Anytime Probabilistically Constrained Provably Convergent Online Belief Space Planning

Andrey Zhitnikov and Vadim Indelman, *Member, IEEE*

***Abstract*—Taking into account future risk is essential for an autonomously operating robot to find online not only the best but also a safe action to execute. In this article, we build upon the recently introduced formulation of probabilistic belief-dependent constraints. In our methodology, safety can be materialized with any general belief-dependent operator we call payoff. We present an anytime approach employing the Monte Carlo Tree Search (MCTS) method in continuous domains in terms of states, actions, and observations and general-belief-dependent reward and payoff operators. Unlike previous approaches, our method ensures safety anytime with respect to the currently expanded search tree without relying on the convergence of the search. We prove convergence in probability with an exponential rate of a version of our algorithms and study proposed techniques via extensive simulations. Even with a tiny number of tree queries, the best action found by our approach is much safer than the baseline. Moreover, our approach constantly yields better than the baseline action in terms of objective function. This is because we revise the values and statistics maintained in the search tree and remove from them the contribution of the pruned actions. We rigorously show that our cleaning routine is necessary. Without it, at the limit of convergence of MCTS, an infinite amount of sampled dangerous actions can be detrimental to the objective function.**

***Index Terms*—Anytime constraint satisfaction, belief-dependent constraints, belief space planning, Monte Carlo Tree Search (MCTS).**

Key Symbols and Interpretation

$\mathcal{X}, \mathcal{A}, \mathcal{Z}$	State, Action, and Observation spaces.
x_ℓ,a_ℓ,z_ℓ	Momentary state, action, and observation, respectively.
$\mathbf{1}_A()$	Indicator function defined on set A.
$b_\ell^-(x_\ell)$	Propagated belief at time instance ℓ defined by (2).
$b_\ell(x_\ell)$	Posterior belief at time instance ℓ defined by (1).
$\bar{b}_\ell^-(x_\ell)$	Propagated belief at time instance ℓ defined by (25).
$\bar{b}_\ell^{\text{safe}}(x_\ell)$	Safe posterior belief at time instance ℓ defined by (23).
$\bar{b}_\ell(x_\ell)$	Posterior belief at time instance ℓ conditioned on safety events at times $0{:}\ell{-}1$ (defined by (24)).
ρ	General belief-dependent reward operator.
ϕ	General belief-dependent payoff operator (to be as large as required).
θ	General belief-dependent cost operator (to be as small as required).
Φ_ℓ^δ	Set of propagated and posterior beliefs at time ℓ that satisfy the constraint.
$\bar{\Phi}_\ell^\delta$	Set of propagated and posterior beliefs at time ℓ, conditioned on safety events at times $0{:}\ell{-}1$, which satisfy the constraint.
BMCTS-DPW	Generic MCTS-DPW running on top of BMDP.
PFT-DPW	Generic BMCTS-DPW with belief update being a particle filter.
PF-PUCT	Algorithm 1; Modified PFT-DPW to satisfy guarantees provided by [18].
PC-SBMCTS-DPW	Algorithm 3; Probabilistically constrained safe beliefs (SB) MCTS-DPW.
PC-MCTS-DPW	Variant of Algorithm 3 but without SB.
PC-SBMCTS-DPW	Provable variant of Algorithm 3.
PC-SB-PFT-DPW	PC-SBMCTS-DPW with particle beliefs.
PC-PFT-DPW	PC-MCTS-DPW with particle beliefs.
AL	Autonomy loop.
n_x	Number of belief particles.

Received 10 April 2025; revised 11 August 2025; accepted 26 August 2025. Date of publication 16 September 2025; date of current version 8 December 2025. This work was supported by Israel Science Foundation (ISF). This article was recommended for publication by Associate Editor J. Pan and Editor S. Behnke upon evaluation of the reviewers' comments. *(Corresponding author: Andrey Zhitnikov.)*

Andrey Zhitnikov is with the Technion Autonomous Systems Program (TASP), Technion—Israel Institute of Technology, Haifa 32000, Israel (e-mail: andreyz@campus.technion.ac.il).

Vadim Indelman is with the Stephen B. Klein Faculty of Aerospace Engineering, Technion—Israel Institute of Technology, Haifa 32000, Israel, and also with the Faculty of Data and Decision Sciences, Technion—Israel Institute of Technology, Haifa 32000, Israel (e-mail: vadim.indelman@technion.ac.il).

This article has supplementary downloadable material available at https://tinyurl.com/4cbtffxc, provided by the authors.

This article has supplementary downloadable material available at https://doi.org/10.1109/TRO.2025.3610176, provided by the authors.

Digital Object Identifier 10.1109/TRO.2025.3610176

I. Introduction and Related Work

CASTING decision-making under uncertainty as a Partially Observable Markov Decision Process (POMDP) is considered State Of The Art (SOTA). Under partial observability, the decision-making agent does not have complete information about the state of the problem, so it can only make its decisions based on its "belief" about the state. In a continuous domain in terms of POMDP state, the belief, in a particular time index, is the Probability Density Function (PDF) of the state given all concurrent information in terms of performed actions and

received observations in an alternating manner, plus the prior belief. A POMDP is known to be undecidable [1] in finite time.

Introducing various constraint formulations into POMDP is essential for, e.g., ensuring safety [2], [3] and efficient autonomous exploration [4]. Yet, the existing online anytime approaches have problems and therefore fall short of providing reliable and safe optimal autonomy. This crucial gap we aim to fill in this article. To specify, the problems are along these lines: dangerous actions participate in the search tree, safety is ensured merely at the limit of Monte Carlo Tree Search (MCTS) convergence while the limit is never reached, learning methods are not reliable due to the sim-to-real problem and the generalization error, and learning components may be slow.

Our method constructs an MCTS search tree and uses the tree to represent the stochastic POMDP policy. However, in contrast to other methods, we prune dangerous actions from the belief tree and revise the values and statistics that the MCTS search tree maintains. Anytime, our search tree contains only the safe actions in accordance with our definition of safe action, which will appear shortly in Section IV. Our work lies in continuous states, actions, and observations domains. In such a setting, there are approaches to tackle averaged cumulative constraint using anytime MCTS methods [5], [6]. We now linger on the explanation of what the averaged constraint is.

