Applied Intelligence (2023) 53:8146–8159
https://doi.org/10.1007/s10489-022-03914-5

High-efficiency online planning using composite bounds search under partial observation

Yanjie Chen[1,4] · Jiangjiang Liu[1] · Yibin Huang[1] · Hui Zhang[2,4] · Yaonao Wang[3,4]

Accepted: 19 June 2022 / Published online: 30 July 2022

Abstract

Motion planning in uncertain environments is a common challenge and essential for autonomous robot operations. Representatively, the determinized sparse partially observable tree (DESPOT) algorithm shows reasonable performance for planning under uncertainty. However, DESPOT may generate a low-quality solution due to inaccurate searches and low efficiencies in the belief tree construction. Therefore, this paper proposes a high-efficiency online planning method built upon the DESPOT algorithm, namely, the DESPOT with discounted upper and lower bounds (DESPOT-DULB) algorithm, to simultaneously improve the efficiency and performance of motion planning. Particularly, the node's information is represented by combining the upper and lower bounds of the node (ULB) in the forward exploration of the action space to reasonably assist the optimal action selection. Then, a discounted factor based on the depth information of the belief tree is introduced to reduce the gap between the upper bound and lower bound both in the action space and observation space. As a result, the proposed method can comprehensively represent the information of the node to ensure a near-optimal forward search. The theoretical proofs of the proposed method are provided as well. The simulation results, including three representative scenario comparisons and a parameter sensitivity analysis, demonstrate that the proposed method exhibits favorable performances in many examples of interest.

Keywords Planning under uncertainty · POMDP · High efficiency · Determinized sparse partially observable tree

1 Introduction

Planning under uncertainty is a common challenge in many robotic applications, such as visual tracking [1] and autonomous navigation [2, 3]. Generally, the robot is difficult to operate accurately due to the uncertainty originating from sensor noise, imperfect robot control, and changing environments.

Recently, the partially observable Markov decision process (POMDP) [4] has provided a principled mathematical framework for robot decision-making and planning under uncertainty. Therefore, the POMDP has undertaken a wide range of challenging tasks in robotics, e.g., autonomous driving [5], grasping [6], robot rescue [7], and intelligent tutoring systems [8]. However, POMDP planning is computationally intractable due to various sources. The state space and belief size grow exponentially with the number of states, also known as the "curse of dimensionality". Moreover, the number of action-observation histories grows exponentially with the planning range, which is the "curse of history" [9]. Both the curse of dimensionality and history cause an exponential

✉ Hui Zhang
zhanghui1983@hnu.edu.cn

Yanjie Chen
chenyanjiehnu@gmail.com

Jiangjiang Liu
200227107@fzu.edu.cn

Yibin Huang
N190227062@fzu.edu.cn

Yaonao Wang
yaonan@hnu.edu.cn

1 School of Mechanical Engineering and Automation, Fuzhou University, Fuzhou 350108, China

2 School of Robotics, Hunan University, Changsha 410082, China

3 College of Electrical and Information Engineering, Hunan University, Changsha 410082, China

4 National Engineering Research Center of Robot Visual Perception and Control Technology, Changsha 410082, China

growth in computational complexity, especially for large-scale POMDP planning.

In addition to computational difficulties, the POMDP planning has certain challenges due to the existing uncertainty. Currently, various attempts have been made to effectively deal with the uncertainty. Examples include Bayesian networks [10], convolutional neural networks [11], and α-vector policy graphs [12]. However, both Bayesian networks and convolutional neural networks need to train the network parameters and the trained parameters are not guaranteed to be optimal. Meanwhile, when the number of α-vectors is large, the policy graph explodes exponentially and cannot be explained. Fortunately, Kocsis et al. [13] introduced a search framework (Monte Carlo tree search) to reduce the uncertainty through multiple simulations and calculations. Then, Silver et al. [14] introduced partially observable Monte Carlo planning (POMCP) to solve real-time uncertainties in large tasks by combining MCTS and a partially observed upper confidence tree (PO-UCT) algorithm [15]. However, the approach still faces some challenges, such as being misguided by the upper confidence bound (UCB) heuristic of the UCT algorithm and an overly greedy convergence.

To further improve the performance of planning under uncertainty, Somani et al. [16] constructed a sparse belief tree through executing a series of searches. In the belief tree, the new child node was expanded based on the node's upper bound information, which was calculated by the particles of the current node performing multiple simulations. However, there are two difficulties in the belief tree search stage. The first difficulty is that the current optimal action branch may be wrongly selected with a high probability because the information of the node may be not considered sufficiently. A reasonable representation of the child node information is helpful for the exploration of belief trees. The information of the nodes on the belief tree not only depends on the upper bound but also needs to consider the lower bound. To construct the optimal belief tree, the combination of upper and lower bounds is considered an exploration item for representing the node's information comprehensively and improving the efficiency of a search. The second difficulty is that the uncertainty of the node has slight fluctuations because the initial upper and lower bound of the node are prone to change. To relieve the fluctuations, many researchers are considered that adjusting the exploration item is a desirable approach. Bougie et al. [17] encouraged high-level explorations by introducing hyperparameters to adjust fast and slow rewards in the exploration item. Chen et al. [18] improved value function approximation by decreasing the exploration discount factor. However, the discount factor is a constant that does not take into consideration the effect of the node's depth in the above methods. Therefore, a depth function is introduced as a dynamic discount factor to adjust the exploration item.

In this paper, the combination of upper and lower bounds of the node is considered to solve the problem of incomplete representation of the node's information. In addition, the depth function as a discount is introduced to relieve the fluctuation of the uncertainty. Therefore, discounted upper and lower bounds of the node are introduced for the DESPOT algorithm to construct the belief tree, as shown in Fig. 1. The proposed online planning method, named DESPOT-DULB, improves the search strategy of the traditional DESPOT by introducing ULB in a forward exploration of action selection and introducing the discount factor in forwarding explorations of action and observation selections, respectively. As a benefit, the uncertainty is reduced, and favorable performances can be attained, such as performances that show high rewards and high efficiency. The main contributions of this paper are listed as follows:

1. In the belief tree expansion stage, the information of the belief node is represented by combining the node's upper and lower bounds. Then, the belief tree is expanded based on the node's information. Compared with merely using the upper bound or the lower bound of the node, the proposed method is useful for efficiently searching for the optimal action.
2. Because of the slight fluctuations in the uncertainty of the belief node, a depth function is proposed as the discount factor to adjust the gap between the upper bound and lower bound of the node to ensure a reasonable reduction in uncertainty. As the uncertainty decreases, the performance of the proposed method can be further improved.
3. The proposed search criteria have certain theoretical guarantees and the convergence of the reconstructed belief tree is demonstrated. Meanwhile, the experimental results show that DESPOT-DULB has performance improvements on tasks of interest.

This paragraph has been reorganized as follows. The rest of this paper is organized as follows. Section 2 shows the related work. Section 3 reviews the background on POMDP model. Section 4 describes the online POMDP algorithm based on the discounted upper and lower bounds. The related theoretical analysis of DESPOT-DULB are shown in Section 4, with Appendix A and B providing detailed theoretical proofs. Section 5 presents the experimental results regarding the performance of DESPOT-DULB, compared with standard POMDP benchmarks. The paper is concluded in Section 6.

