A Partially Observable Monte Carlo Planning Algorithm Based on Path Modification

Qingya Wang WQY17NJU@163.COM
Feng Liu* FENGLIU@NJU.EDU.CN
Bin Luo LUOBIN@NJU.EDU.CN
National Key Laboratory for Novel Software Technology
Software Institute, Nanjing University, Nanjing, China

Editors: Berrin Yanıkoğlu and Wray Buntine

Abstract

Balancing exploration and exploitation has long been recognized as an important theme in the online planning algorithms for POMDP problems. Explorative actions on one hand prevent the planning from falling into the suboptimal dilemma, while hindering the convergence of the planning procedure on the other hand. Therefore, it is meaningful to maintain the exploration as well as taking a step forward towards exploitation. Note that there is a deviation between the action selection criteria in the planning procedure and in the execution procedure, which inspires us to build a bridge between these two criteria to accelerate the convergence. A Partially Observable Monte Carlo Planning algorithm based on Path Modification (POMCP-PM) is presented in the paper, which modifies the backtracing paths by considering the two criteria simultaneously when updating the values of parent nodes. The algorithm is general as the Upper Confidence Bound Apply to Tree (UCT) algorithm used to select actions can be easily replaced by other criteria. Experimental results demonstrate that POMCP-PM outperforms POMCP with varying numbers of simulations on several scenarios with different scales.

Keywords: POMDP; POMCP-PM; Value Updating

1. Introduction

The Partially Observable Markov Decision Processes (POMDPs) provide a general model for planning problems under uncertainty. However, solving the planning problems based on POMDPs exactly has been shown to be PSPACE-complete over finite horizons and undecidable over infinite horizons Papadimitriou and Tsitsiklis (1987); Madani et al. (1999). To alleviate the "curse of history" and the "curse of dimensionality" for POMDPs, many approximate algorithms have emerged over the past few decades. Point-based methods Pineau et al. (2003); Smith and Simmons (2004) greatly reduce the computational cost of solving POMDPs by planning on a set of representative belief points instead of the whole belief space. However, these algorithms still belong to offline settings in which the agent plans for all possible situations prior to execution. To further concentrate the computational resources and extend the application of POMDPs, online algorithms Ross et al. (2008) interleave planning with execution and only plan for current situations. Among online

* Corresponding author.

algorithms, Monte Carlo planning based ones can adapt to environmental changes without extra computation and are in widespread use. In Monte Carlo planning based algorithms, exploration and exploitation seem to be in a dilemma, and how to balance them well has long been a research hot spot.

Partially Observable Monte Carlo Planning (POMCP) Silver and Veness (2010) is one of the most popular online algorithms based on Monte Carlo Tree Search (MCTS) Browne et al. (2012). To balance exploration and exploitation, POMCP adopts the UCT algorithm Kocsis and Szepesvári (2006) to select actions, which takes both the value functions and the uncertainty into account. DESPOT Somani et al. (2013) constructs a determinized sparse partially observable tree which contains all the action branches to be more accurate, but limits its application in problems with large action space. SP-MCTS Schadd et al. (2008); Schadd (2011) modifies the UCT algorithm by adding a third term which represents the "possible deviation" of the node. POMCP-RAVE Liu et al. (2015) shares needed knowledge among actions and augments the action value in the UCT algorithm with All Moves As First (AMAF) heuristic method Helmbold and Parker-Wood (2009). Some recent researches Mern et al. (2021b); Mern et al. (2021a); Lim et al. (2021) design heuristic action selection strategies when the UCT algorithm is invalid in specific problems.

In fact, the online planning procedure can be further divided into two stages, which are the growing stage and the updating stage. The above algorithms all focus on finding a better action selection criterion in the growing stage, while ignoring the abundant information which can be utilized in the updating stage. To make full use of the information contained in current search tree, we find a new backtracing path to update the values of parent nodes in the updating stage and then propose an algorithm called Partially Observable Monte Carlo Planning based on Path Modification (POMCP-PM). The algorithm highlights the deviation between the action selection criteria in the planning procedure and in the execution procedure and divides the action nodes in the search tree into four categories based on their corresponding action selection criteria. The importance of each category determines its weight in the backtracing path during the updating stage. Instead of looking for another way to select actions, POMCP-PM provides a new perspective on balancing exploration and exploitation. Experimental results indicate that POMCP-PM shows better performance than POMCP under three scenarios with different scales. Furthermore, the new backtracing path raised in POMCP-PM does not conflict with the aforementioned algorithms, thus can be applied to a variety of online Monte Carlo planning algorithms.