In the POMDP setting, there are naturally two stages to consider in order to introduce a constraint. The first stage arises from the belief itself. Usually, at this stage, the state-dependent payoff operator is averaged with respect to the corresponding belief to obtain a belief-dependent one. It is then summed up to achieve a cumulative payoff. We use the term *payoff* to differentiate between the reward operator and emphasize that a belief-dependent payoff constraint operator shall be as large as desired as opposed to the *cost* operator. The second stage arises from the distribution of possible future observation episodes. At this stage, commonly, the cumulative payoff is again averaged but with respect to future observations episodes and then thresholded, thereby forming an averaged cumulative constraint. Such a formulation is sufficient for ensuring safety in limited cases, as we will further see in Section IX-A. This is because it permits deviations of the individual values within the summation.

Let us now describe the MCTS methods mentioned above to tackle averaged cumulative constraint. The seminal paper in this direction is [7]. It leans on the rearrangement of the constrained objective using the occupancy measure described in [8]. Such a reformulation is appealing since it transforms the problem into linear programming, bringing convexity to the table and enjoying strong duality. Jamgochian et al. [5] extended the approach from [7] to continuous spaces. Still, both papers [5] and [7] ensure constraint satisfiability only at the limit of the convergence of the iterative procedure, namely, in infinite time. Since these are iterative methods, to assure anytime constraint satisfiability we need to project the obtained occupancy measure at each iteration to the safe space defined by the constraint. If dual methods are involved [9], such a projection does not make much sense, e.g., the projection might lead to a step direction vector on the boundary of all the constraints, making it a zero vector. Employing the primal methods in continuous spaces also appears to be problematic since the summations in [7] are transformed into integrals. Jamgochian et al. [6] provided some sort of anytime satisfiability by introducing high-level action primitives (options). Still, Jamgochian et al. [6] suffered from limitations, e.g., it requires crafting low-level policies, meaning knowing how the robot shall behave a priori. In addition, the options shall be locally feasible. Furthermore, for efficiency reasons, the duality-based approaches perform a single tree query of the MCTS instead of running MCTS until convergence, which will happen in infinite time in the maximization of the Lagrangian dual objective function phase (see [9, Sect 8.5.2]) of dual ascend.

In all three papers [5], [6], and [7], the averaged cumulative constraint is enforced solely from the root of the belief tree. This is made possible in the context of reactive policies since the threshold itself is irrelevant while optimizing an augmented value function (with a frozen Lagrange multiplier) in the MCTS phase. Refer to [5, Listing 1, Line 10]. Such an approach is suboptimal since within a planning session, it is not taken into account that the constraint will be enforced at the future planning sessions. The contemplation of a robot about the future differs from its actual future behavior. This aspect has been fixed by Ho et al. [10]. As we will further see in Section IV, our approach naturally handles this problem. Moreover, Ho et al. [10] assured fulfillment (admission) of the recursive averaged cumulative constraint anytime with respect to search tree constructed partially with the reward bounds and partially with rewards themselves. Yet, the algorithm presented in [10] requires that the value function be bounded on the way down the tree to assure the exploration. This is commonly achieved by assuming that the state-dependent reward is trivially bounded from above and below. General belief-dependent reward functions typically lack trivial bounds. Moreover, the exploration outlined in that paper is valid for discrete spaces only. All in all, the extension of that work to continuous spaces and belief-dependent rewards requires clarification.

a) Support for general belief-dependent rewards and payoff/cost operators and MCTS convergence: We now clarify whether or not the above-mentioned solvers support belief-dependent cost/payoff operators and rewards. It was suggested in [3] and [4] that general belief-dependent payoff/cost operators are extremely important. As mentioned in [3], Value-at-Risk (VaR) and conditional VaR over the distance to the safe space allow for control of the depth the robot can plunge into the obstacle. These operators measure how bad the disaster (collision) will be[1]. The information gain discussed in [4] is relevant for exploration. Zhitnikov and Indelman [4] discussed the general belief-dependent averaged constraint of the form (37) in a high-dimensional setting and in the context of information gain. The iterative schemes in [5] and [7] lean on the convergence of MCTS. It has been shown in [11] that even in discrete spaces and with bounded rewards, it can take a very long time for MCTS to converge. This is because, if such an augmented reward has a large variance, it will need a huge amount of tree queries for the action-value estimate (to be defined shortly in Section II-C) at

[1]See Section 1 of the Supplementary Material, for details.

each belief node of the belief tree to converge. The large variance can be the result of an unrestrained variability of the rewards or a large Lagrange multiplier. In the case of an unbounded reward, e.g., differential entropy, or the cost-augmented objective of [5] and [7], the MCTS may not converge at all.

There are several constraint formulations for POMDP. We discuss the most prominent techniques one by one in the following.

b) Shielding POMDPs: There is a growing body of literature on shielding POMDPs. The shield is a technique to disable the actions that can be executed by the agent and violate the shield definition. There are several shield definitions. Online methods [12], [13] in this category utilize the Partially Observable Monte-Carlo Planning (POMCP) algorithm [14]. These works have the same problems we are solving in this article: one way or another, the actions violating the shield definition participate in the planning procedure, yielding a suboptimal result. As we further show in Section VI-C, not considering safety in future times within the planning session can lead not only to dangerous but also to a suboptimal planning result.

c) Chance Constrained (CC) Online Planning: A recent work [15] tackles online planning with chance constraints in an anytime setting. This article suggests using a Neural Network (NN) to approximate CC enforced, with an adaptive threshold, from each belief considered in the planning session. This work trains NN offline to provide an initial estimator for CC and refines it online. Therefore, the error stemming from the discrepancy of simulated and real data is unknown. Moreover, it is not clear how complex the NN shall be to achieve zero loss in training to ensure no error in CC approximation, so even if no discrepancy discussed before exists, the NN inference may be slow. In this method, dangerous actions may participate in the planning session, namely, in the search tree. This is because the CC approximator is improved as the tree search progresses, and when revisiting the belief node, a previously safe action can be deemed dangerous. No cleaning has been done in this method as opposed to our approach.

d) Safe control under partial observability: There are a variety of robust control approaches natively tailored for continuous state/action/observation spaces [16], [17]. However, these methods are usually limited to very specific rewards/objectives and tasks, such as reaching a goal state or being as close as possible to a nominal trajectory. Moreover, in both papers, the system dynamics are control-affine. Without this assumption, it is not clear how to enforce the constraint through a derivative of the barrier function.