2 Related work

Generally, two kinds of approximate POMDP planning methods are adopted in the current research, namely, offline planning [19–21] and online planning [14, 16, 22]. These

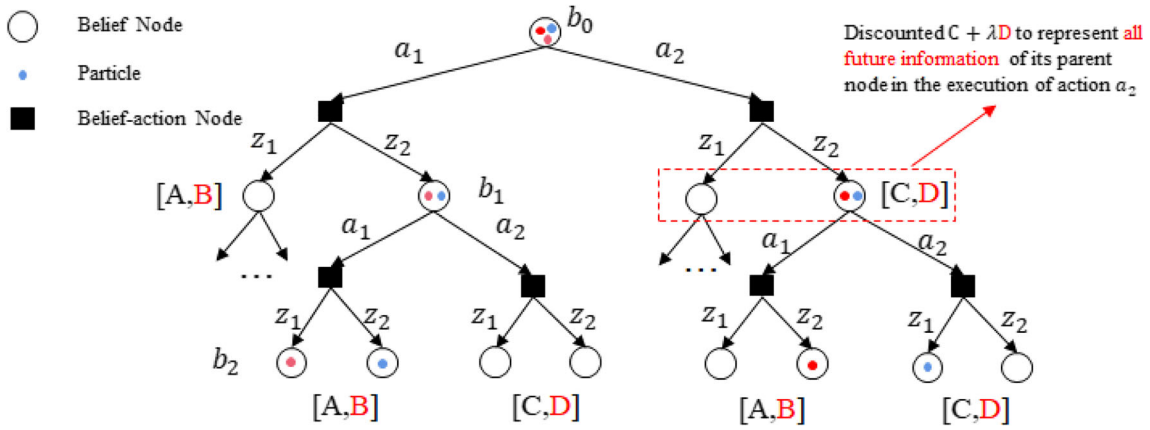

Fig. 1 Online POMDP planning performs forward search on a tree. [A, B] ([C, D]) represents the sum of the upper bound and lower bound of all sibling child nodes after executing an action, respectively. λ represents a discount factor. The DESPOT-DULB introduces the combination of the upper bound (**A** or **C**) and lower bound (**B** or **D**) to represent the node's information comprehensively for constructing the belief tree

methods have been applied to many fields but there are still some challenges. For example, the efficiency is low, and the performance is poor for large-scale spaces. Although offline planning has made great progress [20, 21], the approach is difficult to expand to large POMDPs because the number of states increases. Online planning can expand to large POMDPs, but online planning is slower than offline planning because a search needs to be executed at each step. Online and offline planning have been combined to further improve the planning performance efficiently [23].

In previous work [24], the upper bound as a heuristic was usually more popular for explorations compared with the lower bound. Many algorithms, such as HSVI and SARSOP, can deal with POMDP tasks effectively by relying on the upper bound as a heuristic. Meanwhile, the state-of-the-art online algorithms DESPOT [16] and POMCP [14] have been widely used in the field of robotics, e.g., vision planning [25] and contact manipulation [26]. Both POMCP and DEPSOT adopted the idea of the upper confidence bound (UCB) to search for the optimal action for constructing the search tree. POMCP used a particle to perform multiple simulations for estimating the information of the leaf node to effectively search the optimal action. POMCP++ [27] further improved the performance of the algorithm by using a set of particles rather than a particle to perform simulations to obtain accurate information. Hierarchical POMCP [28] solved large POMDP problems by premeditating hierarchical models. However, the search of the aforementioned approaches is easily misguided and overly greedy. Nonetheless, DESPOT [29] demonstrated strong performances on large POMDP problems by applying the initialize upper bound of the leaf node as a heuristic to search for optimal actions. However, many difficulties still exist that degrade the performance of the DESPOT algorithm. First, the incomplete node information affects the quality of the constructed belief tree. Then, the fluctuation of node uncertainty affects the convergence efficiency of the constructed belief tree.

To improve the performance of the DESPOT algorithm, many researchers have tried in various aspects. DESPOT-α [30] changed the search method of the belief tree based on the α vector to deal with particle divergence. IS-DESPOT [31] introduced importance sampling to improve the performance of DESPOT under certain conditions. However, this sampling only worked when dealing with important events with low probability. Hyp-DESPOT [32] further improved the planning time by integrating the CPU and GPU. Hyp-DESPOT generated a parallel DESPOT tree by using a multi-CPU to traverse multiple independent paths and a GPU to execute parallel Monte Carlo simulations at the leaf nodes of the search tree. However, all of the above approaches generally construct a belief tree without considering node information sufficiently, and then may result in the fluctuation of the uncertainty. Meanwhile, considering the search based on the lower bound is conducive to obtaining the optimal policy; an impressive idea is to switch the upper and lower bounds as a heuristic. LB-DESPOT [33] designed the action heuristic by probabilistically selecting the upper bound or the lower bound of the node. Nevertheless, the switching conditions of the heuristic are difficult to set and the convergence efficiency of the belief tree cannot improve.

In short, this paper proposes the DESPOT-DULB to improve search performances and convergence efficiency. DESPOT-DULB inherits the following parts of DESPOT [29]:

1. the empirical value $V_\pi(b)$ of a policy π is the average total discounted reward obtained by simulating the policy under each scenario,
2. the regularized objective function aims to overcome overfitting,
3. the regularized weighted discounted utility (RWDU) function $\nu(b)$ aims to choose the optimal policy.

Therefore, DESPOT-DULB considers the same method as DESPOT to initialize the upper bound and lower bound of the

node. The main difference is that DESPOT-DULB introduces the discounted upper and lower bounds as a heuristic to search for the optimal action rather than searching based on the upper bound of the node in the forward search stage. Meanwhile, a depth function is considered to reduce the uncertainty by adjusting the gap between the upper bound and lower bound of the node. The theoretical analysis and simulation results verify the favorable performance of the proposed method.

3 Background

Uncertainty originates from noisy sensors, changing environments and imperfect control. These situations cause significant challenges for motion planning. To effectively plan under these uncertainties, the POMDP model is generally introduced to reduce the uncertainty by updating the beliefs according to the received information. Formally, the POMDP model can be designated as a tuple (S, A, Z, T, O, R), where S is a set of states, A is a set of agent actions and Z is a set of observations. $R(s, a)$ is the immediate reward obtained on executing action a in state s. When action a is executed in state s, the probability of the next state s' is defined as state transition function $T(s, a, s') = p(s'|\, s, a)$. In addition, the probability of observing z in state s' reached by performing action a is defined as observation function $O(s', a, z) = p(z|\, s', a)$.

A POMDP agent generally cannot obtain the true state because of the randomness and unpredictability of the environment. However, the agent receives observations continuously that provide partial information about the state. Thus, the agent maintains a belief, which is a probability distribution over S. The agent starts with an initial belief b_0. At the time t, the agent updates the belief according to Bayes rule, by incorporating information from the action a_t taken and the resulting observation o_t:

$$b_t\left(s'\right) = \eta O\left(s', a_t, z_t\right) \sum_{s \in S} T\left(s, a_t, s'\right) b_{t-1}(s) \tag{1}$$

where η is a normalizing constant. The belief $b_t = \tau(b_{t-1}, a_t, z_t) = \tau(\tau(b_{t-2}, a_{t-1}, z_{t-1}), a_t, z_t) = \cdots = \tau(\cdots\tau(\tau(b_0, a_1, z_1), a_2, z_2), \cdots, a_t, z_t)$ is a sufficient statistic that contains all the information from the history of actions and observations $(a_1, z_1, a_2, z_2, \cdots, a_t, z_t)$.

A policy $\pi : B \rightarrow A$ is a mapping from belief space B to action space A. The policy prescribes an action $\pi(b) \in A$ at the belief $b \in B$. The ultimate goal of POMDP planning is to choose a policy π that maximizes the value function $V_\pi(b)$, that is, the expected total discount reward.

$$V_\pi(b) = E\left(\sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(b_t)) | b_0 = b\right) \tag{2}$$

where b_0 is the initial belief. The constant $\gamma \in [0, 1)$ is a discount factor, expressing preferences for immediate rewards over future ones.

In online POMDP planning, the agent starts with an initial belief. Then, the following process is repeated. At each time iteration, (1) the agent searches for an optimal action a^* at the current belief b; (2) the agent executes the action a^* and receives a new observation z; (3) the agent updates the belief using Eq. (1) continuously. To search for an optimal action a^*, a valid way is to construct a belief tree (Fig. 1), treating the current belief b_0 as the initial belief at the root node of the tree. The agent performs a forward search on the tree for a policy π that maximizes the value $V_\pi(b_0)$ at node b_0 and sets $a^* = \pi(b_0)$. Each node of the tree represents a belief. Each node branches into $|A|$ action edges, and each action branches into $|Z|$ observation edges. If the node and its child node are represented by beliefs b and b', respectively, then $b' = \tau(b, a, z)$ for $a \in A$ and $z \in Z$.