The paper is organized as follows: Section 2 provides background references, Section 3 analyzes the importance of belief nodes, Section 4 presents the POMCP-PM algorithm, Section 5 shows the performance of the proposed algorithm over three scenarios with different scales. Finally, Section 6 concludes this paper and discusses our future work.

2. Background

2.1. POMDP Model

The POMDP model can be described by a 7-tuple $(S, A, O, T, Z, R, \gamma)$ Kaelbling et al. (1998), where S is the state set, A is the action set, and O is the observation set. The state-transition model $T(s'|s, a)$ and the reward function $R(s, a)$ specify the probability distribution of successor state s' and immediate reward r received after taking action a at

state s. The observation model $Z(o|s', a)$ gives the probability distribution of observation o received in successor state s' after action a has been taken. $\gamma \in [0, 1)$ is the discount factor.

In partially observable settings, the agent is not able to know the real state. Instead, it receives observations from the environment and then selects the next action according to the history sequence $\{a_0, o_1, a_1, o_2, \ldots, a_{t-1}, o_t\}$ Smallwood and Sondik (1973). However, it is of high complexity to maintain such history sequences. To simplify the model, belief state b is introduced to take the place of the history sequence, b is defined as

$$b_t(s) = P(s_t = s|o_t, a_{t-1}, \ldots, o_1, a_0). \tag{1}$$

The value set of b is defined as the belief space B. The belief state b allows the use of dynamic planning techniques for fully observable Markov Decision Processes. Initially, the probability distribution of the state in POMDP is b_0, then the following belief state b can be computed incrementally according to the Bayesian formula

$$b_a^o(s') = \frac{\sum_s b(s)T(s, a, s')Z(s', a, o)}{\sum_s b(s) \sum_{s'} T(s, a, s')Z(s', a, o)}. \tag{2}$$

The value function of belief state b is defined as the expectation of cumulative discounted reward. The policy in POMDP is a mapping from beliefs to actions: $\pi(b) \to a$. Given a policy π, the value function of belief state b is represented as follows

$$V_\pi(b) = \mathbb{E}\left[\sum_{k=t}^{T-1} \gamma^{k-t} R\left(b_k, \pi(b_k)\right) |\pi, b_t = b\right], \tag{3}$$

the action-value function is defined as follows

$$Q_\pi(b, a) = R(b, a) + \gamma \sum_{o \in O} p(o \mid b, a) V_\pi(b_a^o). \tag{4}$$

The goal of solving POMDP is to find the optimal policy π^* which optimizes the value function. The optimal value function satisfies the following equation

$$V^*(b) = \max_{a \in A} \left\{ R(b, a) + \gamma \sum_{o \in O} p(o|b, a) V^*(b_a^o) \right\}, \tag{5}$$

the optimal action-value function satisfies

$$Q^*(b, a) = R(b, a) + \gamma \sum_{o \in O} p(o|b, a) V^*(b_a^o), \tag{6}$$

and the optimal policy is

$$\pi^*(b) = \arg\max_{a \in A} Q^*(b, a). \tag{7}$$

Algorithm 1 PLAN

procedure PLAN(b)
  for $i \in 1:n$ **do**
    $s \leftarrow$ sample from b
    SIMULATE(s, b, d_{max})
  return $\arg\max_a Q(b, a)$

2.2. POMCP Algorithm

Monte Carlo method has a long history within numerical algorithms and has also been successfully applied to reinforcement learning in recent years. It works by taking random samples in the decision space and computing over the samples based on probability and statistical theory. POMCP is a widely used online algorithm following the framework of Monte Carlo planning, which is a combination of Monte Carlo sampling and tree search. The PLAN procedure is shown in Algorithm 1.

In POMCP, to estimate local reachable belief space, there is a PLAN procedure which involves n simulations before each execution. A Monte Carlo search tree is constructed to record the results of simulations. In the search tree, the action level interleaves with the observation level. A new leaf node will be added to the Monte Carlo tree and the value functions of nodes along the simulation path will be updated once a simulation terminates. Each simulation starts with state s sampled from current belief node b according to the probability distribution and ends with terminal states or the maximum depth d_{max}. The detailed POMCP_SIMULATE procedure is shown in Algorithm 2.

