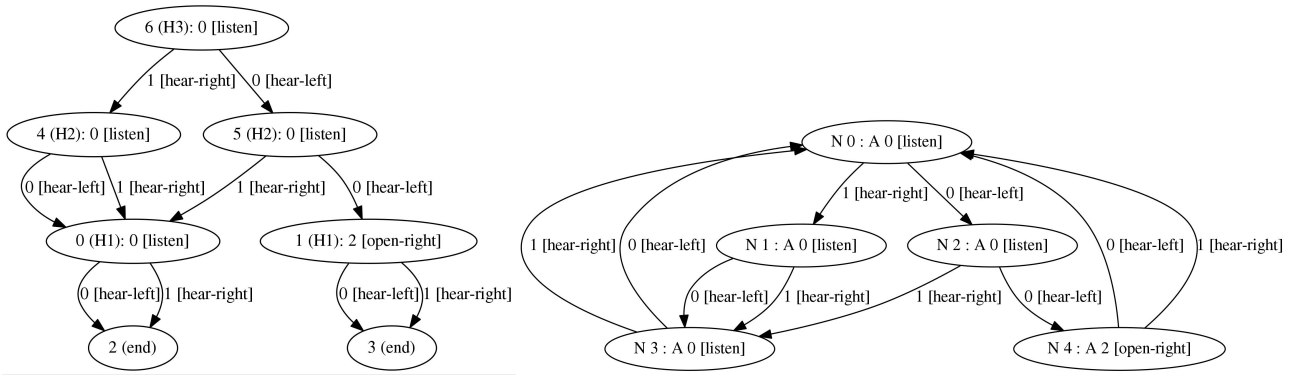
The first experiment has the peculiarity that the PGI algorithm has a horizon of 5 steps and also: 5 nodes. This same number of steps and nodes was possible because the branching factor was set to 1. The G-DICE graph on the other hand has a time horizon of 3 and just 3 nodes. The second experiment has the same number of nodes for both algorithms and the same horizon of 3 steps. For this to happen, the branching factor of PGI was set to a value of 2.



(a) PGI policy graph

(b) G-DICE policy graph

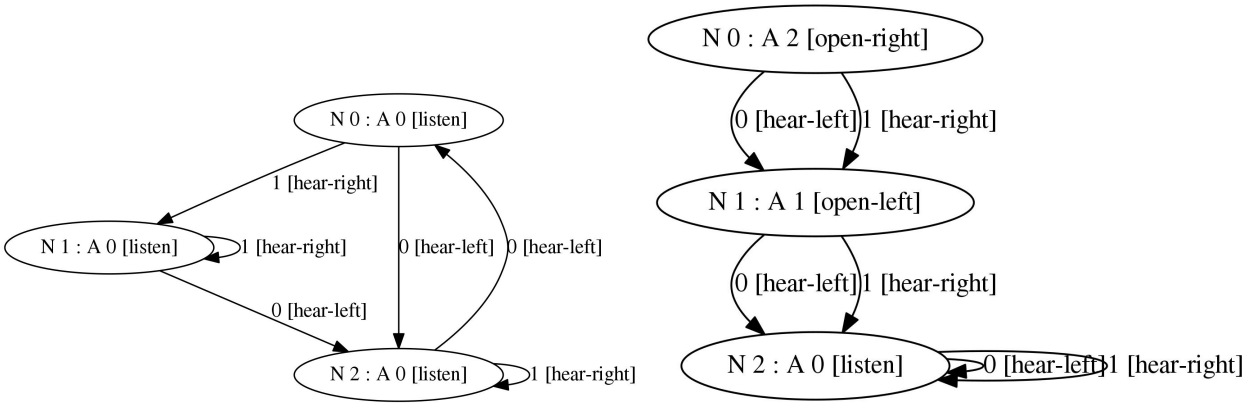
Figure 2: These two policies for the Dec-Tiger problem are equivalent. The validation experiment was meant to compare the newly implemented G-DICE algorithm against the established policy graph improvement (short PGI) algorithm from [2]. They were ran for 10000 simulations and resulted in similar values of -57. This means that the new implementation of G-DICE is calculating correct Bellman values.



(a) PGI policy graph (b) G-DICE policy graph

Figure 3: These two policies for the Dec-Tiger problem are equivalent. The validation experiment was meant to compare the newly implemented G-DICE algorithm against the established policy graph improvement (short PGI) algorithm from [2]. They were ran for 10000 simulations and resulted in similar Bellman equation values in the interval between 0.11 and 0.18. This means that the newly implemented G-DICE algorithm calculates Bellman equation values correctly.

The newly implemented G-DICE algorithm was further tested and an optimal policy was found for a horizon of 2 time steps, which according to `http://masplan.org/optimal_values` has the value -4. The policy graphs for each agent are presented in Figure 4.



(a) G-DICE policy graph of agent 0 (b) G-DICE policy graph of agent 1

Figure 4: During this experiment the expectation was to see if the newly implemented G-DICE algorithm can calculate the optimal policy for the Dec-Tiger problem with a horizon of 2. The number of graph nodes was set to 3. After 10000 simulations the algorithm created these two policies with a value of -4, which is indeed the optimal value for such a problem (see `http://masplan.org/optimal_values`).

References

[1] Shayegan Omidshafiei, Christopher Amato, Miao Liu, Michael Everett, Jonathan P How, and John Vian. Scalable accelerated decentralized multi-robot policy search in continuous observation spaces. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 863–870. IEEE, 2017.

[2] Mikko Lauri, Joni Pajarinen, and Jan Peters. Information gathering in decentralized pomdps by policy graph improvement. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1143–1151. International Foundation for Autonomous Agents and Multiagent Systems, 2019.

[3] Ranjit Nair, Milind Tambe, Makoto Yokoo, David Pynadath, and Stacy Marsella. Taming decentralized pomdps: Towards efficient policy computation for multiagent settings. In *IJCAI*, volume 3, pages 705–711, 2003.