To find a near-optimal policy, the tree is truncated at a maximum depth D, and then the agent searches for the optimal policy on the truncated tree. For each leaf node, an estimated lower bound on its optimal value is obtained by simulating a default policy. A default policy can usually be a random policy or a heuristic. At the internal node b, the Bellman's principle of optimality is applied for computing the maximum value:

$$V^*(b) = \max_{a \in A} \Big\{ \sum_{s \in S} b(s) R(s, a) + \gamma \sum_{z \in Z} p(z|b, a) V^*(\tau(b, a, z)) \Big\} \tag{3}$$

which computes the sum of all the action branches and the average value of all the observation branches. The proposed algorithm traverses the belief tree from the leaf node to the root node and recursively calculates the maximum value of each node using Eq. (3). Then, the agent executes the best policy at the root node b_0.

4 Despot-DULB

In this section, the combination of the initial upper bound and lower bound of the node has been considered to represent the information of the node effectively in an action selection. To further ensure a reasonable reduction in uncertainty, a depth function is proposed for the forward search in the action and observation selection stages respectively. Then, the upper and lower bounds of nodes on the path are adjusted slightly during the backup phase. The specific description of the proposed method is as follows.

Algorithm 1 Algorithm DESPOT-DULB

Input:
ϑ: Initial belief
ε_0: Target gap between $\mu(b_0)$ and $l(b_0)$
ξ: The rate of target gap reduction
K: The number of sampled scenarios
D: The maximum depth of DESPOT-DULB
λ: Regularization constant
$T_{\max}$: The maximum planning time per step
κ: The depth discount factor

1: Initialize $b \leftarrow \vartheta$.
2: **repeat**
3: $l \leftarrow$ BUILD DESPOT-DULB(b).
4: $a^* \leftarrow \max_{a \in A} l(b,a)$.
5: **if** $L_0(b) > l(b,a^*)$ **then** $a^* = \pi_0(b)$.
6: **end if**
7: Execute a^*.
8: Receive the observation z.
9: $b \leftarrow \tau(b,a^*,z)$.
10: **until** termination
11:
12: **function** BUILD DESPOT-DULB(b)
13: Sample randomly a set Φ_{b_0} of K scenarios from the current belief b_0.
14: Build a new DESPOT-DULB D with a node b as the root.
15: Initialize the bounds $U(b_0)$, $L_0(b_0)$, $\mu(b_0)$, and $l(b_0)$ of node.
16: Compute $\varepsilon(b_0) \leftarrow \mu(b_0) - l(b_0)$.
17: **while** $\varepsilon(b_0) > \varepsilon_0$ and $T \le T_{\max}$ **do**
18: $b \leftarrow$ EXPLORE(D, b).
19: BACKUP(D, b).
20: Recompute $\varepsilon(b_0) \leftarrow \mu(b_0) - l(b_0)$.
21: **end while**
22: **return** l
23: **end function**
24:
25: **function** EXPLORE(D, b)
26: **while** $\Delta(b) \le D, E(b) > 0$ **do**
27: **if** b is a leaf node in D **then**,
28: Expand b one level deeper. Insert each new child b' of b into D. Initialize $U(b')$, $L_0(b')$, $\mu(b')$ and $l(b')$.
29: $a^* = \arg\max_{a \in A} d(b,a)$.
30: Depth discount: $\beta = \kappa^{\Delta(b)}$.
31: $z^* = \arg\max_{z \in Z_{b,a}} E(b')$
32: Update belief $b \leftarrow \tau(b,a^*,z^*)$
33: **end if**
34: **end while**
35: **if** $\Delta(b) > D$ **then**
36: $U(b) \leftarrow L_0(b)$
37: $\mu(b) \leftarrow l_0(b)$
38: $l(b) \leftarrow l_0(b)$
39: **end if**
40: **return** b
41: **end function**

4.1 Online planning

Algorithm 1 shows the overall framework and pseudocode of the DESPOT-DULB algorithm. In particular, the BUILD DESPOT-DULB function provides a high-level sketch of the belief tree construction. The specific process is outlined here. 1) The root node b_0 is randomly initialized by sample K scenarios (line 13). 2) The upper and lower bounds of the root node b_0 are initialized (line 14–15). 3) The algorithm conducts a series of explorations to expand the belief tree D and reduce the gap between the upper bound $\mu(b_0)$ and the lower bound $l(b_0)$ at the root node b_0 of D. 4) Each exploration follows the optimal action a^* using Eq. (4)and observation z^* using Eq. (6) are chosen to expand the belief tree (line 18). 5) The algorithm traces the path back to the root and performs a backup on the upper and lower bounds at each node along the way using the Bellman's principle (line 19). 6) The explorations continue until the gap between the bounds $\mu(b_0)$ and $l(b_0)$ at the root node reaches the target value ε_0, i.e., judging $\mu(b_0) - l(b_0) < \varepsilon_0$ is satisfied or not (line 17).

A reasonable belief tree D is constructed by considering the combination of the upper bound and lower bound of the node in Step (4) to improve the search of action selections. Meanwhile, the depth function is introduced to adjust the gap between the upper bound and lower bound of the node in Step (4) to ameliorate the action search and observation

selection. Then, the upper and lower bounds of the nodes on the path are slightly adjusted during Step (5). The forward search and backup are repeated until the terminal conditions are met, such as the gap at the root node reaching a target value and the planning time running out.

1) *Forward exploration*: To construct the belief tree, searching for the optimal action within a finite time must be considered. In current research, two kinds of approaches are generally adopted in action selection. One is dynamic programming that needs to construct a complete belief tree before looking for the optimal action. Another is a forward search algorithm that avoids constructing the complete belief tree in advance. For large-scale POMDPs, the complete belief tree is constructed impractically. To scale up to large POMDPs, the belief tree is constructed incrementally under the guidance of a heuristic. During the heuristic search, an upper bound $\mu(b)$ and a lower bound $l(b)$ on the optimal RWDU are maintained at each node b of D, that is, $l(b) \le \nu^*(b) \le \mu(b)$. An upper bound $U(b)$ and a lower bound $L_0(b)$ on the empirical value are computed so that $U(b) \le \widehat{V}^*(b) \le L_0(b)$. In particular, let $L_0(b) = \widehat{V}_{\pi_0}(b)$ when performing a default policy π_0 at node b. Let $\varepsilon(b) = \mu(b) - l(b)$ denote the gap between the upper and lower RWDU bounds at a node b. The goal of each forward search is to reduce the gap of root node $\varepsilon(b_0)$. The ultimate goal of a forward exploration is to find an action sequence that can make the gap of root node convergence zero.

To search the optimal action branch, the upper and lower bounds of the node are fully adopted to represent the information of the node. Although the combination of upper and lower bounds of the node can converge the gap to a small value, the gap has slight fluctuations. A depth function β is introduced as a discount to further improve the performance of searching. Starting from the root node b_0, along each node b of the search path, the optimal action branch is selected according to the discounted upper and lower bounds information $(\mu(b) + \omega \cdot l(b))/\beta$:

$$a^* = \underset{a \in A}{\operatorname{argmax}}\, d(b,a) = \underset{a \in A}{\operatorname{argmax}} \left\{ \rho(b,a) + \left(\sum_{z \in Z_{b,a}} \mu\left(b'\right) + \omega \sum_{z \in Z_{b,a}} l\left(b'\right) \right) / \beta \right\} \tag{4}$$

where $b' = \tau(b, a, z)$ is the child node of b by performing action branch a and observation branch z at b. $0 \le \omega \le 1$ is the proportion factor.

Considering that the gap has a slight fluctuation, a depth function β is introduced in the forward search. β is the discount factor that is determined by the depth of the node and is defined as follows:

$$\beta = \kappa^{\Delta(b)} \tag{5}$$

where $\kappa > 1$ is a constant. $\Delta(b)$ is the depth of node b.

After executing action a^*, observation branch z is chosen by maximizing the excess uncertainty $E(b')$ at node b'. Due to the change in the updated upper and lower bounds, the gap of the node may exhibit slight fluctuations. The depth function β is introduced to $E(b')$ to ensure a reasonable reduction in uncertainty.

$$\begin{aligned} z^* &= \underset{z \in Z_{b,a}}{\operatorname{argmax}} E\left(b'\right) \\ &= \underset{z \in Z_{b,a}}{\operatorname{argmax}} \left\{ \beta \varepsilon\left(b'\right) - \sum_{\phi \in \Phi_{b'}} \frac{|\Phi_{b'}|}{K} \xi \varepsilon(b_0) \right\} \end{aligned} \tag{6}$$

where b_0 is the root node. Intuitively, the excess uncertainty $E(b')$ measures the gap between the multiple of the current gap at b' and the "expected" gap at b'.

