Graphical Partially Observable Monte-Carlo Planning

Julius Pfrommer
Fraunhofer IOSB
76131 Karlsruhe
julius.pfrommer@iosb.fraunhofer.de

Abstract

Monte-Carlo Tree Search (MCTS) techniques are state-of-the-art for online planning in Partially Observable Markov Decision Problems (POMDP). The recently proposed Factored-Value Multiagent POMCP (FV-MPOMCP) algorithm improves on the scalability of MCTS in Multiagent POMDP (MPOMDP) environments by estimating several Q-values, each considering a subset of the actions and observations, and combining these Q-values via Variable Elimination. However, in MPOMDP, only the cumulated reward for each step is known, with no insight on the reward structure. In this work, we additionally exploit the structure of reward that decomposes into local reward terms. The proposed Graphical Partially Observable Monte-Carlo Planning (GPOMCP) algorithm combines Monte-Carlo Tree Search with a variation of the message passing algorithm (Max-Sum) known from Graphical Probabilistic Models and Distributed Constraint Optimization.

1 Introduction

Partially Observable Markov Decision Processes (POMDP, [10]) are models for sequential decision making under uncertainty where the latent world state can only be estimated from observations. The Partially Observable Monte-Carlo Planning (POMCP, [20]) algorithm is based on Monte-Carlo Tree Search (MCTS, [16]) and currently state-of-the-art for online POMDP planning. However, POMCP still suffers from the curse of dimensionality, especially prevalent in large multiagent settings, as the cardinality of the action- and observation-space grows exponentially in the number of agents. The recently proposed FV-MPOMCP algorithm [3] adapts POMCP for the Multiagent POMDP (MPOMDP, [15]) setting. Instead of estimating the Q-value for the joint history and actions, so-called experts each estimate a separate Q-value with a limited scope for observations and actions. Actions are selected by combining the experts Q-values. The so-called locality of interaction between agents [18] is usually associated with a reward function that decomposes into local reward terms, each depending only on a subset of the state and action variables. But in MPOMDP, only the cumulated reward of each step becomes known. This makes it difficult to infer the causal effect of actions, as the resulting local reward is merged with the reward "noise" generated by actions that are out of scope for the local Q-value that is to be estimated.

2 Background

2.1 POMDP and Distributed POMDP

POMDP can be represented as tuples $\langle \mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{R}, \mathbb{P}_S, \mathbb{P}_O, \mathbb{P}^0_S, \gamma \rangle$, where the set $\mathcal{S}$ contains all possible world states and the initial state s^0 is drawn from the distribution $\mathbb{P}^0_S$. In every step, an action $a \in \mathcal{A}$ is chosen, leading to a new state s' according to the state transition distribution $\mathbb{P}_S(s' \mid s, a)$.

30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

The latent world state can only be observed indirectly. Observations $o \in \mathcal{O}$ are made according to the distribution $\mathbb{P}_O(o \mid s', a)$. An immediate reward $r = \mathcal{R}(s, a)$ is generated from the initial state in the current step and the chosen action. Future rewards may be discounted with a factor $\gamma \in [0, 1)$. Otherwise, a finite number of steps is considered. A policy $\pi : \prod_{t=0}^{\infty}(\mathcal{A} \times \mathcal{O})^t \to \mathcal{A}$ maps the history of actions and observations after t episodes $h^t = (a^0, o^0, \ldots, a^{t-1}, o^{t-1})$ to the next action. The expected value of a policy is $V^\pi(h) = \mathbb{E}\left[\mathcal{R}(s, a) + \gamma V^\pi(hao) \mid h, a = \pi(h)\right]$ where hao denotes the concatenation of the selected action and resulting observation to the history h. The value of the optimal policy is given by Bellman's equation $V^*(h) = \max_a \mathbb{E}\left[\mathcal{R}(s, a) + \gamma V^*(hao) \mid h, a\right]$. The Q-value of action a after history h is $Q(h, a) = \mathbb{E}\left[\mathcal{R}(s, a) + \gamma V^*(hao) \mid h, a\right]$.