B. Contributions

We list down our contributions in the same order as they appear in this article in the following.

1) By directly constraining the problem space and not the dual space, we present an anytime MCTS-based algorithm for safe online decision-making with safety harnessing the general belief-dependent operators and governed by a probabilistic constraint (PC). Our approach enjoys anytime safety guarantees with respect to the belief tree expanded so far and works in continuous state, action, and observation spaces. When stopped anytime, the action returned can be considered as the best safe action under the safe future policy (tree policy) expanded so far. Our search tree *solely* consists of safe actions.
2) We prove convergence in probability with an exponential rate of our approach with constraints and without (see Section VI). Even without our constraints, to the best of the authors' knowledge, we are the first to apply provable MCTS innovated by [18] on Belief-Markov decision processes (BMDPs).
3) Another contribution on our end is constraining the beliefs with incorporated outcome uncertainty stemming from an action performed by the robot and without incorporating the received observation. This is alongside the constraint over the posterior belief with the last observation included. To the best of the authors' knowledge, no previous works do that.
4) We also spot a problem happening in duality-based approaches arising from averaging unsafe actions in the MCTS phase. Therefore, an additional contribution of ours is an analysis of this phenomenon.
5) We simulate our findings on several continuous POMDP problems.

C. Notation

We use the $\square$ as a placeholder for various quantities. We also extensively use the indicator function notation, which is $\mathbf{1}_A(\square)$. This function equals one if and only if $\square \in A$. By lowercase letters, we denote the random variables of their realizations depending on context. By the boldface font, we denote vectors of operators in time of different lengths. We denote estimated values by $\hat{\square}$. We also stick to the widely used notation of the subsequent time index with the superscript prime $\square'$ to avoid the specification of the time instance.

D. Paper Roadmap

The rest of this article is organized as follows. Section II presents relevant background. Section III then formulates the problem. Section IV presents our approach. Section V-B summarizes our approach to actual algorithms. Section VI provides guarantees. In Section VII, we conduct complexity analysis, followed by the limitations and drawbacks in Section VIII. Next, Section IX covers our baseline. Section X gives experimental validation of the proposed methodology. Finally, Section XI concludes this article.

II. Background

This section gives the relevant background for our approach, namely, the belief-dependent POMDP, its reformulation to BMDP, and the MCTS on top of BMDP.

A. Belief-Dependent POMDP

The POMDP is a tuple $\langle \mathcal{X}, \mathcal{A}, \mathcal{Z}, \mathrm{T}, \mathrm{O}, \rho, \gamma, b_0 \rangle$, where $\mathcal{X}$, $\mathcal{A}$, and $\mathcal{Z}$ represent continuous state, action, and observation

spaces with $x\in\mathcal{X}$, $a\in\mathcal{A}$, and $z\in\mathcal{Z}$ being the individual state, action, and observation, respectively. $\mathrm{T}(x',a,x)\triangleq\mathbb{P}_{\mathrm{T}}(x'|x,a)$ is a stochastic transition model from the past state x to the subsequent x' through action a, $\mathrm{O}(z,x)\triangleq\mathbb{P}_{\mathrm{O}}(z|x)$ is the stochastic observation model. $\rho{:}\mathcal{B}\times\mathcal{A}\times\mathcal{Z}\times\mathcal{B}\mapsto\mathbb{R}$ is a belief-dependent reward incurred as a result of taking an action a from the belief b, receiving an observation z' and updating the belief to b'. By $\mathcal{B}$, we denote the space of all possible beliefs. $\gamma\in(0,1]$ is the discount factor, and b_0 is the prior belief. Purely for clarity of the exposition, we further assume that the reward depends solely on a pair of consecutive-in-time beliefs and an action in between. To remove unnecessary clutter, we assume that planning starts from b_0. Extension to the arbitrary planning time is straightforward. Let h_ℓ be a history (posterior history since it includes the last received observation). The history is the set that comprises the prior belief b_0, the actions $a_{0:\ell-1}$, and the observations $z_{1:\ell}$ that would be obtained by the agent up to time instance ℓ such that $h_\ell\triangleq\{b_0,a_{0:\ell-1},z_{1:\ell}\}$. We emphasize by the green color that b_0 is given, but the actions $a_{0:\ell-1}$ and observations $z_{1:\ell}$ can vary. Due to the assumption that the planning session starts from the prior belief b_0, we can have only the future history simulated in planning in this work. For completeness, we define $h_0\triangleq\{b_0\}$. The posterior belief b_ℓ is given by

$$b_\ell(x_\ell)\triangleq\mathbb{P}(x_\ell|b_0,a_{0:\ell-1},z_{1:\ell})=\mathbb{P}(x_\ell|h_\ell)=\mathbb{P}(x_\ell|b_\ell). \quad (1)$$