The leaf node b is expanded by creating a child node b' of b for each action branch $a \in A$ and each observation encountered under a scenario $\phi \in \Phi_b$. For each new child node b', the bounds $\mu_0(b')$, $l_0(b')$, $U_0(b')$ and $L_0(b')$ are initialized. The RWDU bounds $\mu_0(b')$ and $l_0(b')$ can be represented according to the empirical value bounds $U_0(b')$ and $L_0(b')$, respectively. Moreover, the accurate empirical value bounds $U_0(b')$ and $L_0(b')$ are beneficial to obtain accurate RWDU bounds $\mu_0(b')$ and $l_0(b')$, respectively. The accurate RWDU bounds $\mu_0(b')$ and $l_0(b')$ are beneficial to represent the node's information and reduce the uncertainty for expanding the belief tree effectively. Applying the default policy π_0 at node b' and using the definition of the RWDU function, we have

$$l_0\left(b'\right) = \nu_{\pi_0}\left(b'\right) = \frac{|\Phi_{b'}|}{K} \gamma^{\Delta(b')} L_0\left(b'\right) \tag{7}$$

$$\mu_0\left(b'\right) = \max\left\{ l_0\left(b'\right), \frac{|\Phi_{b'}|}{K} \gamma^{\Delta(b')} U_0\left(b'\right) - \lambda \right\} \tag{8}$$

The initial empirical upper bounds U_0 can be constructed for several methods, such as the uniform bound and hindsight optimization bound. The simple initial empirical upper bound is the *uninformed bound*

$$U_0(b) = \frac{R_{\max}}{1-\gamma} \tag{9}$$

This bound is slack. *Hindsight Optimization* (HO) [34] provided a principled method to construct an upper bound. However, HO may be expensive to compute when the state space is large. In addition, an approximate hindsight optimization bound [29] is calculated by assuming that the states were fully observed, converting the POMDP into a corresponding MDP, and solving for its optimal value function V_{MDP}.

$$U_0(b) = \frac{1}{|\Phi_b|} \sum_{\phi \in \Phi_b} V_{MDP}(s_\phi) \tag{10}$$

In addition to constructing the upper bound, the initial lower bound $L_0(b)$ at node b is defined based on a default policy π_0 by simulating π_0 for a finite number of steps under each scenario and calculating the average total discounted reward. A default policy is usually a random policy or a fixed action policy. Contracting the appropriate initial upper bound and lower bound effectively improves the performance of the proposed method.

2) *Termination of exploration:* To construct a reasonable belief tree within a finite time, the exploration terminates at node b under the following conditions. The first one is, $\Delta(b) > D$, i.e., the search depth reaches the maximum depth of the tree. In addition, $E(b) < 0$, meaning that the expected gap at b is reached and further exploration may be unprofitable. Last, node b is blocked by its ancestor node b', i.e., the number of sampled scenarios is insufficient at ancestor node b':

$$\frac{|\Phi_{b'}|}{K} \gamma^{\Delta(b')} \left(U\left(b'\right) - L_0\left(b'\right) \right) \le \lambda l\left(b', b\right) \tag{11}$$

where $l(b', b)$ is the number of nodes on the path from b' to b.

3) *Backup:* When the exploration is terminated, the belief tree is constructed. A path is obtained from the leaf node to the root node. To reasonably obtain the upper and lower bounds of the node on the path, the upper and lower bounds of a back-up are adjusted slightly according to the initial upper and lower bounds of the node. Then, the upper and lower bounds are backed up for each node b on the path by using the Bellman's principle:

$$\mu(b) = \max\left\{ l_0(b), \max_{a \in A} \left\{ \rho(b,a) + \left(\sum_{z \in Z_{b,a}} \mu\left(b'\right) + \omega_1 \sum_{z \in Z_{b,a}} l\left(b'\right) \right) / \beta_1 \right\} \right\} \tag{12}$$

$$l(b) = \max\left\{ l_0(b), \max_{a \in A} \left\{ \rho(b,a) + \left(\sum_{z \in Z_{b,a}} l\left(b'\right) + \omega_1 \sum_{z \in Z_{b,a}} \mu\left(b'\right) \right) / \beta_1 \right\} \right\} \tag{13}$$

$$U(b) = \max_{a \in A} \left\{ \frac{1}{|\Phi_b|} \sum_{\phi \in \Phi_b} R(s_\phi, a) + \gamma \sum_{z \in Z_{b,a}} \frac{|\Phi_b|}{|\Phi_{b'}|} U\left(b'\right) \right\} \tag{14}$$

where b' is a child of b with $b' = \tau(b, a, z)$. $0 \le \omega_1 \le 1$ is a proportion factor. $\beta_1 = \kappa_1^{\Delta(b)}$ is similar to β. $\kappa_1 > 1$ is a constant. β_1 and ω_1 are usually set to a small value. Otherwise, the lower bound of the root node is approximate for each action.

When the nodes on the path are backed up, the proposed method returns the lower bound of the root node. Meanwhile, a bound is maintained by using the default policy. The optimal action is the action corresponding to the maximum lower bound. Then, the agent executes the optimal action.

4.2 Analysis

Dynamic programming constructs a full DESPOT-DULB D. The anytime forward search algorithm constructs a DESPOT-DULB incrementally and terminates with a partial DESPOT-DULB D', which is a subtree of D. The main purpose of the analysis is to show that the optimal regularized policy $\widehat{\pi}$ derived from D' converges to the optimal regularized policy derived from D.

Theorem 1 proves the choices of the action branch and observation branch in the anytime algorithm. Theorem 1 states that the excess uncertainty at node b is bounded by the sum of the excess uncertainty of its child nodes. DESPOT-DULB provides a greedy way to reduce excess uncertainty by iteratively searching for the action branch and observation branch with the greatest excess uncertainty, establishing Eqs. (4) and (6) as the selection criteria of the action and observation branches, respectively. **Theorem 2** proves the convergence of the DESPOT-DULB. As the gap $\varepsilon(b_0)$ decreases, the calculated policy converges to the optimal policy.

Theorem 1 For any DESPOT-DULB node b, if $E(b) > 0$ and $a^* = \text{argmax}_{a \in A}\, d(b, a)$, then

$$E(b) \leq \sum_{z \in Z_{b,a^*}} E\left(b'\right)$$

where $b' = \tau(b, a^*, z)$ is a child of b. The detailed proof is provided in the Appendix 1.

Theorem 2 For every node b of DESPOT-DULB, we assume that the initial upper bound $U_0(b)$ is $\delta-$ *approximate:*

$$U_0(b) \geq \widehat{V}^*(b) - \delta \tag{16}$$

Suppose that $T_{\max}$ is bounded and that the anytime DESPOT-DULB algorithm terminates with a partial DESPOT-DULB D' that has gap $\varepsilon(b_0)$ between the upper and lower bounds at the root b_0. The optimal regularized policy $\widehat{\pi}$ derived from D' satisfies

$$\nu_{\widehat{\pi}} \geq \nu^*(b_0) - \varepsilon(b_0) - \delta \tag{17}$$

where $\nu^*(b_0)$ is the value of an optimal regularized policy derived from the full DESPOT-DULB D at b_0.

As $T_{\max}$ grows, the uncertainty $\varepsilon(b_0)$ decreases incrementally. The analysis shows that the performance of $\widehat{\pi}$ approaches that of an optimal regularized policy as the running time increases. In addition, the error of the initial upper bound approximation affects the final result at most δ. The detailed proof is provided in the Appendix 2.

