Communicative Interactive Partially Observable Monte Carlo Planning

Sarit Adhikari, Piotr Gmytrasiewicz
University of Illinois at Chicago
Chicago, Illinois 60607
sadhik6@uic.edu, piotr@uic.edu

Abstract

Communicative Interactive POMDPs (CIPOMDPs) provide a ToM (Theory of Mind) approach to interaction and communication among agents in a partially observable stochastic environment. The sophistication of nested opponent modeling comes at a high computation cost. Monte Carlo simulations provide a highly efficient technique for both tree search and belief state updates. As POMCP has been shown to scale well to POMDP problems with large state spaces, we adopt the technique to communicative interactive POMDPs. The approach provides scalability for a higher time horizon when the number of interactive states explodes in size and significantly improves time for policy computation compared to the offline point-based approach.

Introduction and Background

The theory of mind approach to interaction and communication is important in both collaborative and deceptive settings. When it comes to human-machine teams, not only explicitly modeling beliefs, capabilities, and preferences of one another, but also communicating their beliefs might lead to better collective performance and higher reward. The single-agent frameworks cannot sufficiently account for nested opponent modeling structure and communication among the agents. CIPOMDP (Gmytrasiewicz 2020) is the first general framework for an autonomous self-interested agent to communicate and interact with other agents. A finitely nested communicative interactive POMDP of agent i in an environment with agent j is defined as:

$$CIPOMDP_i = \langle IS_{i,l}, A, M, \Omega_i, T_i, O_i, R_i \rangle \quad (1)$$

As in Interactive Partially Observable Markov Decision Processes (IPOMDPs) (Gmytrasiewicz and Doshi 2005), $IS_{i,l}$, A, Ω_i, T_i, O_i and R_i denote a set of interactive states, a set of actions, a set of observations, transition function, observation function and reward function respectively. M is a set of messages the agents can send to $(m_{i,s})$ and receive from $(m_{i,r})$ each other. Each message in M can be interpreted as a marginal probability distribution spanned on the agents' interactive state spaces IS_i (and IS_j). The model of agent j (θ_j) is a part of the interactive state of agent i. i.e. $is = (s, \theta_j)$; $is \in IS_i$. Belief update for CIPOMDP is defined analogous to IPOMDP where messages are treated as additional actions and observations. Like actions, messages are used to achieve valuable belief states.

The sophistication of nested modeling and communication comes at a high computation cost. The offline point-based solution technique (Adhikari and Gmytrasiewicz 2021), improves upon an exact solution but can only solve problems with a limited horizon. Also, since only maximizing value function is backed up from the previous time-step, the probability distribution over action, message pair cannot be calculated. Monte Carlo simulations provide a highly efficient technique for both tree search and belief state updates. As POMCP (Silver and Veness 2010) has been shown to scale well to POMDP problems with large state space, the communicative and interactive variant of POMCP should provide scalability for a higher time horizon when the number of interactive states explodes in size. Further, at each node of the tree, we have access to the utility function for each action, message pair. This allows us to compute the probability of sending each action message pair while taking bounded rationality into account. The bounded rationality is modeled using the quantal response equation

$$P(a_j, m_j|\theta_j) = \frac{exp[\lambda U_j(a_j, m_j)]}{\sum_{a_j, m_j} exp[\lambda U_j(a_j, m_j)]} \quad (2)$$

where λ is the rationality parameter. When λ is 0, the choice is random, when λ is infinity the soft max criterion becomes hard max. The utility U_j in equation 2 is defined by equation 3 except the max term, which makes U the function of action and message.