The belief is a function of history in this article (see Section VIII for more details) such that we sometimes write $b(h)$ instead of $b(x)$ and use the corresponding h notation to point to the belief $b(h)$. The actions within the history are coming from the execution policy. A deterministic policy π is a sequence of functions $\pi=\pi_{0:\ell-1}$ for $\ell\in[1\dots L-1]$, where the momentary function $\pi_i{:}\mathcal{B}\mapsto\mathcal{A}$ $\forall i$. At each time index, the policy maps belief to action. For better readability, sometimes we will omit the time index for policy or denote $\pi_{0:\ell-1}$ as π_{0+} and $\pi_{1:\ell-1}$ as π_{1+}. The policy can also be stochastic. In this case, it is a distribution of taking an action a_ℓ from a belief $\pi_\ell(a_\ell,b_\ell)=\pi_\ell(a_\ell,h_\ell)=\mathbb{P}^\pi_\ell(a_\ell|b_\ell(h_\ell))=\mathbb{P}^\pi_\ell(a_\ell|h_\ell)$.[2] Here, the action space $\mathcal{A}$ is the space of outcomes, and the mapping is $\pi_i{:}\mathcal{B}\times\mathcal{A}\mapsto\mathbb{R}$. We have that $\pi_{0:L-1}=\{\mathbb{P}^\pi_i\}_{i=0}^{L-1}$. Yet, in h_ℓ we have a specific realization of actions of such a policy in previous time instances. When the agent performs an action a and receives an observation z', it shall update its belief from b to b'. Let us denote the update operator by ψ such that $b'=\psi(b,a,z')$. In our context, it will be a particle filter (PF) since we focus on the setting of nonparametric beliefs. However, this is not an inherent limitation of our approach. Any belief update method would be suitable. We define a propagated belief b'^- as the belief b after the robot performed an action a and before it received and observation, namely

$$b^-_\ell(x_\ell)\triangleq\mathbb{P}(x_\ell|h_{\ell-1},a_{\ell-1})=\mathbb{P}(x_\ell|h^-_\ell)=\mathbb{P}(x_\ell|b^-_\ell). \quad (2)$$

[2]Here, the capability of history being switched with the belief has to be inspected for a particular belief update. In MCTS, as we will shortly see, the stochastic policy is history-dependent and can vary even if the belief is the same at different history nodes. In this article, the belief update is a PF. Thus, the probability of obtaining the same belief at different histories is zero.

We also define the propagated history as $h^-_\ell\triangleq h_\ell\backslash\{z_\ell\}=\{b_0,a_{0:\ell-1},z_{1:\ell-1}\}$. The unconstrained, online decision-making objective function is the action-value function specified as

$$Q^\pi(b_0,a_0;\boldsymbol{\rho}_1)\triangleq\gamma\mathbb{E}_{z_1}\left[\rho_1(b_0,a_0,b_1)+V^\pi(b_1;\boldsymbol{\rho}_2)\middle|b_0,a_0\right]. \quad (3)$$

Here, we added the subscript to the reward $\rho_{\square+1}(b_\square,b_{\square+1})$ to emphasize that it is a random variable and it is allowed not to specify dependence on consecutive-in-time beliefs and the action in between. The $V^\pi(b_\square;\boldsymbol{\rho}_{\square+1})$ is the value function under the stochastic policy π, and $\boldsymbol{\rho}_\ell$ is a vector of belief-dependent operators of appropriate length. The value function materializes as

$$V^\pi(b_0;\boldsymbol{\rho}_1)\triangleq\mathbb{E}\left[\sum_{\ell=0}^{L-1}\gamma^{\ell+1}\rho_{\ell+1}(b_\ell,a_\ell,b_{\ell+1})\middle|b_0,\pi\right]. \quad (4)$$

Let us present the following lemma to better understand the structure of (4) under a stochastic policy.

Lemma 1 (Representation of the Value Function): The value function under a stochastic execution policy complies with the following form:

$$\mathbb{E}\left[\sum_{\ell=0}^{L-1}\gamma^{\ell+1}\rho_{\ell+1}\middle|b_0,\pi\right]=\sum_{\ell=0}^{L-1}\gamma^{\ell+1}\mathbb{E}\left[\rho_{\ell+1}\middle|b_0,\pi\right]=$$
$$\sum_{\ell=0}^{L-1}\gamma^{\ell+1}\mathop{\mathbb{E}}_{a_0}\left[\mathop{\mathbb{E}}_{b_1}\left[\mathop{\mathbb{E}}_{a_1}\left[\mathop{\mathbb{E}}_{b_2}[\dots\right.\right.\right.$$
$$\left.\left.\left.\mathop{\mathbb{E}}_{a_\ell}\left[\mathbb{E}\left[\rho_{\ell+1}|b_\ell,a_\ell\right]\middle|b_\ell,\pi_\ell\right]\dots\middle|b_1,a_1\right]\middle|b_1,\pi_1\right]\middle|b_0,a_0\right]\middle|b_0,\pi_0\right]. \quad (5)$$

We laid out the detailed proof in Section 2.1 of the Supplementary Material.[3] In online decision-making, the future belief tree policy π_{1+} is approximated as part of the decision process. We denote the best future policy as $\pi^*_{(k+1)+}$. The best deterministic policy for the present time is given by $\pi_0(b_0)=\arg\max_{a_0\in\mathcal{A}}Q^{\pi^*_{1+}}(b_0,a_0;\boldsymbol{\rho}_1)$. The best stochastic policy is the solution of the $\max_{\pi_\ell}\mathbb{E}_{a_\ell\sim\mathbb{P}^\pi_\ell(a_\ell|b_\ell)}[Q^{\pi^*_{(\ell+1)+}}(b_\ell,a_\ell;\boldsymbol{\rho}_{\ell+1})]$. The interlinks between (3) and (4) are $V^\pi(b_\ell;\boldsymbol{\rho}_{\ell+1})=Q^{\pi_{(\ell+1)+}}(b_\ell,\pi_\ell(b_\ell);\boldsymbol{\rho}_{\ell+1})$, in the case of deterministic policies, and $V^\pi(b_\ell;\boldsymbol{\rho}_{\ell+1})=\mathbb{E}_{a_\ell\sim\mathbb{P}^\pi_\ell(a_\ell|b_\ell)}[Q^\pi(b_\ell,a_\ell;\boldsymbol{\rho}_{\ell+1})]$, in the case of the stochastic policies.

In Section II-C, we will see why in time zero we have a deterministic policy and in future time, the policy is stochastic.

B. PF Belief State MDP

To employ solvers crafted for fully observable MDPs, one can cast POMDP as a BMDP. The BMDP is the following tuple $\langle\mathcal{B},\mathcal{A},\mathrm{T}_b,\rho,\gamma,b_0\rangle$, where $\mathcal{B}$ is the space of all possible beliefs defined by (1). The belief state transition model is

$$\mathrm{T}_b(b,a,b')\triangleq\mathbb{P}_{\mathrm{T}_b}(b'|b,a)=\int_{z'\in\mathcal{Z}}\mathbb{P}(b'|b,a,z')\mathbb{P}(z'|b,a)\mathrm{d}z'. \quad (6)$$

[3]Find the Supplementary Material here https://tinyurl.com/4cbtffxc.