5 Results and analysis

This section presents comparative and analytical studies of different POMDP algorithms in the standard POMDP benchmark. To verify that (1) the discounted upper and lower bounds are beneficial to search the optimal action efficiently, (2) the proposed depth function can adjust the gap to ensure a reasonable reduction in uncertainty. The proposed DESPOT-DULB algorithm is evaluated by a computer simulation comparison for the following tasks: a) Tag, b) Laser Tag, and c) Pocman. The DESPOT-DULB is compared with the state-of-the-art online algorithms POMCP, DESPOT and LB-DESPOT [33] to verify (1). Moreover, the DESPOT-ULB is a version that does not consider discounting to verify (2). Similar simulation settings as in [29] are utilized to verify the performance of the algorithm. Considering that small parameters cannot attain an optimal performance and large parameters lead to over-idealization of the algorithm, then, a reasonable range is given for the parameters $\omega \in (0, 0.4)$ and $\kappa \in (1.0, 1.1)$. Through a multi-round test using the trial-and-error method, the initial values of each parameter are selected to guarantee a favorable performance. For Tag, the parameters are set as $\omega = 0.20$, $\omega_1 = 0.02$, $\beta_1 = 1.0$ and $\kappa = 1.02$. For Laser Tag, three tasks of different sizes are designed. The corresponding scenario parameters are set as (1) $\omega = 0.20$, $\omega_1 = 0.02$, $\beta_1 = 1.0$, $\kappa = 1.05$, (2)$\omega = 0.25$, $\omega_1 = 0.025$, $\beta_1 = 1.0$, $\kappa = 1.05$, and (3) $\omega = 0.3$, $\omega_1 = 0.03$, $\beta_1 = 1.0$, $\kappa = 1.05$. For Pocman, the parameters are set as $\omega = 0.003$, $\omega_1 = 0.002$, $\beta_1 = 1.0$ and $\kappa = 1.012$. All experiments were conducted on a computer with Intel(R) Core(TM) i5–6300 HQ, 4 cores running at 2.30 GHz, and 12 G main memory. The operating system of the laptop computer was Ubuntu 18.04. All the algorithms were given exactly 1 second per step to choose an action.

The size of the test domain ranged from small to extremely large. The simulation results are shown in Table 1. Overall, the offline method SARSOP has good performance in the small-scale range but cannot scale up. DESPOT-DULB has a certain performance improvement over DESPOT, POMCP and LB-DESPOT for suitable tasks, especially for large-scale observation space tasks like Laser Tag. Figure 3 shows the average running time ratio in different tasks for DESPOT, DESPOT-DULB and LB-DESPOT. Figure 4 presents the change in the gap between the upper and lower bounds of the optimal node on the current search tree for DESPOT, DESPOT-ULB and DESPOT-DULB. Figure 5 shows the effect of parameters ω and κ on the performance of DESPOT-DULB, where R represents the corresponding reward obtained by running the algorithm 500 times. Reasonable parameters can effectively improve the performance of the algorithm. The details on each domain are described below.

Table 1 Performance comparison

	Tag	Laser Tag(6,8)	Laser Tag(7,11)	Laser Tag(9,12)	Pocman
$\|S\|$	830	1806	4830	9312	$\sim 10^{56}$
$\|A\|$	5	5	5	5	4
$\|Z\|$	30	$\sim 3.5 \times 10^5$	$\sim 1.5 \times 10^6$	$\sim 4.1 \times 10^6$	1024
SARSOP	−6.04±0.15	–	–	–	–
POMCP	−13.33±0.49	−13.79±0.58	−14.81±0.59	−17.27±0.47	273.68±9.57
DESPOT	−6.42±0.27	−6.24±0.38	−8.46±0.42	−11.42±0.37	317.43±8.88
LB-DESPOT	−5.95±0.25	−5.92±0.40	−7.93±0.42	−10.62±0.40	322.23±8.91
DESPOT-ULB	−5.73±0.26	−5.96±0.38	−7.92±0.40	−10.93±0.41	321.25±8.74
DESPOT-DULB	−5.61±0.26	−5.31±0.36	−7.68±0.40	−10.25±0.40	323.83±8.50

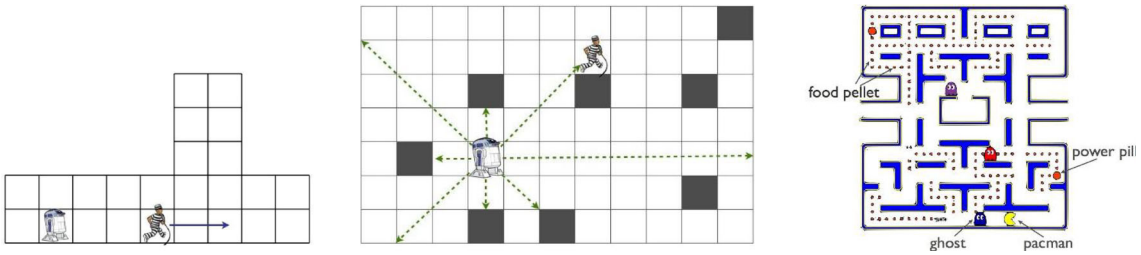

Fig. 2 Three test domains. (**a**) Tag. A robot chases an unobserved target that runs away. (**b**) Laser Tag. A robot chases a target in a 7 × 11 grid environment populated with obstacles. The robot is equipped with a laser range finder for self-localization. (**c**) Pocman. The original Pacman game

5.1 Tag

Tag is a standard POMDP benchmark. A robot and a target move in a grid with 29 possible positions (Fig. 2a). The goal of the robot is to find the target that consciously escapes. For the Tag environment, the specific description is as follows. At first, both the robot and the target obtain a random position. The robot knows its position, but the robot cannot obtain the position of the target. The robot can observe the target's position when the robot and the target are in the same position. Moreover, the robot tries to choose an action from five actions: stay in place and move in four adjacent directions, when paying −1 for each step. Finally, the robot attempts to tag the target, rewarding it with +10 if the attempt is successful and punishing it with −10 otherwise.

Table 1 shows the obtained average total discounted rewards for several POMDP algorithms. On this small-size domain, SARSOP achieves the best result. The proposed DESPOT-DULB exhibits strong competitiveness in Tag compared with the other algorithms. Figure 3 shows that DESPOT-DULB requires less planning time than DESPOT and LB-DESPOT in Tag. Figure 4a shows that DESPOT-DULB has a smaller gap between the upper and lower bounds than DESPOT and DESPOT-ULB. Meanwhile, DESPOT-DULB can suppress the sharp rise of the gap by discounting compared with DESPOT-ULB without discounting. Combined with Theorem 2, the calculated policy is closer to the optimal policy than DESPOT under the case of a constant δ. Figure 5a shows the node's gap curve of DESPOT-DULB and the corresponding reward when the proportion factor ω is 0.1, 0.2, 0.25 and 0.3. When the proportion factor ω increases, the gap curve has large fluctuations and the corresponding reward decreases gradually. Reducing the proportion factor ω is one of the methods to improve the gap but DESPOT-DULB needs a reasonable proportion factor to represent the node's information. Figure 5(d) presents the node's gap curve and the corresponding reward of DESPOT-DULB when the parameter κ is 1.0, 1.02, and 1.04. As the parameter κ increases, the gap can smoothly converge to a small value but the obtained reward increases first and then decreases. Choosing a suitable κ is a good way to reduce the gap and obtain the maximum reward. Based on the above analysis, the parameters $\omega = 0.20$ and $\kappa = 1.02$ are chosen to obtain a maximum reward and small uncertainty for DESPOT-DULB.