The POMCP_SIMULATE procedure terminates when d equals to zero, which means that the maximum depth d_{max} is reached. Otherwise, the action a to be simulated is selected by the UCT algorithm, then the observation o, the immediate reward r and the successor state s' will be generated by the black-box function $G(s, a)$ according to the POMDP model. h represents the history sequence corresponding to the current belief node

Algorithm 2 POMCP_SIMULATE

procedure POMCP_SIMULATE(s, h, d)
  if $d = 0$ **then**
    return 0
  $a \leftarrow \arg\max_{a \in C(h)} \left\{ Q(h_a) + c\sqrt{\frac{\log[N(h)]}{N(h_a)}} \right\}$
  $s', o, r \leftarrow G(s, a)$
  $M(h_{ao}) \leftarrow M(h_{ao}) + 1$
  if $M(h_{ao}) = 1$ **then**
    return $r + \gamma$ROLLOUT($s', h_{ao}, d-1$)
  $total \leftarrow r + \gamma$POMCP_SIMULATE($s', h_{ao}, d-1$)
  $N(h) \leftarrow N(h) + 1$
  $N(h_a) \leftarrow N(h_a) + 1$
  $Q(h_a) \leftarrow Q(h_a) + \frac{total - Q(h_a)}{N(h_a)}$
  return $total$

b, h_a corresponds to the history sequence when action a has been taken from the current belief node b, and h_{ao} corresponds to the history sequence when action a has been taken and observation o has been received from the current belief node b. $M(h_{ao})$ records the occurrence number of h_{ao}. If $M(h_{ao}) = 1$, which means the first time that some action or observation occurs, then a new action or observation node will be added to the search tree, otherwise the existing node will be reused. By performing a rollout using a baseline policy, the value of the new added leaf node can be obtained. Other nodes will call the POMCP_SIMULATE procedure recursively to get the values of child nodes and then update their own value functions using values of child nodes and immediate rewards. Then one is added to both the counts of h denoted as $N(h)$ and the counts of h_a denoted as $N(h_a)$. $Q(h_a)$ denotes the action-value function, which records the expected value function of h_a.

3. Analysis of the Importance of Belief Nodes

3.1. Action Sampling Strategies

In the planning procedure, in order to balance exploration and exploitation, POMCP selects actions to be simulated through the UCT Algorithm. We call the action selection criterion as behavior policy, and the policy is represented as follow

$$a_{plan} = \arg\max_a \left\{ Q(b,a) + c\sqrt{\frac{\log\left[N(b)\right]}{N(b,a)}} \right\}. \tag{8}$$

The first term of this formula represents the action value function of the current node, which means the exploitation of existing results. The second term reflects the relative relationship between the total visited times of the belief point and the visited times of a given action. If the relative visited times of the action is low, then the value of the second term will be large. It means that the uncertainty of the action value function is high and more explorations should be conducted on this action. The coefficient c is responsible for keeping the balance between exploration and exploitation. The larger the coefficient c is, the more it inclines to exploration. POMCP uses the behavior policy to select actions to be simulated, combined with the observations generated by the black box function, to construct more promising subtrees in the growing stage of the tree search method.

In the execution procedure, the agent adopts action selection criterion that directly selects the action corresponding to the maximum action value, which is different from the one in the planning procedure. We call the action selection criterion as target policy, and the policy is represented as follow

$$a_{act} = \arg\max_a Q(b,a). \tag{9}$$

For a given belief node, the behavior policy and the target policy may point to the same action or two different actions. According to this feature, action nodes in the search tree can be divided into four categories. The action nodes in four categories correspond to different levels of importance in the planning procedure.

3.2. Importance of Belief Nodes

In the Monte Carlo search tree, action nodes can be divided into the following four categories according to their importance, where actions in the first two categories are contained in the simulation trajectories and those in the last two categories are not.

The first category of action nodes is related to both the behavior policy and the target policy. As the nodes correspond to the target policy, their action value functions are the largest ones. In the updating stage, the belief values will be updated to be the maximum of the action values, which is consistent with the criterion of the target policy. As action nodes in the first category also correspond to the behavior policy, the action values of these nodes possibly have high uncertainty. Exploring these nodes and their subtrees can help determine their values more accurately, providing more information for the parent nodes. Therefore, nodes in this category own the highest level of importance.