Value Function in CIPOMDPs

The utility of interactive belief of agent i, contained in i's type θ_i, is:

$$U_i(\theta_i) = \max_{(m_{i,s}, a_i)} \Big\{ \sum_{is \in IS} b_i(is) ER_i(is, m_{i,s}, a_i) + \quad (3)$$

$$+\gamma \sum_{(m_{i,r}, o_i)} P(m_{i,r}, o_i|b_i, a_i) \times$$

$$\times U_i(\langle SE_{\theta_i}(b_i, a_i, m_{i,s}, o_i, m_{i,r}), \hat{\theta}_i \rangle) \Big\}$$

Here, $b_i(is)$ denotes the belief ascribed to interactive state is . $ER_i(is, m_{i,s}, a_i)$ above is the immediate reward to i for sending $m_{i,s}$ and executing action a_i given the interactive state is. γ denotes the discount factor. SE is the belief state estimation function. The term $P(m_{i,r}, o_i|b_i, a_i)$ depends on i's observation and message it receives from another agent j and is calculated as follows:

$$P(m_{i,r}, o_i|b_i, a_i) = \sum_{is \in IS} b_i(is) \sum_{a_j} O_i(s, a_i, a_j, o_i) Pr(m_{i,r}, a_j|\theta_j) \tag{4}$$

Here the term $O_i(s, a_i, a_j, o_i)$ is observation function of i and is part of its model and $Pr(m_{i,r}, a_j|\theta_j)$ is calculated from equation 2.

Monte Carlo Tree Search

Monte Carlo Tree Search is an iterative best-first search algorithm where each iteration comprises four steps - selection, expansion, simulation, and backpropagation. Since the search tree is built incrementally in each iteration, a portion of the tree already exists in the memory. Selection is the process of traversing from the root node to a leaf node in the memory using a tree policy. The expansion adds a new child node to the tree based on the last action executed in the selection phase. If maximum depth (as determined by horizon of the game) is not reached, a Monte-Carlo simulation is performed till the maximum depth. This is done using a rollout policy (which can be as simple as random action selection) and is called the rollout step. Finally, the final utilities are propagated to all nodes along the path from the leaf node to the root. The step is called backpropagation.

(Silver and Veness 2010) introduced a partially observable variant called partially observable Monte Carlo planning (POMCP) which uses forward search from the current node to find an approximation of optimal value function and consequently the optimal action to take from the current node. The node corresponds to a history (h) or a belief state. The node also keeps track of the number of times the history has been visited $N(h)$ and the average return obtained from history h $V(h)$. The algorithm combines the ideas of upper confidence tree search for action selection and particle filtering for belief state update. Both of these procedures are handled by using the same set of Monte-Carlo simulations. The desirable properties like convergence to optimal value function of fully observable MCTS are inherited by POMCP as well. The POMCP algorithm is shown to achieve high performance in the large POMDPs compared to traditional algorithms like value iteration which uses full-width computation of values in the search tree. Our work builds upon the POMCP algorithm and extends it for multi-agent and communicative settings. Similar to POMCP, we search through possible belief states of agent i, reachable by executing actions, receiving observations and exchanging messages.

Hence, the main contribution of the paper is the scalable sampling based monte-carlo tree search algorithm for interaction and communication among self-interested agents which improves upon the previous point-based approximation approach.

Algorithm 1 Communicative Interactive Monte Carlo Tree search

procedure SEARCH(h, I)
  repeat
    if $h = empty$ **then**
      $is \sim I$
    else
      $is \sim B(h)$
    end if
    $SIMULATE(is, h, 0)$
  until TIMEOUT()
  return $\arg\max_{b,m} V(hbm)$
end procedure