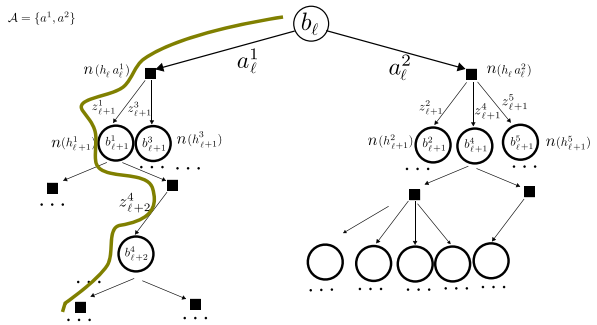


Fig. 1. Here, we plot the asymmetric search tree approximating stochastic future policy. For simplicity, the action space here is $\mathcal{A}=\{a^1, a^2\}$. We behold that many actions emanate from each belief node, and each action has a weight defined by the relevant visitation count as in (8). Thus, the BMCTS-DPW approximates stochastic future policy. Note that here the observations and beliefs have a global index (superscript) while actions have a local index according to the action number in the space $\mathcal{A}$.

C. Continuous Belief MCTS

We now describe the anytime SOTA approach to approximately solve unconstrained continuous POMDP online. This technique operates on the level of BMDP to accommodate belief-dependent rewards. We refer to this type of MCTS as Belief-MCTS with double progressive widening (BMCTS-DPW). BMCTS-DPW estimates (3) and (4) online, leaning on sampling and constructing the search tree. Further, we shorten the notation and mark $\hat{V}^{\pi^*}(b;\boldsymbol{\rho}')$ by $\hat{V}^*(h)$ and $\hat{Q}^{\pi^*}(ba;\boldsymbol{\rho}')$ by $\hat{Q}(ha)$. BMCTS-DPW constructs the search tree comprised of sampled belief nodes (transparent circles) and belief-action nodes (black squares) by iteratively descending down the tree and ascending back to the root (see Figs. 1 and 2). The BMCTS-DPW algorithm descends down the tree, cycling through the BMDP loop [see Fig. 3(a)] with subsequent in-time beliefs. Recall that our belief update ψ is PF. In case the beliefs are represented by particles, BMCTS-DPW is called a PF tree with DPW (PFT-DPW) [19]. On the way down the tree, the (polynomial [18]) DPW manages the sampling of new actions and beliefs/observations. The exploration rule selects one of the already sampled actions. If no new belief is sampled, then one of the already bookkept beliefs is sampled uniformly or selected deterministically according to minimal visitation count. On the way back to the root, BMCTS-DPW updates action-value estimates at each belief action node [see Fig. 2(a)] and relevant visitation counts. The DPW technique enables gradually expanding new actions and beliefs as the tree search progresses. Moreover, MCTS without progressive widening of actions necessitates discrete action space or sampling a finite set of actions at once.

1) Dependence of the Estimates on History: We, with a slight abuse of notation, will sometimes switch the dependence of various quantities on belief $b(h)$ and dependence on the corresponding history h. This is because, in general, the same belief can correspond to different histories. Even though in the case of PF, as we mentioned in Section II-A (with regard to policy), it cannot happen; to properly mark the position at the search tree and support general belief updates, e.g., parametric belief update, we shall use history h instead of belief $b(h)$.

2) Search: The exploration score is defined as

$$\mathrm{sc}(h,a) \triangleq \underbrace{\hat{Q}(h,a)}_{\text{belief action node } ba \text{ indexed by } ha} + \kappa\sqrt{f(n(h))/n(ha)} \quad (7)$$

and it governs the selection of the (sampled or discrete) actions down the tree, where $n(h)$ is the visitation count of the belief nodes, $n(ha)$ is the visitation count of belief-action nodes, and κ is the exploration constant [see Fig. 2(b)]. The notation ha is the history h with action a appended to the end, alias to h^- with action a explicitly seen. The function f is $\log$ in the case of upper confidence bound (UCB) [20] exploration and power in the case of Polynomial Upper Confidence Tree (PUCT) [18].

The BMCTS-DPW can be run with rollout and without rollout. In the case of a rollout configuration, from each new belief node, the rollout is initiated to provide an initial $\hat{V}^*$ of the newly added belief node. This is not mandatory since if no rollout is initiated, BMCTS-DPW will continue to descend down the tree until the deepest level with the first action from the action space $\mathcal{A}$ (first sampled action in case of continuous action space). If rollout is ON, then DPW solves the problem of shallow trees in a continuous setting. This problem arises because in this setting, it is impossible to sample the same action and observation twice. In case of no rollout, DPW is needed to improve the $\hat{Q}$ estimates down the tree. Without DPW, BMCTS will always sample new beliefs. After a single step ahead, such a BMCTS will always descend with the newly sampled $\mathcal{A}$ action (or first action from possibly shuffled $\mathcal{A}$) and new observation, constructing the tree only from rollouts.

In addition, it shall be noted that *not in every tree query* will the BMCTS-DPW expand a new node. In some queries, only visitation counts are promoted (lace already present in the tree incorporated to pertinent $\hat{Q}$). In continuous spaces, it happens because of DPW. DPW and increasing the visitation counts without adding a new lace introduce a shift in the beliefs and observations distributions, $\mathbb{P}(b_{1:L}|b_0,\pi)$ and $\mathbb{P}(z_{1:L}|b_0,\pi)$, respectively, when stopped in finite time. This is out of the scope of this article.