A general distributed POMDP is represented as $\langle I, \{\mathcal{S}_\sigma\}, \{\mathcal{A}_\alpha\}, \{\mathcal{O}_\omega\}, \{\mathcal{R}_\rho\}, \mathbb{P}_S, \mathbb{P}_O, \mathbb{P}_S^0, \gamma\rangle$. The state space factors into state variables $\sigma \in S$ so that $\mathcal{S} = \times_\sigma \mathcal{S}_\sigma$. Similarly, the actions and observations factor into variables $\alpha \in A$ and $\omega \in O$. The reward decomposes into reward terms $\rho \in R$ with the domain $\mathcal{R}_\rho : \mathcal{S}_\rho \times \mathcal{A}_\rho \to \mathbb{R}$ for $S_\rho \subseteq S, A_\rho \subseteq A$. The global reward function is vector-valued $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^{|R|}$ with the reward terms as components. The sum of reward terms $R' \subseteq R$ is written as $\mathcal{R}_{R'}(s, a) = \sum_{\rho \in R'} \mathcal{R}(s, a)_\rho$. The distributions $\mathbb{P}_S$, $\mathbb{P}_O$ and $\mathbb{P}_S^0$ are defined over the joint variables similarly to standard POMDP. Furthermore, let $i \in I$ denote the decision-making agents. Every agent has unique control over a (pairwise disjoint) set of action variables $A_i \subseteq A$. Agents each make observations $O_i \subseteq O$ and are notified about the reward generated in $R_i \subseteq R$. It is assumed that agents can exchange messages over noiseless communication channels with infinite bandwith to coordinate their actions. Agents also have perfect memory to store historical observations, received messages, policies, and so on. Based on this very permissive definition, additional limiting assumptions yield the different distributed POMDP models from the literature [15, 17, 18] with their respective complexity classes for computing policies with maximizum expected joint reward [6, 2].

2.2 Monte-Carlo Tree Search

Monte-Carlo Tree Search (MCTS, [16]) is a technique for sequential decision making under uncertainty. During the search phase, MCTS uses trial rollouts to iteratively improve an estimation of the Q-value $\widehat{Q}$. The current estimation is used to guide the actions selected during later rollouts. Here, a tradeoff needs to be made between exploration, searching in scenario branches from which only few samples have been taken so far, and exploitation, refining the strategy in a scenario branch with a good expected performance. The Upper-Confidence Bound (UCB) principle, originally developed for bandit-games [4], has become a popular choice to resolve the trade-off and its application to MCTS is known as UCT (Upper Confidence bounds applied to Trees) in the literature. In MDP, actions are selected as $\arg\max_a \widehat{Q}(sa) + c\sqrt{\log(n[s] + 1)/n[sa]}$. The counter n returns how often a scenario tree branch has been sampled in the past. The exploration/exploitation trade-off is tuned by the weighting parameter c. Lower c lead to an exploration strategy that takes only few samples from less promising branches. But all branches are sampled eventually, once they have been visited comparatively less often. UCT has led to huge performance increases in game-playing AIs, in particular for games with large branching factors, such as Go [8].

2.3 (Factored-Value) Partially Observable Monte-Carlo Planning

Partially Observable Monte-Carlo Planning (POMCP) is the application of UCT to POMDP settings [20]. The Q-value estimate $\widehat{Q}(ha)$ is simply the average of the total rewards received in the scenario tree branch ha, where rewards in later steps may be discounted. POMCP decreased the search time compared to existing POMDP planning algorithms over several orders of magnitude on the relevant benchmarks. Still, in multiagent settings, the cardinality of the joint actions and observations grows exponentially in the number of agents. So using communication to centrally solve the planning problem based on all actions and observations suffers from the curse of dimensionality. The Factored-Value Multiagent POMCP algorithm (FV-MPOMCP, [3]) improves the scalability of POMCP in multiagent settings. The underlying model is the Multiagent POMDP (MPOMDP, [19]), a variant of distributed POMDP where agents can communicate and the total reward (sum of reward terms) is made available to the agents in every step.

Definition 1. *Given a subset of actions and observations $B \subseteq A, P \subseteq O$, the marginal Q-value for the partial action assignment $a_B \in \mathcal{A}_B$ after the partial history $h_{BP} \in \prod_{t=0}^{\infty} (\mathcal{A}_B \times \mathcal{O}_P)^t$ is $\overline{Q}(h_{BP}, a_B) = \mathbb{E}\left[\max_{a \in \mathcal{A}:a_B} Q(h, a) \,|\, h_{BP}\right]$.*

The set $\{a \in \mathcal{A} : a_B\}$ contains the joint actions a agreeing with the partial assignment a_B for actions $\alpha \in B$. The definition assumes that, given a partial assignment a_B, the full history is available for choosing the remaining actions and the optimal joint policy is used for the remaining steps. Note that two kinds of marginalization are performed: Computing the expectation over joint histories compatible with the partial history and max-marginalization over the unassigned action variables.