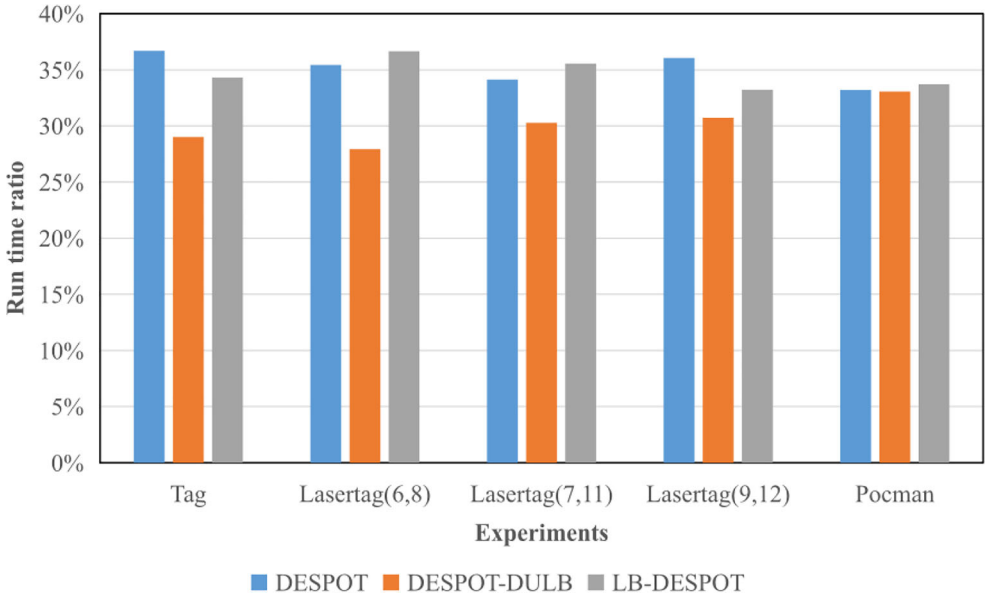

Fig. 3 Comparison of the average running time ratio of algorithm in different tasks

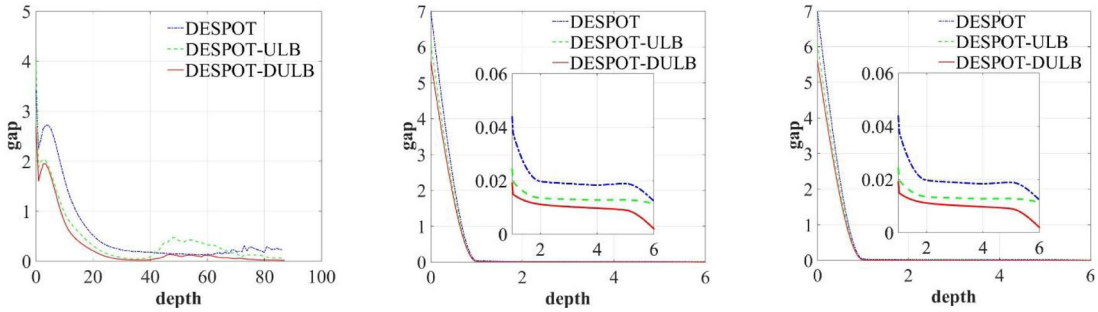

Fig. 4 The gap between the upper and lower bound of the optimal node on the current search tree. (**a**) Tag. (**b**) Laser Tag. (**c**) Pocman. The blue line, green line and red line represents DESPOT algorithm, DESPOT-ULB algorithm and DESPOT-DULB algorithm, respectively.

5.2 Laser tag

Laser Tag is an expanded version of Tag with a large observation space. In Laser Tag, the robot moves in a $n \times m$ rectangular grid with o randomly placed obstacles. Three scenes of different sizes are set as follows: a 6×8 grid with 6 randomly placed obstacles, a 7×11 grid with 8 randomly placed obstacles (Fig. 2b), and a 9×12 grid with 12 randomly placed obstacles. The settings of the robot and target are the same as those of Tag. However, the robot does not know its own position and the robot is initially distributed uniformly over the grid. To localize, the robot is equipped with a laser range finder that measures the distances in eight directions. The side length of each cell is set to 1. The laser reading in each direction is generated from a normal distribution centered at the true distance of the robot to the nearest obstacle in that direction, with a standard deviation of 2.5. The readings are rounded to the nearest integers. Hence, an observation comprises a set of eight integers and the total number of observations is approximately 3.5×10^5, 1.5×10^6 and 4.1×10^6, respectively.

With the large observation space, the SARSOP algorithm cannot run successfully. Table 1 shows that DESPOT-DULB has substantially better quality than DESPOT, POMCP, LB-

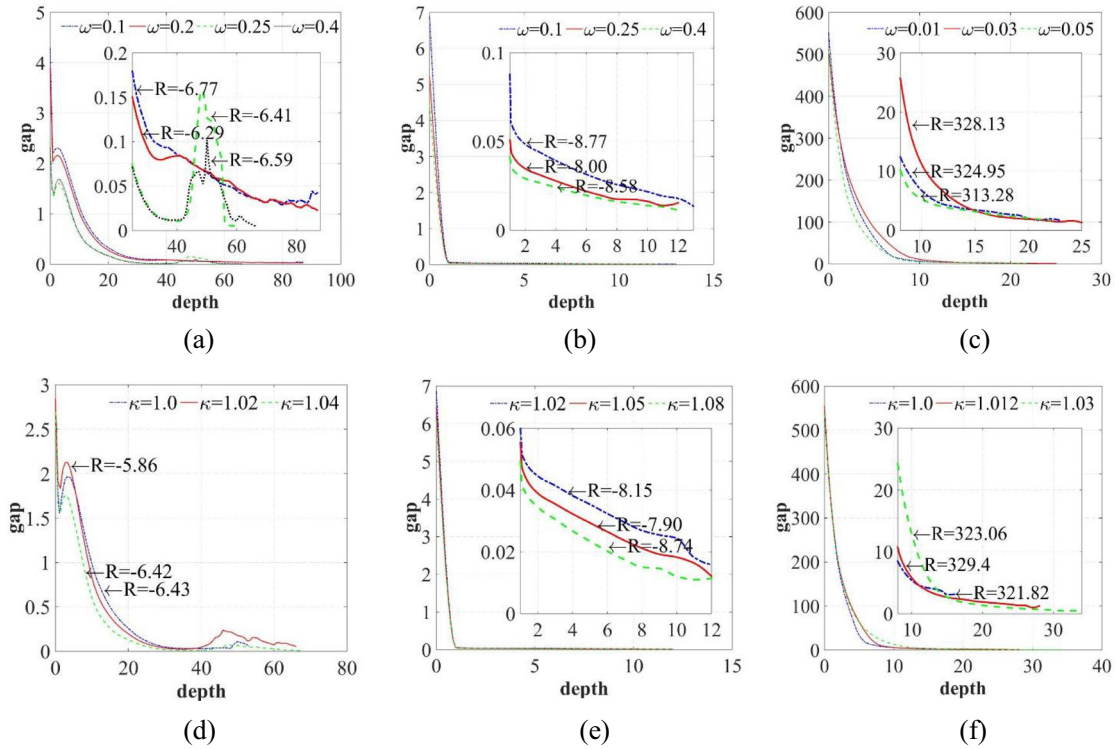

Fig. 5 The analysis of the parameters ω and κ. (**a**), (**b**) and (**c**) correspond the change of parameter ω for Tag, Laser Tag and Pocman respectively. (**d**), (**e**) and (**f**) correspond the change of parameter κ for Tag, Laser Tag and Pocman respectively. R represents the average total discounted rewards by simulating 500 times for each tasks under the corresponding parameters.

DESPOT and DESPOT-ULB according to the obtained average discounted rewards. Figure 3 shows that DESPOT-DULB requires less planning time than DESPOT and LB-DESPOT for different sizes of laser tags. Figure 4b shows that DESPOT-DULB provides a smaller gap between the upper and lower bounds of the current node than DESPOT at the same depth. Similar to Tag, the obtained policy is closer to the optimal policy than DESPOT when δ is determined. Figure 5b shows the node's gap curve and the obtained reward of DESPOT-DULB for a 7 × 11 grid of Laser Tag when the proportion factor ω is 0.1, 0.25 and 0.4. Figure 5e shows the node's gap curve and the obtained reward of DESPOT-DULB for a 7 × 11 grid of Laser Tag when the proportion factor κ is 1.02, 1.05 and 1.08. As the parameters ω or κ increase, the gap of the node drops significantly at the same depth and the corresponding reward increases first and then decreases. Decreasing ω or κ is beneficial for reducing the gap and obtaining the maximum reward. Based on the above considerations, the parameters ω and κ are selected as 0.25 and 0.15, respectively.

5.3 Pocman

Pocman [14] is a partial observation variant of the popular video game Pacman (Fig. 2c). In Pocman, an agent and four ghosts move in a 17 × 19 maze populated with food pellets. If the agent eats a food pellet, the agent obtains a reward of +10. The agent costs −1 for each move. If the agent is caught by a ghost, the game terminates with a penalty of −100. Moreover, there are four power pills in the maze. The agent can eat a ghost and receive a reward of +25 within the next 15 steps after eating a power pill. A ghost chases the agent if the agent is within a Manhattan distance of 5. A ghost runs away if the agent is in a state of eating a power pill. In addition, the agent does not know the exact position of the ghost. However, the agent receives information on whether (1) it sees the ghost in each of the cardinal directions, (2) it hears the ghost within a Manhattan distance of 2, (3) it feels a wall in each of the four cardinal directions, and (4) it smells food pellets in the adjacent or diagonally adjacent cells. Pocman has a large state space of approximately 10^{56} states.