The second category of action nodes is related to the behavior policy rather than the target policy, which is the main source of the deviation between the policy in planning procedure and the policy in execution procedure. The values of these action nodes are not the largest ones and the actions corresponding to them are explorative. Therefore, in many cases, only low rewards can the agent obtain after taking these actions. However, the explorative actions can help update the potential substrees, bringing significant value to overcome the suboptimal dilemma and obtain the global optimal policy, thus own a certain degree of importance.

The third category of action nodes is related to the target policy rather than the behavior policy. On the contrary to the action nodes in the second category, the values of these action nodes are the largest with low uncertainty, thus it is not so meaningful to explore it again within the limited planning time. But for the update of the parent nodes, these action nodes can provide important information about the current optimal action value functions. Therefore, this category of action nodes has a certain degree of importance as well.

Other nodes in the Monte Carlo search tree can be included into the fourth category, and they are related to neither the behavior policy nor the target policy. This category of action nodes accounts for the largest proportion, but with extremely low importance. Due to their small value functions and low uncertainty, there is no necessity for further exploration. Ignoring the information contained in this category of action nodes is beneficial to the efficiency of the planning algorithm.

3.3. Modification via Importance Level

This paper focuses on the inconsistency of action selection criteria between the planning procedure and the execution procedure, which is similar to the off-policy methods in reinforcement learning. Nearly all the off-policy methods in reinforcement learning adopt a technique called Importance Sampling Sutton and Barto (1998), which is often used to estimate expected values from one distribution while sampling from another. Inspired by the idea of importance sampling, this paper designs a specific online planning algorithm for POMDPs after considering the characteristics of Monte Carlo planning under partially observable settings.

To explain the principle of the algorithm, first we introduce the concept of importance sampling. Starting from the initial state s_t, under the target policy π_{act}, the occurrence probability of state-action subsequence $a_t, s_{t+1}, a_{t+1}, \cdots, s_T$ is

$$\begin{aligned} &P(a_t, s_{t+1}, a_{t+1}, \cdots, s_T \mid s_t, a_{t:T-1} \sim \pi_{act}) \\ &= \prod_{k=t}^{T-1} \pi_{act}(a_k \mid s_k) P(s_{k+1} \mid s_k, a_k). \end{aligned} \tag{10}$$

Under the behavior policy π_{plan}, the occurrence probability of state-action subsequence $a_t, s_{t+1}, a_{t+1}, \cdots, s_T$ is

$$\begin{aligned} &P(a_t, s_{t+1}, a_{t+1}, \cdots, s_T \mid s_t, a_{t:T-1} \sim \pi_{plan}) \\ &= \prod_{k=t}^{T-1} \pi_{plan}(a_k \mid s_k) P(s_{k+1} \mid s_k, a_k). \end{aligned} \tag{11}$$

The importance coefficient can be represented as

$$\begin{aligned} \rho_{t:T-1} &= \frac{\prod_{k=t}^{T-1} \pi_{act}(a_k \mid s_k) P(s_{k+1} \mid s_k, a_k)}{\prod_{k=t}^{T-1} \pi_{plan}(a_k \mid s_k) P(s_{k+1} \mid s_k, a_k)} \\ &= \prod_{k=t}^{T-1} \frac{\pi_{act}(a_k \mid s_k)}{\pi_{plan}(a_k \mid s_k)}. \end{aligned} \tag{12}$$

In the importance sampling method, if the probability of sampling the action a_k according to the target policy is greater than that of sampling the action according to the behavior policy, it indicates that the probability of this action being selected in the execution procedure is greater than the probability of being sampled in the planning procedure, so the importance of the action node is underestimated in the planning procedure and the value function should be amplified and vice versa. If the probability of sampling according to the target policy is equal to that of sampling according to the behavior policy, then the value function remains unchanged.