FV-MPOMCP clusters agents into agents $e \subseteq I$ so that $e \in E$. It is assumed that the true Q-value decomposes to $Q(h, a) \approx \sum_{e \in E} Q_e(h, a_e)$ where a_e is the set of actions controlled by the agents that partake in expert e. This locality of interaction between agents [18] is then used to reduce the complexity of learning an estimation marginal Q-values for the global reward as $\widehat{Q}_e$. The authors propose two versions of FV-MPOMCP. In Factored-Statistics FV-MPOMCP, the experts estimate $\overline{Q}_e(h, a_e)$. In Factored-Tree FV-MPOMCP, the experts estimate $\overline{Q}_e(h_e, a_e)$ with a reduced history based on the visibility of actions and observations of the agents in e. In both versions, actions are selected as $\arg\max_{a \in \mathcal{A}} \sum_{e \in E} \widehat{Q}_e(\,\cdot\,, a_e) + c\sqrt{\log(n[\,\cdot\,] + 1)/n[\,\cdot\, a_e]}$ (the dot represents either the global or expert history) with the additional Upper Confidence Bound bias for exploration. Variable Elimination [9] is used to compute the best joint actions. The Q-value estimations are updated similarly to standard POMCP based on the visibility of actions and observations of the experts.

In settings with many agents, FV-MPOMCP shows drastic improvements in the speed of convergence over POMCP. This makes intuitive sense, as the large branching factor of the scenario tree forces POMCP to nearly always encounter a previously unvisited scenario branch. The Factored-Tree version of FV-MPOMCP scales to even larger scenarios as it avoids the curse of history. Factored-Tree FV-MPOMCP has however converged to lower-quality solutions than its Factored-Statistics counterpart on some benchmarks.

2.4 Advantage of knowing the reward structure

For MDP, it was shown that the sample complexity for learning the Q-value grows linearly in the number of local reward terms $|R|$ and in $\log |R|$ when the local rewards can be considered separately [5]. The global reward can be seen as a noisy "noisy measurement" for the local reward causally influenced by the individual actions. Fittingly, [7] uses a Kalman Filter to estimate the reward generated by local actions from the global reward. They experimentally show an improvement in the sample efficiency for Reinforcement Learning. Note that this approach is different from reward shaping in Reinforcement Learning [14], where a reward that is achieved only after a number of steps is decomposed into "progress indicators" to guide search.

3 Graphical Partially Observable Monte-Carlo Planning

Consider now a distributed POMDP model where observations and rewards may be visible to several agents, so the O_i and R_i are not necessarily disjoint. Importantly, the agents receive a vector of the rewards generated by R_i instead of a single scalar with the sum of rewards. Furthermore, we assume that actions may be visible to several agents, so the A_i may overlap as well. This visibility could be represented by an additional observation variable with no uncertainty. But we opted for a lighter notation for reasons that will become apparent later on. Agents can communicate at runtime, so the sharing of actions, observations and rewards can be enforced. We denote actions that both agents i and j observe as $A_{ij} = A_i \cap A_j$. Similarly, jointly observed rewards are $R_{ij} = R_i \cap R_j$. The neighborhood of i is defined as the agents with whom i shares at least one observed action $N(i) = \{j \in I \setminus i : A_{ij} \neq \varnothing\}$. Agents coordinate actions with their neighbors. It is not relevant which agent is actually executing an action once its value has been decided.

Definition 2. *Given a subset of the reward terms $R' \subseteq R$, the reduced Q-value for R' is $\widetilde{Q}_{R'}(h,a) = \mathbb{E}\left[\mathcal{R}_{R'}(s,a) + \gamma \widetilde{Q}_{R'}(hao, \arg\max_{b\in\mathcal{A}} Q(hao,b)) \,|\, h\right]$, the reward generated by R' assuming optimal action selection with regards to all reward terms R.*

Even though only a subset of the reward terms is considered, actions are still chosen according to the global Q-value. It is easy to show that $Q(h,a) = \sum_{W\in U} \widetilde{Q}_W(h,a)$ for a decomposition of the reward terms into disjoint sets $R = \bigcup_{W\in U} W$.