The $\hat{Q}(b(h),a)$ estimates are assembled from the laces (yellow curve in Fig. 1) of the cumulative rewards calculated over the beliefs along the simulated histories. Imagine that at the depth ℓ of the belief tree, each belief has a *global in tree* index i_ℓ per depth ℓ, say the index runs from left to right over all the belief nodes at level ℓ. Let us define the set of global indices of posterior beliefs that are children of $b_\ell^{i_\ell}(h_\ell^{i_\ell})$ and action a_ℓ by $C(h_\ell^{i_\ell}a_\ell)$. We also define the set of actions emanating from $b_\ell^{i_\ell}$ by $C(h_\ell^{i_\ell})$. Only at time zero do we make these sets and visitation counts depend on belief instead of history. In the next equation, we omit the subscript denoting the time instance of histories, beliefs, and actions. Suppose BMCTS-DPW is configured to run without rollout. In this case, $\hat{Q}(h_\ell^{i_\ell},a_\ell)$ reads

$$\hat{Q}(h^{i_\ell},a) = \gamma \overbrace{\sum_{i_{\ell+1}\in C(h^{i_\ell}a)} \frac{n(h^{\prime,i_{\ell+1}})}{n(h^{i_\ell}a)}}^{\text{single immediate action}} \left(\rho_{\ell+1}(b^{i_\ell},a,b^{\prime,i_{\ell+1}}) +\right.$$

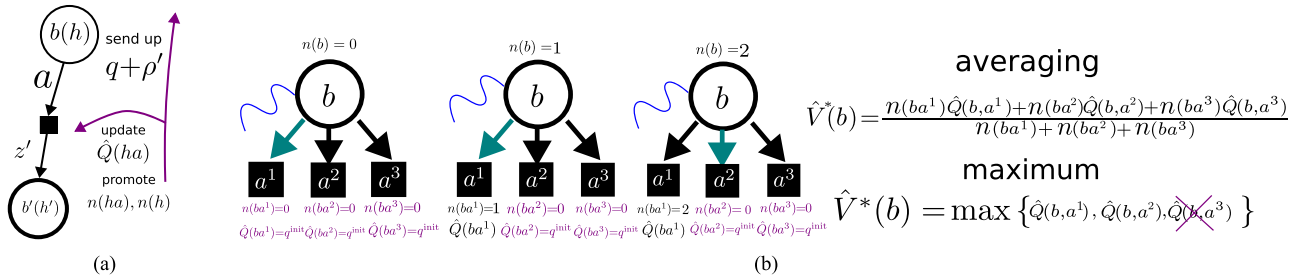


Fig. 2. (a) Visualization of the BMCTS-DPW operations when ascending up the search tree. We update $\hat{Q}(ha)$, visitation counts $n(ha)$, and $n(h)$, send up the lace q of the cumulative reward. (b) Illustration of the BMCTS-DPW operation with discrete action space comprised by three actions $\mathcal{A} \triangleq \{a^1, a^2, a^3\}$. First, upon reaching a leaf node, the current action space is unfolded to belief-action nodes. BMCTS-DPW then selects each action infinitely often (only if the $\mathcal{A}$ is discrete) and descends down the tree. At the way up the belief tree the classical BMCTS-DPW takes the average of the actions tried so far (after relevant updates on the way up) to update the estimator of (3). In this illustration, a^3 still did not tried and therefore do not participate. On the way up $n(ha^3)$ stays zero.

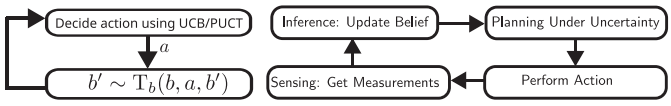


Fig. 3. (a) BMDP loop within planning session. (b) AL.

$$\gamma \underbrace{\left(\sum_{a' \in C(h'^{,i_{\ell+1}})} \frac{n(h'^{,i_{\ell+1}}a')}{n(h'^{,i_{\ell+1}})} \hat{Q}(h'^{,i_{\ell+1}}, a') \right)}_{\hat{V}^{\pi^*}(h'^{,i_{\ell+1}})} \tag{8}$$

(different actions due to (7) approximating the best exploratory future tree policy π^*)

The future policy highlighted by the magenta color is tree query-dependent (see Fig. 1). In the same manner, the sets $C(h^{i_\ell}a)$ and $C(h'^{,i_{\ell+1}})$ implicitly depend on the tree query number. One of our crucial insights in this article is the summation over the actions in (8) marked by the red color. This average can also be perceived as a stochastic policy. Crucially, anytime this summation can include *unsafe* actions in an unconstrained BMCTS-DPW approach. As we will further see, our approach does not suffer from this problem. We clean dangerous actions from the search tree.

III. Problem Formulation and Rationale

We now proceed to our theoretical problem formulation. To reduce clutter, we assume that the planning time index is zero. This is not an inherent limitation of our approach; every further relation can be easily modified to accommodate the general planning time index. We endow the BMDP described in Section II-B with a belief-dependent payoff operator ϕ and obtain $\langle \mathcal{B}, \mathcal{A}, \mathrm{T}_b, \rho, \phi, \gamma, b_0 \rangle$.

A. Problem Formulation

Our aim is to tackle the problem presented in [3] and [4] narrowed to the multiplicative form of the inner constraint and considering a stochastic future policy. In [3] and [4], we presented our probabilistic constraint (PC) defined as such: $\mathrm{P}(c{=}1|b_0, a_0, \pi){=}1$, where c is a Bernoulli random variable. In this work, c maps to one of the event $b_{0:L} \in \bigcap_{\ell=0}^{L} \Phi_\ell^\delta$ such that the problem we want to solve is

$$a_0^* \in \arg\max_{a_0 \in \mathcal{A}} Q^{\pi^*}(b_0, a_0; \boldsymbol{\rho}_1) \text{ subject to} \tag{9}$$

$$\underbrace{\mathrm{P}\left(b_{0:L} \in \bigcap_{\ell=0}^{L} \Phi_\ell^\delta \middle| b_0, a_0, \pi^*_{1:L-1} \right) = 1}_{\text{outer constraint}} . \tag{10}$$

In this article, we define the following sets as said $\Phi_0^\delta \triangleq \{b_0 : \phi(b_0) \geq \delta\}$, and for $\ell \in [1{:}L]$, the relevant set is:

$$\Phi_\ell^\delta \triangleq \left\{ b_\ell^-, b_\ell : b_\ell^- \in \mathcal{B}_\ell^-, b_\ell \in \mathcal{B}_\ell, \phi(b_\ell^-) \geq \delta, \phi(b_\ell) \geq \delta \right\} . \tag{11}$$