procedure SIMULATE($is, h, depth$)
  if $\gamma^{depth} < \epsilon$ **then**
    return 0
  end if
  if $h \notin T$ **then**
    for $a \in A, m \in M$ **do**
      $T(ham) \leftarrow (N_{init}(ham), V_{init}(ham), \emptyset)$
    end for
    return $ROLLOUT(is, h, depth)$
  end if
  $s \leftarrow is.state$
  $\theta_j \leftarrow is.model$
  $a_i^*, m_i^* \leftarrow \arg\max_{a_i, m_i} V(ha_i m_i) + c\sqrt{\frac{logN(h)}{N(ha_i m_i)}}$
  if $\theta_j \in cache$ **then**
    $a_j^* \leftarrow cache[\theta_j]$
  else
    if $\theta_j.level == 0$ **then**
      $a_j^* \leftarrow \theta_j.search(h_j, b_j)$
      $\mathcal{M} \leftarrow getMessageDistribution(M, b_j, \alpha)$
      $m_j \sim \mathcal{M}_{j,0}$
      $(s', o, r) \sim G_j(s, a_j^*)$
      $\theta_j' \leftarrow (h_j a_j^* o_j, \theta_j.frame)$
    else
      $a_j^*, m_{j,s} \leftarrow \theta_j.search(h_j, b_j)$
      $(s', o, r) \sim G_j(s, a_j^*, a_i)$
      $\theta_j' \leftarrow (h_j a_j^* m_{j,s} o_j m_{i,s})$
    end if
  end if
  $(s', o, r) \sim G_i(s, a_i^*, a_j^*)$
  $is' \leftarrow (s', \theta_j')$
  $R \leftarrow r + \gamma SIMULATE(is', ha_i^* m_i^* o_i^* m_j^*, depth + 1)$
  $B(h) \leftarrow B(h) \cup \{s\}$
  $N(h) \leftarrow N(h) + 1$
  $N(ha) \leftarrow N(ha) + 1$
  $V(ha) \leftarrow V(ha) + \frac{R - V(ha)}{N(ha)}$
  return R
end procedure

procedure ROLLOUT($is, h, depth$)
  if $\gamma^{depth} < \epsilon$ **then**
    return 0
  end if
  $a_i, m_i \sim \pi_{rollout}(h, .)$
  $a_j, m_j \sim \pi_{rollout}(h, .)$
  $(s', o, r) \sim G(s, a_i, a_j)$
  $is' \leftarrow (s', is.model)$
  return $r + \gamma ROLLOUT(is', ha_i m_i o_i m_j, depth + 1)$
end procedure

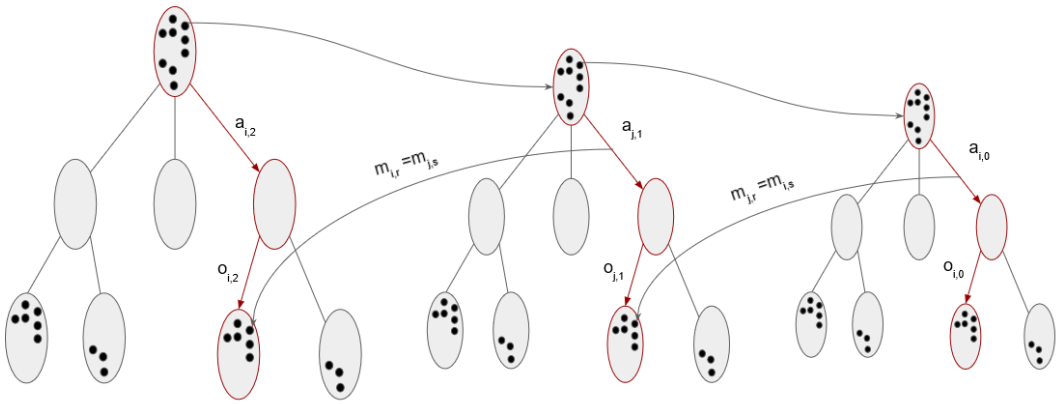

Figure 1: A part of the search space for the CIPOMCP agent i. The leftmost search tree corresponds to level 2 agent i . Each node of the tree consists of particles corresponding to interactive states. Each sample of the interactive state consists of a physical state and the model of the other agent j. The model of the other agent j implicitly contains the search tree (middle one in the figure) and is solved using CIPOMCP as $level > 0$. The rightmost search tree again corresponds to agent i as modeled by agent j. and is solved using POMCP since particles in the node correspond to the physical states.