Assume now that a policy π can be conditioned by pre-assigning partial actions $a' \in \mathcal{A}_{A'}$. The resulting policy is denoted as $\pi_{a'}$.

Definition 3. *For a fixed joint policy π and observed history h_i of agent i, the marginal reduced value for i is $\widetilde{V}_i^{\pi}(h_i) = \mathbb{E}\left[\mathcal{R}_i(s,a) + \gamma \widetilde{V}_i^{\pi}(h_i a_i o_i) \,|\, a = \pi(h), h_i\right]$. The marginal reduced Q-value for a partial assignment a_i is $\widetilde{Q}_i^{\pi}(h_i, a_i) = \mathbb{E}\left[\mathcal{R}_i(s,a) + \max_{a'_i\in\mathcal{A}_i} \gamma \widetilde{Q}_i^{\pi}(h_i a_i o_i, a'_i) \,|\, a = \pi_{a_i}(h), h_i\right]$.*

Graphical Partially Observable Monte-Carlo Planning (GPOMCP) estimates the $\widetilde{Q}_i$ for the agents based on trial rollouts. However, no fixed policy π exists (see Definition 3) since all agents update their policies after every rollout. The joint policy for action selection is recovered from the local estimations using a variation of the message passing algorithm based on the Generalized Distributive Law [1] for the Max-Plus semiring. Differently from inference in probabilistic models, the messages represent reduced marginal Q-values instead of probability distributions. Besides [3], to the best of our knowledge, previous work on planning and reinforcement learning in distributed POMDP has considered either message-passing [24, 23] or action selection according to the Upper Confidence Bound principle [11, 21], but not both.

Assume now that the agent neighborhood relations form a tree $H = (I, E)$ with agents as vertices and neighborhood relations as edges $E = \{(i,j) \in I^2 : j \in N(i)\}$ and that agents only share access to a reward variable if they are neighbors $j \notin N(i) \Rightarrow R_{ij} = \varnothing$. These assumptions will be relaxed later on. On a tree, there exists an ordering of the agents $\sigma : I \to \{1, \ldots, |I|\}$, where every agent has at most one neighbor with a lower index number. The ordering is used to construct a forward/backward message passing schedule Γ. An agent i can only send a message to a neighbor $j \in N(i)$ when it has received a messages from all other neighbors $N(i) \setminus j$. This schedule starts at the leaf nodes and propagates throughout the tree until a message has been sent over every (directed) edge (i,j). If the scenario induces a tree-structure of the agents, the CONSTRUCTTREESCHEDULE procedure returns a valid forward/backward schedule. Other possible schedules are discussed after introducing the main algorithm.

The procedure GPOMCP takes the initial joint history h^0 as input. Similar to POMCP, GPOMCP performs a serious of rollouts to iteratively improve the Q-value estimations. For this GPOMCP needs access to a generative model that lets it draw states s according to a belief distribution for the history $B(h)$. The belief for the empty history $\varnothing$ is $B(\varnothing) = \mathbb{P}_S^0$. In POMCP, the number of considered steps T is taken such that $\gamma^T < \epsilon$ for convergence to an ϵ-solution. While we have no proof of convergence for GPOMCP, taking a similar history length is advised.

The UPDATE procedure takes the history of a complete rollout (including rewards) as input. First, we increase a counter how often a certain history h_i^t was visited by each agent i. Furthermore, the generated reward is abbreviated as $r = \mathcal{R}(s,a)$. The sum of rewards visible to agent i is r_i and the sum of rewards visible to both agents i and j is r_{ij}. The reward from the current and later steps is aggregated in ρ_i. This is in turn aggregated in Q_i as the average reward that has been observed for choosing a_i^t after an observed prior history h_i^t. Similar statistics are kept for reward observed by both i and j.