On this large-scale domain, Table 1 shows that DESPOT-DULB has a slight improvement compared to DESPOT and DESPOT-ULB according to the obtained average rewards, while the SARSOP algorithm cannot run successfully. Figure 3 shows that all the algorithms, DESPOT-DULB, LB-DESPOT and DESPOT, take almost the same amount of time because the agent needs to execute the same number of steps for each simulation. Figure 4c shows that DESPOT-DULB converges to a smaller gap and a lower depth than DESPOT and DEPOT-ULB. A lower uncertainty is beneficial in determining the state of the agent. Then, the agent can obtain a near-optimal policy. Figure 5c shows the node's gap curve and the corresponding reward of DESPOT-DULB when the proportion factor ω is 0.01, 0.03 and 0.05. A smaller uncertainty can be obtained by increasing the proportion factor ω but if the proportion factor is too large, the maximum reward cannot be obtained. After comprehensive consideration, the parameter $\omega = 0.03$ is chosen to implement the testing. Figure 5f shows the node's gap curve and the corresponding reward of DESPOT-DULB when the parameter κ is 1.0, 1.012 and 1.03. As the parameter κ increases, the gap can converge to a small value, but the ultimate depth is large, resulting in a long planning time. For a reward, the parameters of $\omega = 0.03$ and $\kappa = 1.012$ are chosen to obtain the maximum reward and a lower uncertainty.

6 Conclusion and future work

This paper has proposed the online POMDP algorithm DESPOT-DULB, which has considered that the node's upper bound cannot adequately represent all the node information. The proposed DESPOT-DULB contains the combination of the upper and lower bounds of the node and a depth function based on the current depth. The combination of the upper and lower bounds of the node has represented the node information to improve the quality of the forward search. Meanwhile, a depth function has been considered to adjust the gap between the upper bound and lower bound of the node to ensure a reasonable reduction in uncertainty. With the support of computer simulation comparisons using standard POMDP benchmarks, both the simulation and theoretical analysis have shown that DESPOT-DULB not only retains the DESPOT's desirable properties for online planning but also shows a certain improvement in the quality of policy and the efficiency of search.

There are two potential directions to expand this work. First, an appropriate heuristic is considered to express the amount of information of the upper and lower bounds of the node, e.g., the information entropy. Therefore, the node's accurate information can be used to search the optimal node to obtain the maximum reward. Second, a reasonable approach is premeditated to obtain the compact upper and lower bounds. The compact upper and lower bounds can improve the planning efficiency.

Appendix 1

Proof Let $CH(b, a^*)$ be all the set of $b' = \tau(b, a, z)$ for some $z \in Z_{b,a^*}$, that is, the set of children nodes of b in the DESPOT-DULB tree. If $E(b) > 0$, then $\varepsilon(b) > 0$, that is, $\mu(b) - l(b) > 0$, and thus $\mu(b) \neq l_0(b)$. Hence we have

$$\mu(b) = d(b,a^*) = \rho(b,a^*) + \left(\sum_{b'\in CH(b,a^*)} \mu(b') + \omega \sum_{b'\in CH(b,a^*)} l(b')\right)/\beta \tag{A1}$$

and

$$l(b) \geq m(b,a^*) \geq \rho(b,a^*) + \left(\sum_{b'\in CH(b,a^*)} l(b') + \omega \sum_{b'\in CH(b,a^*)} \mu(b')\right)/\beta \tag{A2}$$

where $0 < \omega < 1$, and $\beta > 1$. Subtracting the Eq. (A1) by the Eq. (A2), we have

$$\mu(b) - l(b) \leq (1-\omega) \sum_{b'\in CH(b,a^*)} \left[\mu(b') - l(b')\right]/\beta \tag{A3}$$

where

$$\beta = \kappa^{\Delta(b)} \tag{A4}$$

Considering $0 < 1 - \omega < 1$, $\kappa > 1$, we have

$$(1-\omega) \sum_{b'\in CH(b,a^*)} \left[\mu(b') - l(b')\right]/\beta < \sum_{b'\in CH(b,a^*)} \left[\mu(b') - l(b')\right] \tag{A5}$$

That is

$$\mu(b) - l(b) \leq \sum_{b'\in CH(b,a^*)} \left[\mu(b') - l(b')\right] \tag{A6}$$

Combining the Eq. (A4) and Eq. (A6), we have

$$\kappa^{\Delta(b)}[\mu(b) - l(b)] \leq \kappa^{\Delta(b)} \sum_{b'\in CH(b,a^*)} \left[\mu(b') - l(b')\right] \leq \kappa^{\Delta(b')} \sum_{b'\in CH(b,a^*)} \left[\mu(b') - l(b')\right] \tag{A7}$$

Note that

$$\frac{|\Phi_b|}{K}\xi\varepsilon(b_0) = \sum_{b'\in CH(b,a^*)} \frac{|\Phi_{b'}|}{K}\xi\varepsilon(b_0) \tag{A8}$$

Hence, we have

$$\kappa^{\Delta(b)}[\mu(b) - l(b)] - \frac{|\Phi_b|}{K}\xi\varepsilon(b_0) \leq \sum_{b'\in CH(b,a^*)} \left\{\kappa^{\Delta(b')}\left[\mu(b') - l(b')\right] - \frac{|\Phi_{b'}|}{K}\xi\varepsilon(b_0)\right\} \tag{A9}$$

That is, $E(b) \leq \sum_{z\in Z_{b,a^*}} E(b')$.

Appendix 2

Proof Let $U_0'(b) = U_0(b) + \delta$, then U_0' is an exact upper bound. Let μ_0' be the corresponding initial upper bound, and μ' be the corresponding upper bound on $\nu^*(b)$. Then μ_0' is a valid initial upper bound for $\nu^*(b)$ and the backup equations ensure that $\mu'(b)$ is a valid upper bound for $\nu^*(b)$. On the other hand, it is easily shown by induction that

$$\mu(b) + \gamma^{\Delta(b)}\frac{|\Phi_b|}{K}\delta \geq \mu'(b) \tag{B10}$$

When a special case for $b = b_0$, we have

$$\mu(b) + \delta \geq \mu'(b_0) \tag{B11}$$

Hence, when the algorithm terminates, we have

$$\mu(b_0) + \delta \geq \mu'(b_0) \geq \nu^*(b_0) \tag{B12}$$

Equivalently,

$$\nu_{\widehat{\pi}} = l(b_0) \geq \nu^*(b_0) - (\mu(b_0) - l(b_0)) - \delta = \nu^*(b_0) - \varepsilon(b_0) - \delta \tag{B13}$$

The Eq. (B13) holds because the initialization and the computation of the lower bound l via the backup equations are exactly that for finding a regularized optimal policy value in the partial DESPOT-DULB.

Acknowledgments This work was supported in part by the National Key Research and Development Program of China (No. 2021ZD0114503), the Major Research plan of the National Natural Science Foundation of China (No. 92148204), the National Natural Science Foundation of China (No. 61803089, 61971071, 62027810, 62133005), the Hunan Science Fund for Distinguished Young Scholars (No. 2021JJ10025), the Hunan key research and development program (No. 2021GK4011, 2022GK2011), the Changsha Science and Technology Major Project (No. kh2003026), the Joint Open Foundation of State Key Laboratory of Robotics (No. 2021-KF-22-17), the Tianjin University-Fuzhou University Independent Innovation Fund (No. TF2022-4), the China University industry-University-research Innovation Fund (No. 2020HYA06006).