One example of an operator ϕ is the probability to be safe given belief, specified as

$$\phi(b_\ell) = \mathrm{P}\left(\{x_\ell \in \mathcal{X}_\ell^{\text{safe}}\} \middle| b_\ell \right) = \mathbb{E}_{x_\ell \sim b_\ell}[\mathbf{1}_{\{x_\ell \in \mathcal{X}_\ell^{\text{safe}}\}}] \tag{12}$$

$$\phi(b_\ell^-) = \mathrm{P}\left(\{x_\ell \in \mathcal{X}_\ell^{\text{safe}}\} \middle| b_\ell^- \right) = \mathrm{P}\left(\{x_\ell \in \mathcal{X}_\ell^{\text{safe}}\} \middle| h_\ell^- \right) . \tag{13}$$

Here, $\mathcal{X}^{\text{safe}}$ is the safe space, e.g., the space where a robot can move without inflicting damage on itself. Therefore, we can think about the event $\bigcap_{\ell=0}^{L} \Phi_\ell^\delta$ as the *safe belief space*.

$\mathcal{B}_\ell^-$ and $\mathcal{B}_\ell$ in (11) are the reachable spaces in time ℓ of propagated beliefs b_ℓ^- and posterior beliefs b_ℓ, respectively, from a given b_0. By the green color in (11), we highlight that we constrain the propagated beliefs in addition to the posteriors.

The probability of the event $\bigcap_{\ell=0}^{L} \Phi_\ell^\delta$ equals the probability of the event $(\mathbf{1}_{\Phi_0^\delta}(b_0) \prod_{\ell=1}^{L} \mathbf{1}_{\Phi_\ell^\delta}(b_\ell^-, b_\ell)){=}1$. In this work, although we use PF as the belief update ψ, we do not take into account the stochasticity of the belief update operator as opposed to [21] and [22] and treat the ψ operator as deterministic. Since it would significantly complicate this article, we leave this aspect to future work.

One can extract the propagated belief from the belief update ψ, namely

$$b' = \psi(b, a, z') \triangleq \psi^{\text{post}}(\psi^{\text{prop}}(b, a), z') = \psi^{\text{post}}(b'^{,-}, z'). \tag{14}$$

where $\psi^{\text{prop}}(b, a){=}b'^{,-}$. Thus, to make the exposition clearer, from now on, the indicator $\mathbf{1}_{\Phi_\ell^\delta}(b_\ell)$ depends solely on the posterior b_ℓ and not on both the posterior b_ℓ and the propagated

belief b_ℓ^-. In algorithms, for the sake of clarity, we make the indicators dependent on both beliefs, propagated and posterior.

The $\pi^*_{1:L-1}$ is the best future exploratory stochastic policy approximated by our probabilistically constrained BMCTS-DPW, as we will further see. The approximation of the best future tree policy improves over time, as proved by Auger et al. [18] for an unconstrained problem. In our problem, instead of the best future stochastic tree policy, we have the best future stochastic probabilistically constrained policy. This is because our PC is automatically enforced in future times due to its recursive nature, as we will see in Section III-C. From the discussion above and indicator properties, (10) is equal to

$$\mathrm{P}\Big(\underbrace{\big(\mathbf{1}_{\Phi_0^\delta}(b_0)\prod_{\ell=1}^{L}\mathbf{1}_{\Phi_\ell^\delta}(b_\ell)\big)=1}_{\substack{\text{inner}\\ \text{constraint}}}|b_0,a_0,\pi\Big)= \mathbb{E}[\mathbf{1}_{\Phi_0^\delta}(b_0)\prod_{\ell=1}^{L}\mathbf{1}_{\Phi_\ell^\delta}(b_\ell)|b_0,a_0,\pi]. \tag{15}$$

The outer condition (10) coupled with the inner condition outlined by (15) says that with probability one (almost surely), future propagated and posterior beliefs b^- and b, L steps ahead, will satisfy $\phi(b^-)\geq\delta$ and $\phi(b)\geq\delta$ correspondingly.

Constraining the propagated belief (13) means constraining on average (theoretical expectation) the posterior as discussed in the next section.

B. Implications of Constraining Propagated Belief

In this section, we shed light on the question: what does it mean to constrain the propagated beliefs alongside posterior beliefs? To cancel the constraining of the propagated beliefs, one must redefine the set Φ_ℓ^δ for every ℓ as follows: $\Phi_\ell^\delta\triangleq\{b_\ell^-, b_\ell:\cancel{b_\ell^-\in\mathcal{B}_\ell^-}, b_\ell\in\mathcal{B}_\ell, \cancel{\phi(b_\ell^-)\geq\delta}, \phi(b_\ell)\geq\delta\}$. Further in the article, all the developments are valid for both versions of the set Φ_ℓ^δ. The probability to be safe given a propagated belief equals

$$\mathrm{P}\left(\{x_\ell\in\mathcal{X}_\ell^{\text{safe}}\}\big|b_\ell^-\right)=\mathrm{P}\left(\{x_\ell\in\mathcal{X}_\ell^{\text{safe}}\}\big|h_\ell^-\right)= \int_{z_\ell\in\mathcal{Z}}\mathrm{P}\left(\{x_\ell\in\mathcal{X}_\ell^{\text{safe}}\}\big|h_\ell^-,z_\ell\right)\mathbb{P}(z_\ell|h_\ell^-)\mathrm{d}z_\ell= \mathbb{E}_{z_\ell}[\mathrm{P}\left(\{x_\ell\in\mathcal{X}_\ell^{\text{safe}}\}\big|h_\ell^-,z_\ell\right)|h_\ell^-]=\mathbb{E}_{z_\ell}[\mathrm{P}\left(\{x_\ell\in\mathcal{X}_\ell^{\text{safe}}\}\big|b_\ell\right)|b_\ell^-]. \tag{16}$$