Procedure SELECTACTION takes as input the current history and the exploration weight c. The latter is set to zero for the final action selection. First, the agents update the message sent to their neighbors according to the schedule Γ. The message m_{ij} sent from agent i to j can be seen as a tabular representation of a function $m_{ij} : \mathcal{A}_{ij} \to \mathbb{R}$. On a tree-structure H, m_{ij} represents the expected reward in the current and future steps within the sub-tree behind the edge (i,j) conditioned on the jointly visible action a_{ij}. To compute the messages in lines 5 and 6, the agents first select the best

Algorithm 1 The GPOMCP algorithm

```
1: procedure GPOMCP(h^0, c)
2:     Γ ← ConstructSchedule
3:     for i ∈ I do
4:         Q_i[·] ← 0, n_i[·] ← 0
5:         Q_ij[·] ← 0,   ∀j ∈ N(i)
6:     while enough time do
7:         h ← h^0
8:         s ~ B(h^0)
9:         for t ∈ {1, ..., T} do
10:            a ← SelectAction(h, c)
11:            s' ~ P_S(s, a), o ~ P_O(s', a),
               r ← R(s, a), h ← haor
12:            s ← s'
13:        Update(h)
14:    return SelectAction(h^0, 0)
```

```
1: procedure SelectAction(h^t, c)
2:     for i ∈ I, j ∈ N(i) do
3:         m_ij[·] ← 0
4:     for (i, j) ∈ Γ, a_ij ∈ A_ij do
5:         b_i ← argmax_{d_i ∈ A_i : a_ij} [Q_i[h_i^t d_i] + Σ_{j ∈ N(i)} m_ji[d_ij]]
6:         m_ij[a_ij] ← Q_i[h_i^t b_i] − Q_ij[h_i^t b_i] + Σ_{k ∈ N(i)\j} m_ki[b_ik]
7:     a ← ∅^{|A|}, A' ← ∅
8:     for i ∈ I do
9:         if ∃d_i ∈ {A_i : a_A'}, n[h_i^t d_i] = 0 then
10:            a_i ~ U({d_i ∈ {A_i : a_A'}, n[h_i^t d_i] = 0})
11:        else
12:            a_i ← argmax_{d_i ∈ A_i : a_A'} [Q_i[h_i^t d_i] + Σ_{j ∈ N(i)} m_ji[d_ij]
                   + c sqrt(log(n[h_i^t]+1) / n[h_i^t d_i])]
13:        A' ← A' ∪ A_i
14:    return a
```

```
1: procedure ConstructTreeSchedule
2:     Γ ← ∅
3:     for i ∈ {σ^{-1}(1), ..., σ^{-1}(|I|)}
4:         for j ∈ N(i) : σ(j) > σ(i)
5:             Γ ← Γ ∪ (i, j)
6:     for i ∈ {σ^{-1}(|I|), ..., σ^{-1}(1)}
7:         for j ∈ N(i) : σ(j) < σ(i)
8:             Γ ← Γ ∪ (i, j)
9:     return Γ
```

```
1: procedure Update(h^T)
2:     ρ_i ← 0,   ∀i ∈ I
3:     ρ_ij ← 0,   ∀i ∈ I, j ∈ N(i)
4:     for t ∈ {T − 1, ..., 0} do
5:         for i ∈ I do
6:             n_i[h_i^t] ← n_i[h_i^t] + 1
7:             n_i[h_i^t a_i^t] ← n_i[h_i^t a_i^t] + 1
8:             ρ_i ← γρ_i + r_i^t
9:             Q_i[h_i^t a_i^t] ← Q_i[h_i^t a_i^t] + (ρ_i − Q_i[h_i^t a_i^t]) / n[h_i^t a_i^t]
10:            for j ∈ N(i) do
11:                ρ_ij ← γρ_ij + r_ij^t
12:                Q_ij[h_i^t a_i^t] ← Q_ij[h_i^t a_i^t] + (ρ_ij − Q_ij[h_i^t a_i^t]) / n[h_i^t a_i^t]
```

estimated action given the partial assignment a_{ij}. For this, expected reward that is not directly visible to i is considered in the form of the received messages. But for computing the message entry, the message coming from j is ignored and reward that is visible to both i and j is removed. This ensures that every reward variable is considered only once in the action selection after the message passing schedule has finished. Action selection is performed in lines 7 to 13. The joint action assignment is initialized with an empty sentinel value for every component. Then, the agents select the best action assignment within their scope, in accordance to the previously fixed partial assignment $a_{A'}$ with an additional Upper Confidence Bound bias for exploration. Actions that were not previously taken have precedence and are sampled from a uniform distribution.