References

1. Dai XY, Meng QH, Jin S (2021) Uncertainty-driven active view planning in feature-based monocular vSLAM. Appl Soft Comput 108:107459
2. Nakrani NM, Joshi MM (2022) A human-like decision intelligence for obstacle avoidance in autonomous vehicle parking. Appl Intell 52(4):1–20
3. Hubmann C, Schulz J, Becker M, Althoff D, Stiller C (2018) Automated driving in uncertain environments: planning with interaction and uncertain maneuver prediction. IEEE Trans Intell Veh 3(1):5–17
4. Smallwood R, Sondik E (1973) The optimal control of partially observable Markov processes over a finite horizon. Oper Res 21: 1071–1088
5. Bai H, Cai S, Ye N, Hsu D, Lee WS (2015) Intention-aware online POMDP planning for autonomous driving in a crowd. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), pp 454–460
6. Garg NP, Hsu D, Lee WS (2019) Learning to grasp under uncertainty using POMDPs. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), pp 2751–2757
7. Wu K, Lee WS, Hsu D (2015) POMDP to the rescue: boosting performance for Robocup rescue. In: proceedings of the IEEE international conference on intelligent robots and systems (IROS), pp 5294–5299
8. Folsom-Kovarik JT, Sukthankar G, Schatz S (2013) Tractable POMDP representations for intelligent tutoring systems. ACM Trans Intell Syst Technol 4(2):1–22
9. Kaelbling LP, Littman ML, Cassandra AR (1998) Planning and acting in partially observable stochastic domains. Artif Intell 101(1–2):99–134
10. Deb S, Tammi K, Gao XZ, Kalita K, Mahanta P, Cross S (2022) A robust two-stage planning model for the charging station placement problem considering road traffic uncertainty. IEEE Trans Intell Transp Syst 23(7):1–15
11. Sung I, Choi B, Nielsen P (2021) On the training of a neural network for online path planning with offline path planning algorithms. Int J Inf Manag 57:102142
12. Nicol S, Chads I (2012) Which states matter? An application of an intelligent discretization method to solve a continuous POMDP in conservation biology. PLoS One 7(2):e28993
13. Browne CB, Powley E, Whitehouse D, Lucas SM, Cowling PI, Rohlfshagen P, Tavener S, Perez D, Samothrakis S, Colton S (2012) A survey of Monte Carlo tree search methods. IEEE Trans Comput Intell AI in Games 4(1):1–43
14. Silver D, Veness J (2010) Monte-Carlo planning in large POMDPs. Adv Neural Inf Proces Syst 23:2164–2172
15. Auer P, Cesa-Bianchi N, Fischer P (2002) Finite-time analysis of the multiarmed bandit problem. Mach Learn 47(2–3):235–256
16. Somani A, Ye N, Hsu D, Lee WS (2013) DESPOT: online POMDP planning with regularization. Adv Neural Inf Proces Syst 58:231–266
17. Bougie N, Ichise R (2021) Fast and slow curiosity for high-level exploration in reinforcement learning. Appl Intell 51(2):1086–1107
18. Chen Y, Kochenderfer MJ, Spaan MTJ (2018) Improving offline value-function approximations for POMDPs by reducing discount factors. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp 3531–3536
19. Kurniawati H, Hsu D, Lee WS (2008) SARSOP: efficient point-based POMDP planning by approximating optimally reachable belief spaces. Robot: Sci Syst 4:65–72
20. Bai H, Hsu D, Lee WS (2014) Integrated perception and planning in the continuous space: a POMDP approach. Int J Robot Res 33(9): 1288–1302
21. Zhang Z, Hsu D, Lee WS, Lim ZW, Bai A (2015) Please: palm leaf search for pomdps with large observation spaces. In: Proceedings of the Twenty-Fifth International Conference on Automated Planning and Scheduling, pp. 249–258
22. Wu B, Zheng HY, Feng YP (2014) Point-based online value iteration algorithm in large POMDP. Appl Intell 40(3):546–555
23. He R, Brunskill E, Roy N (2011) Efficient planning under uncertainty with macro-actions. J Artif Intell Res 40:523–570
24. Ross S, Pineau J, Paquet S, Chaib-Draa B (2008) Online planning algorithms for POMDPs. J Artif Intell Res 32:663–704
25. Zhang S, Sridharan M, Washington C (2013) Active visual planning for mobile robot teams using hierarchical pomdps. IEEE Trans Robot 29(4):975–985
26. Koval M, Hsu D, Pollard N, Srinivasa SS (2020) Configuration lattices for planar contact manipulation under uncertainty. In: Proceedings of International Workshop on the Algorithmic Foundations of Robotics, pp. 768–783
27. Sun K, Schlotfeldt B, Pappas GJ (2020) Stochastic motion planning under partial observability for mobile robots with continuous range measurements. IEEE Trans Robot 37(3):979–995
28. Vien NA, Ngo H, Lee S, Chung T (2014) Approximate planning for Bayesian hierarchical reinforcement learning. Appl Intell 41(3): 808–819
29. Ye N, Somani A, Hsu D, Lee WS (2017) DESPOT: online POMDP planning with regularization. J Artif Intell Res 58:231–266
30. Garg NP, Hsu D, Lee WS (2019) DESPOT-alpha: online POMDP planning with large state and observation spaces. Robot: Sci and Syst. https://doi.org/10.15607/RSS.2019.XV.006
31. Luo Y, Bai H, Hsu D, Lee WS (2019) Importance sampling for online planning under uncertainty. Int J Robot Res 38(2–3):162–181
32. Cai P, Luo Y, Hsu D, Lee WS (2021) HyP-DESPOT: a hybrid parallel algorithm for online planning under uncertainty. Int J Robot Res 40(2–3):558–573
33. Wu C, Kong R, Yang G, Kong X, Zhang Z, Yu Y, Liu W (2021) LB-DESPOT: efficient online POMDP planning considering lower bound in action selection. In: Proceedings of the AAAI Conference on Artificial Intelligence 35(18):15927–15928
34. Yoon S, Fern A, Givan R, Kambhampati S (2008) Probabilistic planning via determinization in hindsight. In: Proceedings of AAAI Conference on Artificial Intelligence 2:1010–1016

Yanjie Chen received the B.S. degree in electrical engineering and its automation from Southwest Jiaotong University, Chengdu, China in 2011, the M.S. and Ph.D. degrees in control science and engineering from Hunan University, Changsha, China, in 2013 and 2017, respectively. From 2017 to 2021, he was an Assistant Professor with School of Mechanical Engineering and Automation, Fuzhou University, Fuzhou, China. He is currently an Associate Professor with School of Mechanical Engineering and Automation, Fuzhou University, Fuzhou, China. He is also Associate Scientist with the National Engineering Research Center of Robot Visual Perception and Control Technology, Changsha, China. His research interests include robotics, unmanned aerial manipulator, motion planning and artificial intelligence.

Hui Zhang received the B.S., M.S., and Ph.D. Degrees in pattern recognition and intelligent system from Hunan University, Changsha, China, in 2004, 2007, and 2012, respectively. He is currently a professor with the School of Robotics, Hunan University, and he is the Deputy Director of the National Engineering Research Center of Robotic Vision Perception and Control Technology. He was a Visiting Scholar with Common Vulnerability Scoring System Laboratory, Department of Electrical and Computer Engineering, University of Windsor, Windsor, ON, Canada, in 2017. His research interests include machine vision, sparse representation, and visual tracking.

Jiangjiang Liu received the B.S. degree in mechanical design manufacture and automation from Jiangxi University of Science and Technology, Ganzhou, China, in 2020, and he is currently working toward the M.S. degree in mechanical engineering from Fuzhou University, Fuzhou, China. His research interests include motion planning and mobile robot.

Yaonan Wang received the B.S. degree in computer engineering from East China University of Science and Technology, Fuzhou, China, in 1981 and the M.S. and Ph.D. degrees in control engineering from Hunan University, Changsha, China, in 1990 and 1994, respectively. He was a Post-Doctoral Research Fellow with the National University of Defense Technology, Changsha, from 1994 to 1995, a Senior Humboldt Fellow in Germany from 1998 to 2000, and a Visiting Professor with the University of Bremen, Bremen, Germany, from 2001 to 2004. He has been a Professor with Hunan University since 1995. He has been an academician of China Engineering Academy since 2019. His research interests include robot control, intelligent control and information processing, industrial process control, and image processing.

Yibin Huang received the B.S. degree in mechanical engineering from Huaqiao University, Quanzhou, China, in 2018, and the M.S. degree in mechanical engineering from Fuzhou University, Fuzhou, China, in 2022. His research interests include motion aerial manipulator robot and motion planning.