However, in the POMCP algorithm, both the behavior policy and the target policy are deterministic rather than stochastic, which is different from the general form of importance sampling mentioned above. The probability of sampling the action a_t at belief b and time t under the behavior policy is denoted as $\pi_{plan}(a_t \mid b_t)$, and the probability under the target policy is denoted as $\pi_{act}(a_t \mid b_t)$. As the actions sampled according to the UCT algorithm are deterministic, the probability of the selected action being sampled at belief b_t is always one hundred percent, which satisfies $\pi_{plan}(a_t \mid b_t) \equiv 1$. If the action is also of maximum value at belief b_t, which means $\pi_{act}(a_t \mid b_t) = 1$, then the corresponding term of this belief node in the importance coefficient is 1. If the action is not the one who has the maximum value at belief b_t, but an explorative action obtained by considering the action value function and the number of visits simultaneously, then $\pi_{act}(a_t \mid b_t) = 0$ and the corresponding term of the belief node in the importance coefficient is 0. Therefore, it can be seen that in the deterministic policy settings, the backtracing termination is easy to occur when the importance sampling method is directly applied.

In order to extend the idea of importance sampling to the POMCP algorithm, it is necessary to design an approach that takes both the efficiency and the effect into account. In a planning trajectory, all the actions are sampled via the UCT algorithm. Among these actions, some also have the maximum action values, so they have the highest importance which is assigned as 1. Other actions in the trajectory are not the optimal and their importance levels are assigned as *imp*. In order to match the action value functions in scale, we assign $(1 - imp)$ to the actions not sampled but with the maximum values as a supplement. Then the value of the parent node will be updated as the weighted sum of the maximum action value and the value of the action node selected by the UCT algorithm, where the weight is based on the importance level. Regardless of whether the actions sampled by two criteria coincide or not, the above analysis is always valid.

4. POMCP-PM Algorithm

4.1. Details of POMCP-PM Algorithm

This paper proposes a new Partially Observable Monte Carlo Planning algorithm based on Path Modification (POMCP-PM), which selects important nodes across multiple paths to update the values of belief nodes during the updating stage rather than simply selecting nodes in the exploring path. The detailed PM_SIMULATE procedure is shown in Algorithm 3, where $C(h)$ means the number of child nodes under the history sequence h, *imp* represents the importance of the sampled action and $imp \in [0, 1]$.

Algorithm 3 PM_SIMULATE

procedure PM_SIMULATE(s, h, d)
  if $d = 0$ **then**
    return 0
  $a_{UCB} \leftarrow \arg\max_{a \in C(h)} \left\{ Q(h_a) + c\sqrt{\frac{\log[N(h)]}{N(h_a)}} \right\}$
  $a_{maxQ} \leftarrow a_{UCB}$
  if $C(h) = |A|$ **then**
    $a_{maxQ} \leftarrow \arg\max_{a \in C(h)} Q(h_a)$
  $s', o, r \leftarrow G(s, a_{UCB})$
  $M(h_{a_{UCB}o}) \leftarrow M(h_{a_{UCB}o}) + 1$
  if $M(h_{a_{UCB}o}) = 1$ **then**
    return $r + \gamma$ROLLOUT$(s', h_{a_{UCB}o}, d - 1)$
  $total \leftarrow r + \gamma$PM_SIMULATE$(s', h_{a_{UCB}o}, d - 1)$
  $N(h) \leftarrow N(h) + 1$
  $N(h_{a_{UCB}}) \leftarrow N(h_{a_{UCB}}) + 1$
  $Q(h_{a_{UCB}}) \leftarrow Q(h_{a_{UCB}}) + \frac{total - Q(h_{a_{UCB}})}{N(h_{a_{UCB}})}$
  return $imp \cdot total + (1 - imp) \cdot Q(h_{a_{maxQ}})$

4.2. Algorithm Analysis

In the updating stage of the planning procedure in POMCP-PM, the return value is set to be $imp \cdot total + (1 - imp) \cdot Q(h_{a_{maxQ}})$, which is the weighted sum of the value functions

of the sampled action and the optimal action. When *imp* is set to one, the algorithm is equivalent to POMCP.

Note that if the action selected by the UCT algorithm is the same as the action selected by the maximum function, *total* and $Q(h_{a_{maxQ}})$ will share the same expected value, $\mathbb{E}[total] = \mathbb{E}[Q(h_{a_{maxQ}})] = e$, then $\mathbb{E}\left[imp \cdot total + (1 - imp) \cdot Q(h_{a_{maxQ}})\right] = e$. It shows that the expected return value is the same as that in the original algorithm. If the action selected by the UCT algorithm is different from the action selected by the maximum function, it means that the sampled action is explorative, which benefits the growth and update of the subtrees but hinders the parent node from calculating and selecting the optimal value function, thus has medium importance. In POMCP-PM, the return value can be divided into two parts. The first one is *total* times the discount factor *imp*, which reduces the contribution of the sampled actions to the parent node. The second one is the optimal action value $Q(h_{a_{maxQ}})$ times the discount factor $(1 - imp)$, added the contribution of the optimal action to the parent node.