Proposition 1. *If the agent neighborhoods form a tree-structure H, then FV-MPOMCP can be recovered by adding the Upper Confidence Bound bias term $c\sqrt{\log(n[h_i]+1)/n[h_i d_i]}$ also to the message computation in both lines 5 and 6 of* SelectAction *and assigning a unique reward value to every agent that takes on the globally generated reward in every step.*

Proof. First, the agent definition in GPOMCP corresponds to the experts in FV-MPOMCP, where the agents in FV-MPOMCP are non-overlapping in terms of the visible actions. Second, by comparing the update mechanisms, it is easy to see that $Q_i[h_i^t a_i]$ in GPOMCP equals $Q_e(h_e^t, a_e)$ in FV-MPOMCP for the same observed histories. Third, even though all agents in GPOMCP receive the global reward, every agent accesses a unique reward variable. So there is no overlap in the visible rewards and the term $-Q_{ij}[h_i^t bi]$ for reducing overcounting is always zero. Fourth, message passing on a tree-structure yields the same assignments as Variable Elimination [12]. Including the Upper Confidence Bound bias in the messages, thus recovers the result of Variable Elimination over the biased Q-value estimations of the experts. □

Proposition 1 holds for both the Factored-Statistics and Factored-Tree versions of FV-MPOMCP. Their difference lies only in the observations visible to each agent (agents have access to all observations in the Factored-Statistics version). The assumption of tree-structured neighborhood relations can be relaxed. In that case, the CONSTRUCTTREESCHEDULE is replaced and, for example, returns a randomized schedule where several messages are passed over each directed edge. This approach is known as Loopy Belief Propagation and gives good results in practice although convergence is not guaranteed [22]. Message-passing schemes with guaranteed convergence on loopy graphs exist [13] but are not considered here.

4 Benchmark

For the benchmark, we adopt the Firefighting domain introduced in [18]. The model dynamics are identical to the original description and we omit a full reproduction due to lack of space. For FV-MPOMCP, we assign an expert to every firefighter (agent). Experts see the actions and observations of the (left and right) neighbors and receive the global reward after every step. So we consider the Factored-Tree version of FV-MPOMCP where experts estimate the Q-value for a partial history h_e. For GPOMCP, each firefighter is represented by an agent that receives the actions and observations of his left and right neighbors. Furthermore, the agents receive the reward generated at the houses they can potentially visit. For POMCP, a central controller has access to all actions and observations and the cumulative global reward. We benchmarked two scenarios with 5 houses (4 firefighters) and 10 houses (9 firefighters) respectively. Each scenario had a length of 10 undiscounted steps. Figure 1 shows the expected performance over 10 steps for a certain number of trial rollouts before each action selection. Each measurement was repeated 100 times with the initial state sample randomly drawn. All algorithms ran with an exploration weight factor of $c = 50$ for the scenario with 5 houses and $c = 100$ for the scenario with 10 houses.

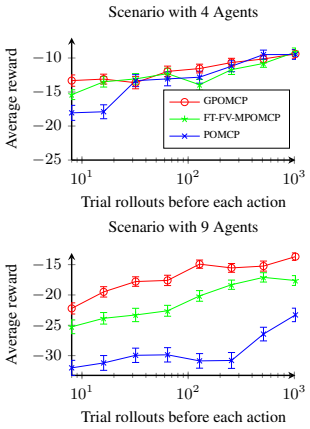

Figure 1: Benchmark results for the firefighting scenario averaged over 100 simulations.

The benchmark results show good performance of GPOMCP compared to POMCP and FV-MPOMCP on the larger scenario with 9 agents when information from only a few rollouts is available. On the small scenario with 4 agents, the differences between the algorithms become smaller than the standard error for more than 32 rollouts are made before each action selection. Our C++ implementation of GPOMCP is based on the original POMCP code by Silver and Veness and can be accessed at `https://github.com/jpfr/gpomcp`.

5 Summary

In this contribution, we introduced the Graphical Partially Observable Monte-Carlo Planning (GPOMCP) algorithm for planning in distributed POMDP. GPOMCP combines Monte-Carlo Tree Search with a message passing scheme to combine local estimations of so-called marginal reduced Q-values. The advantage over previous work (notably FV-MPOMCP) is that the reward structure is considered in the estimation. This gives better insight into the causal relations between actions and the different (local) reward terms. On a commonly used benchmark problem, GPOMCP was shown to compare favorably to previous work.