Approach

As each interactive state consists of the model of other agents, the CIPOMCP search tree also consists of nested (CI)POMCP search trees for each model. Before the observation and transitioned states are obtained from the simulator, the action of the other agent needs to be known to pass to the simulator. The action and message from the other agent are obtained by using CIPOMCP if $l > 0$ and POMCP if $l = 0$, where l denotes strategy (or nesting) level of the agent. The history corresponding to the node in the search tree now consists of (action, message sent, observation, message received) across multiple time-steps. Figure 1 shows the nested structure for CIPOMCP. Each node corresponds to the collection of particles, each particle denoting a sample of interactive state. Hence, the particles corresponding to a node approximate the belief distribution over the interactive states. As the model for the sampled interactive state is solved, the action and message sent from the opponent are obtained. We avoid solving the same model again in each iteration by caching the action, message pair for each model.

Algorithm - CIPOMCP

Algorithm 1 shows steps for communicative interactive Monte Carlo tree search. The search in each iteration starts by sampling the interactive state from the history if history is not empty, else the interactive state is sampled from initial belief distribution (line 1-11). In simulate function (line 13-49), the search tree is constructed incrementally if the node corresponding to the current history doesn't exist. In the algorithm, T denotes the search tree, and $T(h)$ denotes the node in the tree corresponding to the history h. If the node already exists for the history, the action-message pair is selected by Upper Confidence Bound (UCB) (line 25).

$$V^{(+)}(ham) = V(ham) + c\sqrt{\frac{logN(h)}{N(ham)}} \tag{5}$$

Table 1: Tiger game friend reward function

$\langle a_i, a_j \rangle$	TL	TR
OR, L	9.5	-100.5
OL, L	-100.5	9.5
L, L	-1.5	-1.5
OR, OL	-40	-95
OL, OL	-150	15
L, OL	-51	4
OR, OR	15	-150
OL, OR	-95	-40
L, OR	4	-51

Then action message pair is selected by

$$\arg\max_{a,m} V^{(+)}(ham) \tag{6}$$

Line 29-39 involves recursive solving of search tree for the model in the current sample of interactive state if the model wasn't solved before. If the node doesn't have any child nodes, a rollout simulation is performed (line 51-60). Here as rollout policy, we sample action, message pair from a uniform distribution for both the agents.

Results

The results are reported on benchmark multi-agent tiger game with friend reward function (Table 1). In this setting, the agents face two doors, behind one door there's a hungry tiger and behind the other, there's a pot of gold. State is defined in terms of location of a tiger, TL denotes tiger left and TR denotes tiger right. In each time-step, the agents can open either door (OL denotes open left door and OR denotes open right door) or choose to listen (L). Opening the door behind gold and tiger gives positive and large negative rewards respectively. Information gathering action (listen) also has a small negative reward. If agents choose to

Table 2: Comparison of computation cost (in seconds) for a different number of iterations and size of message spaces for Level 1 CIPOMCP in cooperative multi-agent tiger game

$\|M\|$	iterations=500				iterations=1000				iterations=1500			
	horizon=4		horizon=5		horizon=4		horizon=5		horizon=4		horizon=5	
	Time	Reward	Time	Reward	Time	Reward	Time	Reward	Time	Reward	Time	Reward
12	13.24 $\pm$ 1.16	6.65 $\pm$ 11.68	15.85 $\pm$ 0.86	9.0 $\pm$ 13.69	22.53 $\pm$ 0.89	7.47 $\pm$ 10.79	27.84 $\pm$ 0.76	10.65 $\pm$ 10.01	43.28 $\pm$ 1.75	9.4 $\pm$ 2.2	51.54 $\pm$ 0.86	12.37 $\pm$ 4.26
22	13.73 $\pm$ 0.83	6.92 $\pm$ 11.96	18.23 $\pm$ 0.36	9.0 $\pm$ 10.14	29.70 $\pm$ 0.71	8.02 $\pm$ 8.42	38.48 $\pm$ 0.34	11.2 $\pm$ 7.85	45.76 $\pm$ 0.70	9.13 $\pm$ 2.38	55.92 $\pm$ 0.91	12.92 $\pm$ 3.76

Table 3: Comparison of computation cost (in seconds) for a different number of iterations and size of message spaces for Level 2 CIPOMCP in cooperative multi-agent tiger game