As with POMCP, POMCP-PM also samples actions according to the UCT algorithm in the planning procedure, which keeps a good balance between exploration and exploitation. In fact, the UCT algorithm can be replaced with any improved action selection criterion. Moreover, POMCP-PM reduces the impact of explorative actions during the updating stage, so that the convergence of the planning procedure can be accelerated.

5. Experiments and Analysis

We compared the simulation results of POMCP-PM and POMCP on three scenarios with different scales: Tiger, RockSample[7,8] and RockSample[11,11]. The scales of three benchmark problems are shown in Table 1, where $|S|$, $|A|$ and $|O|$ denote the number of states, actions and observations.

Table 1: Scales of Benchmark Problems

	|S|	**|A|**	**|O|**
Tiger	2	3	2
RockSample[7,8]	12545	13	3
RockSample[11,11]	247808	16	3

Tiger is a small-scale POMDP problem with only two states, three actions, and two observations. In this problem, there is one door on the left and another on the right. A tiger hides behind one of the doors, marked as state S_L and state S_R respectively, and the two states share the same initial probability. At each step, the optional actions for the agent include opening the left door, opening the right door and listening. If the agent makes a decision to open one of the doors, then it will score 10 as a reward if there is no tiger behind the opened door or score -100 as a punishment otherwise. If the agent chooses to listen, it will score -1 as a punishment and receive the correct information with the probability of

0.85. Once one of the doors is opened, the problem will be reset. There is no terminal state for the problem and the episode will terminate once the maximum number of time steps setted is reached.

RockSample is a POMDP problem about ore mining whose scale can be specified. RockSample$[n, k]$ represents a map with k rocks whose size is $n \times n$ and Figure 1 is an illustration of RockSample$[5, 5]$. At each step, the optional actions for the agent include moving towards four directions, sampling the rock and detecting one of the k rocks, so there are altogether $(k + 5)$ optional actions. Only when the agent and the rock are in the same location can the agent sample the rock. It is of high cost to sample a rock but not all rocks are valuable. If the agent samples the valuable rocks, it will be rewarded, otherwise it will be punished. Other than sampling, the agent can also choose to detect the value of a specified rock, and the detection error increases with the distance between the agent and the rock. When the agent reaches the exit, the episode terminates and the agent receives a reward.

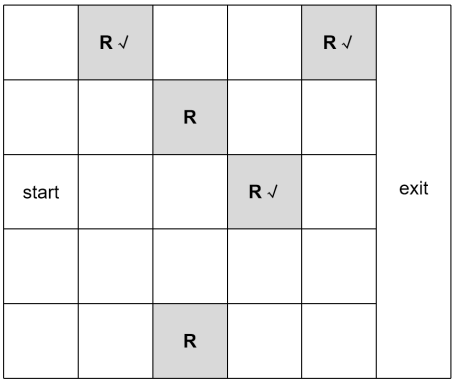


Figure 1: An illustration of RockSample$[5, 5]$.

We implemented POMCP-PM based on POMDPs.jl Egorov et al. (2017) in Julia and all experiments were carried out with a 64-bit Linux operating system, Intel(R) Core(TM) i7-10700 CPU@2.90GHz processor, and Julia 1.7. In each benchmark problem, the hyper-parameters (*e.g.*, the maximum steps *max_steps*, the discount factor γ and c in the UCT algorithm) shared the same values in two algorithms. The numbers of simulations before each execution was set to different values that increase exponentially. Each episode terminated when the agent met the terminal states or the maximum depth was reached. Table 2, 3 and 4 shows the Average Discounted Rewards (ADRs) in POMCP-PM and POMCP. To be more intuitive, the results of the experiments are also presented in Figure 2, which record the changes of ADRs with the number of simulations under the scenarios of Tiger, RockSample$[7, 8]$ and RockSample$[11, 11]$.