In their current form, FV-MPOMCP and GPOMCP still require a central component to run the trial rollout simulations. To enable truly distributed decision making, future work might extend GPOMCP for independent trial rollout simulations using a generative model of limited scope for each agent. A characterization of the convergence properties of GPOMCP and the sample complexity for estimating the reduced marginal Q-values is highly desirable and subject of ongoing work.

References

[1] Srinivas M Aji and Robert J McEliece. The generalized distributive law. *Information Theory, IEEE Transactions on*, 46(2):325–343, 2000.

[2] Christopher Amato, Girish Chowdhary, Alborz Geramifard, N Kemal Üre, and Mykel J Kochenderfer. Decentralized control of partially observable markov decision processes. In *52nd IEEE Conference on Decision and Control*, pages 2398–2405. IEEE, 2013.

[3] Christopher Amato and Frans A Oliehoek. Scalable Planning and Learning for Multiagent POMDPs. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[4] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256, 2002.

[5] Drew Bagnell and Andrew Y Ng. On local rewards and scaling distributed reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 91–98, 2005.

[6] Daniel S Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The complexity of decentralized control of markov decision processes. *Mathematics of operations research*, 27(4):819–840, 2002.

[7] Yu-han Chang, Tracey Ho, and Leslie P Kaelbling. All learning is local: Multi-agent learning in global reward games. In *Advances in Neural Information Processing Systems*, pages 807–814, 2004.

[8] Sylvain Gelly, Levente Kocsis, Marc Schoenauer, Michele Sebag, David Silver, Csaba Szepesvári, and Olivier Teytaud. The grand challenge of computer go: Monte carlo tree search and extensions. *Communications of the ACM*, 55(3):106–113, 2012.

[9] Carlos Guestrin, Daphne Koller, and Ronald Parr. Multiagent planning with factored mdps. In *NIPS*, volume 1, pages 1523–1530, 2001.

[10] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1):99–134, 1998.

[11] Jelle R Kok and Nikos Vlassis. Collaborative multiagent reinforcement learning by payoff propagation. *Journal of Machine Learning Research*, 7(Sep):1789–1828, 2006.

[12] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[13] Vladimir Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *IEEE transactions on pattern analysis and machine intelligence*, 28(10):1568–1583, 2006.

[14] George Konidaris and Andrew Barto. Autonomous shaping: Knowledge transfer in reinforcement learning. In *Proceedings of the 23rd international conference on Machine learning*, pages 489–496. ACM, 2006.

[15] Joao V Messias, Matthijs Spaan, and Pedro U Lima. Efficient offline communication policies for factored multiagent pomdps. In *Advances in Neural Information Processing Systems*, pages 1917–1925, 2011.

[16] Rémi Munos et al. From bandits to monte-carlo tree search: The optimistic principle applied to optimization and planning. *Foundations and Trends in Machine Learning*, 7(1):1–129, 2014.

[17] Ranjit Nair, Pradeep Varakantham, Milind Tambe, and Makoto Yokoo. Networked distributed pomdps: A synthesis of distributed constraint optimization and pomdps. In *AAAI*, volume 5, pages 133–139, 2005.

[18] Frans A Oliehoek, Matthijs TJ Spaan, Shimon Whiteson, and Nikos Vlassis. Exploiting locality of interaction in factored dec-pomdps. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 1*, pages 517–524. International Foundation for Autonomous Agents and Multiagent Systems, 2008.

[19] David V. Pynadath and Milind Tambe. The communicative multiagent team decision problem: Analyzing teamwork theories and models. *Journal of Artificial Intelligence Research*, 16:389–423, 2002.

[20] David Silver and Joel Veness. Monte-carlo planning in large pomdps. In *NIPS*, pages 2164–2172, 2010.

[21] Feng Wu, Shlomo Zilberstein, and Nicholas R Jennings. Monte-carlo expectation maximization for decentralized pomdps. In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, pages 397–403. AAAI Press, 2013.

[22] Jonathan S Yedidia, William T Freeman, and Yair Weiss. Constructing free-energy approximations and generalized belief propagation algorithms. *Information Theory, IEEE Transactions on*, 51(7):2282–2312, 2005.

[23] Chongjie Zhang and Victor Lesser. Coordinated multi-agent reinforcement learning in networked distributed pomdps. In *Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

[24] Xinhua Zhang, Douglas Aberdeen, and SVN Vishwanathan. Conditional random fields for multi-agent reinforcement learning. In *Proceedings of the 24th international conference on Machine learning*, pages 1143–1150. ACM, 2007.