$\|M\|$	iterations = 500				iterations = 1000				iterations = 1500			
	horizon=4		horizon=5		horizon=4		horizon=5		horizon=4		horizon=5	
	Time	Reward	Time	Reward	Time	Reward	Time	Reward	Time	Reward	Time	Reward
12	608.63 $\pm$ 30.95	6.1 $\pm$ 13.42	1049.23 $\pm$ 62.15	8.45 $\pm$ 15.73	2080.14 $\pm$ 58.74	8.02 $\pm$ 8.42	3192.18 $\pm$ 204.43	11.2 $\pm$ 12.1	1315.00 $\pm$ 42.32	9.67 $\pm$ 1.96	1889.56 $\pm$ 54.59	12.85 $\pm$ 3.93
22	1198.69 $\pm$ 5.78	7.2 $\pm$ 11.97	2296.19 $\pm$ 79.43	7.9 $\pm$ 13.65	3541.38 $\pm$ 132.62	7.75 $\pm$ 10.79	6207.64 $\pm$ 231.46	11.75 $\pm$ 7.68	5650.40 $\pm$ 26.82	9.4 $\pm$ 2.2	14964.3 $\pm$ 219.16	13.4 $\pm$ 3.30

Table 4: Computation cost (in seconds) comparison between IPBVI-Comm and CIPOMCP

Level	Horizon	IPBVI-Comm	CIPOMCP
1	4	2256.63 $\pm$ 73.31	**20.45 $\pm$ 0.57**
	5	10799.55 $\pm$ 382.73	**23.96 $\pm$ 0.17**
2	4	2682.32 $\pm$ 20.73	**252.15 $\pm$ 4.86**
	5	13112.54 $\pm$ 65.29	**406.43 $\pm$ 2.75**

listen, they get growl indicative of the location of the tiger. The agents, while interacting with the environment, can also communicate their beliefs in each time-step which involves sending and receiving messages. The friend reward function is defined such that the agent gets its share of the reward and also half of the reward obtained by the other agent.

The table 4 shows that CIPOMCP significantly improves time for policy computation compared to offline point based approach (Adhikari and Gmytrasiewicz 2021). The table 2 and table 3 show the comparison of time of computation and reward collected in multi-agent cooperative tiger game for level 1 and level 2 CIPOMCP agents respectively for different iterations, discretization intervals and time horizons. The more number of iterations yields a better solution but at a higher computational cost. The results are averaged over 20 trials. The experiments were conducted on a machine with Intel Core i5 2GHz, 16GB RAM, and Windows OS.

Conclusion and Future Work

We presented an online sampling-based algorithm to compute policy for a theory of mind (ToM) agent acting in a stochastic and partially observable environment. We showed the online approach outperforms the offline point-based approximation algorithm in terms of computation time while achieving similar performance in a benchmark multi-agent tiger game.

As a future work, we want to leverage the online algorithm to study more interesting communicative and interactive scenarios between agents with varied preferences, strategy levels and bounded rationality. Further we want to explore other rollout policies which might provide more accurate estimate compared to random policy.

References

Adhikari, S.; and Gmytrasiewicz, P. 2021. Point Based Solution Method for Communicative IPOMDPs. In Rosenfeld, A.; and Talmon, N., eds., *Multi-Agent Systems*, 245–263. Cham: Springer International Publishing. ISBN 978-3-030-82254-5.

Gmytrasiewicz, P.; and Doshi, P. 2005. A Framework for Sequential Planning in Multiagent Settings. *Journal of Artificial Intelligence Research* 24: 49–79. Http://jair.org/contents/v24.html.

Gmytrasiewicz, P. J. 2020. How to Do Things with Words: A Bayesian Approach. *J. Artif. Intell. Res.* 68: 753–776. doi:10.1613/jair.1.11951. URL https://doi.org/10.1613/jair.1.11951.

Silver, D.; and Veness, J. 2010. Monte-Carlo Planning in Large POMDPs. In Lafferty, J.; Williams, C.; Shawe-Taylor, J.; Zemel, R.; and Culotta, A., eds., *Advances in Neural Information Processing Systems*, volume 23. Curran Associates, Inc.