With the growth of the number of simulations, the ADRs corresponding to POMCP-PM and POMCP both increase. It is obvious that POMCP-PM always outperforms POMCP with varying numbers of simulations in scenarios with different scales. In Tiger, POMCP-PM has certain advantage when the number of simulations is small. When the number of simulations increases to 256, the performance is equivalent to POMCP. As the number of simulations further increases, both algorithms experience little growth in their performance.

Table 2: ADRs for POMCP-PM and POMCP On Tiger

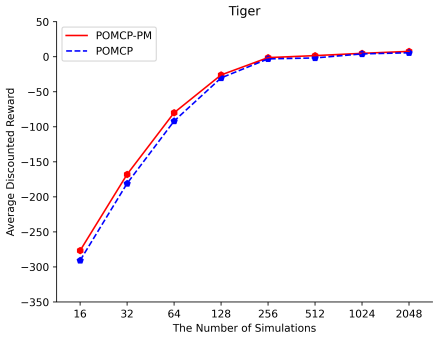
Simulation Times	**POMCP-PM**	**POMCP**
16	−276.680	−290.750
32	−167.883	−180.865
64	−79.944	−91.832
128	−26.034	−30.238
256	−1.264	−3.098
512	1.567	−1.772
1024	4.839	3.994
2048	7.616	5.626

Table 3: ADRs for POMCP-PM and POMCP On RockSample[7, 8]

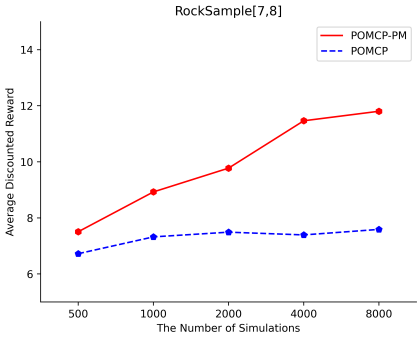
Simulation Times	**POMCP-PM**	**POMCP**
500	7.505	6.722
1000	8.926	7.322
2000	9.775	7.490
4000	11.464	7.392
8000	11.804	7.589

Table 4: ADRs for POMCP-PM and POMCP On RockSample[11, 11]

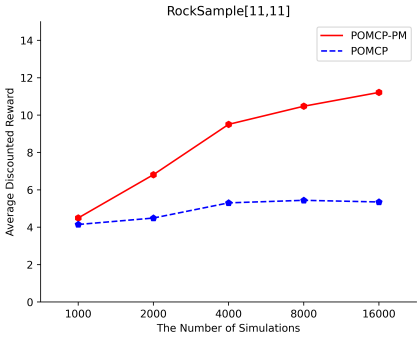
Simulation Times	**POMCP-PM**	**POMCP**
1000	4.495	4.146
2000	6.811	4.491
4000	9.504	5.303
8000	10.475	5.440
16000	11.214	5.352



(a) Results of Tiger.



(b) Results of RockSample[7, 8].



(c) Results of RockSample[11, 11].

Figure 2: ADRs Changing with Simulation Times for POMCP-PM and POMCP on Tiger, RockSample[7, 8] and RockSample[11, 11].

The reason is that Tiger is a small-scale problem and POMCP can also achieve ideal results given enough simulations. In larger problems RockSample[7, 8] and RockSample[11, 11], as the number of simulations increases, POMCP-PM shows more and more advantage over POMCP. Furthermore, POMCP-PM still maintains continuous growth momentum, while the performance of POMCP keeps nearly the same as before.

6. Conclusion and Future Work

In this paper, we have introduced a new algorithm called POMCP-PM, which proposes an importance based way to update the values of parent nodes in the updating stage. The importance of action nodes is evaluated by analyzing the deviation between action selection criteria in the planning procedure and in the execution procedure and dividing the nodes into four categories. Different from most existing algorithms, the proposed algorithm improves the online planning procedure by focusing on the updating stage instead of the growing stage, providing a new perspective on balancing exploration and exploitation. Moreover, it is orthogonal to the existing algorithms and can be combined with different action selection

criteria to be more general. As shown in the experimental results, POMCP-PM achieves higher ADRs than POMCP with varying numbers of simulations in scenarios with different scales, which indicates that the proposed algorithm accelerates the convergence and keeps a better balance between exploration and exploitation.

In POMCP-PM, the importance of the second and the third categories of action nodes relies on tuning the hyperparameter *imp*. Future work will focus on designing an adaptive method to compare the importance of the two categories and set a proper importance value.

Acknowledgments

This paper is supported by the National Natural Science Foundation of China (Grant No. 62192783, 62376117), the Collaborative Innovation Center of Novel Software Technology and Industrialization at Nanjing University.

References

Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A Survey of Monte Carlo Tree Search Methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

Maxim Egorov, Zachary N Sunberg, Edward Balaban, Tim A Wheeler, Jayesh K Gupta, and Mykel J Kochenderfer. POMDPs.jl: A Framework for Sequential Decision Making under Uncertainty. *The Journal of Machine Learning Research*, 18(1):831–835, 2017.

David P. Helmbold and Aleatha Parker-Wood. All-Moves-As-First Heuristics in Monte-Carlo Go. In *Proceedings of the 2009 International Conference on Artificial Intelligence*, pages 605–610, 2009.

Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and Acting in Partially Observable Stochastic Domains. *Artificial Intelligence*, 101(1-2):99–134, 1998.

Levente Kocsis and Csaba Szepesvári. Bandit Based Monte-Carlo Planning. In *European Conference on Machine Learning (ECML)*, pages 282–293. Springer, 2006.

Michael H. Lim, Claire J. Tomlin, and Zachary N. Sunberg. Voronoi Progressive Widening: Efficient Online Solvers for Continuous State, Action, and Observation POMDPs. In *60th IEEE Conference on Decision and Control (CDC)*, pages 4493–4500, 2021.

P. Liu, C. Jing, and H. Liu. An improved Monte Carlo POMDPs online planning algorithm combined with RAVE heuristic. In *IEEE International Conference on Software Engineering & Service Science*, 2015.

Omid Madani, Steve Hanks, and Anne Condon. On the Undecidability of Probabilistic Planning and Infinite-Horizon Partially Observable Markov Decision Problems. In *Proceedings of the 16th AAAI Conference on Artificial Intelligence*, pages 541–548, 1999.

John Mern, Anil Yildiz, Lawrence Bush, Tapan Mukerji, and Mykel J. Kochenderfer. Improved POMDP Tree Search Planning with Prioritized Action Branching. In *Thirty-Fifth AAAI Conference on Artificial Intelligence*, pages 11888–11894. AAAI Press, 2021a.

John Mern, Anil Yildiz, Zachary Sunberg, Tapan Mukerji, and Mykel J. Kochenderfer. Bayesian Optimized Monte Carlo Planning. In *AAAI Conference on Artificial Intelligence*. AAAI Press, 2021b.

Christos H Papadimitriou and John N Tsitsiklis. The Complexity of Markov Decision Processes. *Mathematics of Operations Research*, 12(3):441–450, 1987.

Joelle Pineau, Geoff Gordon, Sebastian Thrun, et al. Point-Based Value Iteration: An Anytime Algorithm for POMDPs. In *International Joint Conference on Artificial Intelligence (IJCAI)*, volume 3, pages 1025–1032, 2003.

Stéphane Ross, Joelle Pineau, Sébastien Paquet, and Brahim Chaib-Draa. Online planning algorithms for POMDPs. *Journal of Artificial Intelligence Research (JAIR)*, 32:663–704, 2008.

Maarten P. D. Schadd, Mark H. M. Winands, H. Jaap van den Herik, Guillaume Chaslot, and Jos W. H. M. Uiterwijk. Single-Player Monte-Carlo Tree Search. In *Computers and Games, 6th International Conference (CG), Beijing, China*, volume 5131 of *Lecture Notes in Computer Science*, pages 1–12, 2008.

Maarten Peter Dirk Schadd. Selective search in games of different complexity. 2011.

David Silver and Joel Veness. Monte-Carlo Planning in Large POMDPs. *Advances in Neural Information Processing Systems (NIPS)*, 23:2164–2172, 2010.

Richard D. Smallwood and Edward J. Sondik. The Optimal Control of Partially Observable Markov Processes over a Finite Horizon. *Operations Research*, 21(5):1071–1088, 1973.

Trey Smith and Reid Simmons. Heuristic Search Value Iteration for POMDPs. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 520–527, 2004.

Adhiraj Somani, Nan Ye, David Hsu, and Wee Sun Lee. DESPOT: Online POMDP Planning with Regularization. *Advances in Neural Information Processing Systems (NIPS)*, 26:1772–1780, 2013.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998. ISBN 978-0-262-19398-6.