The theoretical expectation in (16) is out of reach. Yet, we evaluate it using the propagated belief $b^-(h^-)$. Defining the set Φ_ℓ^δ as (11), with the propagated beliefs, allows us to account for all the possible posterior beliefs in (16). Since the sample mean converges in probability, it holds that $\forall\epsilon>0$

$$\lim_{|C(h_\ell^-)|\to\infty}\mathrm{P}\left(\Big|\mathrm{P}\left(\{x_\ell\in\mathcal{X}_\ell^{\text{safe}}\}\big|h_\ell^-\right) - {}^1\!/_{|C(h_\ell^-)|}\sum_{z_\ell^l\in C(h_\ell^-)}\mathrm{P}\left(\{x_\ell\in\mathcal{X}_\ell^{\text{safe}}\}\big|h_\ell^-,z_\ell^l\right)|>\epsilon\Big|h_\ell^-\right)=0. \tag{17}$$

With a slight abuse of notation, $C(h_\ell^-)$ is now a list of the enumerated observations that are children of h_ℓ^-. Equation (17) means that for any arbitrary small error ϵ, the difference between (16) and its approximation by the children of h_ℓ^- tends to zero as the number of children of h_ℓ^- grows. Our method constrains *anytime* both members of (17).

Theorem 1 (Necessary condition for entire observation space $\mathcal{Z}$ of children of h_ℓ^- to be safe): Fix $\delta\in[0,1]$ and assume that

$$\mathrm{P}\left(\{x_\ell\in\mathcal{X}_\ell^{\text{safe}}\}\big|h_\ell^-\right)\geq\delta. \tag{18}$$

Equation (18) is a necessary condition for the entire observation space $\mathcal{Z}$ of children of h_ℓ^- to be safe. To rephrase that $\mathrm{P}(\{x_\ell\in\mathcal{X}_\ell^{\text{safe}}\}\big|h_\ell^-)<\delta$ implies that $\exists b_\ell(h_\ell)$ a child of h_ℓ^-, which is not safe, namely, $\mathrm{P}(\{x_\ell\in\mathcal{X}_\ell^{\text{safe}}\}\big|h_\ell^-,z_\ell)<\delta$.

See Section 2 of the Supplementary Material for the detailed proof. We still need to check the children posteriors $\{z_\ell^l\}_{l=1}^{|C(h_\ell^-)|}$. This is because condition (18) is only necessary and not sufficient. If for all the children $\forall z_\ell\in\mathcal{Z}$ of h_ℓ^-, it holds that $\mathrm{P}(\{x_\ell\in\mathcal{X}_\ell^{\text{safe}}\}\big|h_\ell^-,z_\ell)\geq\delta$, it has to be that $\mathrm{P}(\{x_\ell\in\mathcal{X}_\ell^{\text{safe}}\}\big|h_\ell^-)\geq\delta$. Since the condition is not sufficient, we cannot say that $\mathrm{P}(\{x_\ell\in\mathcal{X}_\ell^{\text{safe}}\}\big|h_\ell^-)<\delta$ implies that $\forall b_\ell(h_\ell)$ that are children of h_ℓ^-, it will hold that $\mathrm{P}(\{x_\ell\in\mathcal{X}_\ell^{\text{safe}}\}\big|h_\ell)<\delta$. Note that if for every sampled observation $\mathrm{P}(\{x_\ell\in\mathcal{X}_\ell^{\text{safe}}\}\big|h_\ell^-,z_\ell^l)\geq\delta$, it implies that

$$\left({}^1\!/_{|C(h_\ell^-)|}\sum_{l=1}^{|C(h_\ell^-)|}\mathrm{P}\left(\{x_\ell\in\mathcal{X}_\ell^{\text{safe}}\}\big|h_\ell^-,z_\ell^l\right)\right)\geq\delta. \tag{19}$$

To conclude, by constraining the propagated belief, we constrain the theoretical expectation of the posteriors given h_ℓ^-, and by constraining each posterior, we also constrain its sample approximation portrayed by (19). Without constraining the propagated belief, if the number of children of $b_\ell^-(h_\ell^-)$ is small, namely, $|C(h_\ell^-)|$ is small, then we anticipate poor robot's safety in the execution of the best action found by our planner (e.g., number of collisions). This will happen if the number of BMCTS-DPW tree queries is small. This is because constraining the propagated belief allows us to account in expectation for all the observations in the observation space, and not only the sampled observations.

Remark: To ensure feasibility of our PC (10) at the limit of BMCTS-DPW convergence, the robot has to have a bounded support of the belief b_0 and bounded motion models. If we deal with a particle-based representation of b_0, we perceive the particles as true robot positions, so it is left only to assure that the motion model is bounded. This is, however, natural since the robot cannot have limitless actuators. We delve into this aspect in Section IV-B.

C. Recursive Form of PC

Our constraint depends on a stochastic policy. Similar to the objective (5) adhering to recursive form (see Lemma 1) in our PC, we land at the following result.

Theorem 2 (Representation of PC, recursive form): The PC defined by (15) conforms to the following recursive form:

$$\mathbf{1}_{\Phi_0^\delta}(b_0)\mathbb{E}_{b_1}[\mathbf{1}_{\Phi_1^\delta}\mathbb{E}_{a_1}[\mathbb{E}_{b_2}[\mathbf{1}_{\Phi_2^\delta}\dots$$
$$\mathbb{E}[\mathbf{1}_{\Phi_L^\delta}|b_{L-1},a_{L-1}]\dots|b_1,a_1]|b_1,\pi_1,|b_0,a_0] =$$
$$\mathbf{1}_{\Phi_0^\delta}(b_0)\mathbb{E}_{b_1}[\mathop{\mathbb{E}}_{a_1\sim\mathbb{P}_1^\pi(a_1|b_1)}[$$
$$\mathrm{P}\big(\big(\prod_{\ell=1}^{L}\mathbf{1}_{\Phi_\ell^\delta}(b_\ell)\big)=1|b_1,a_1,\pi\big)|b_1,\pi_1]|b_0,a_0]. \quad (20)$$

We provide the detailed proof in Section 2.3 of the Supplementary Material. The recursive form displays an important fact. In future planning sessions simulated in the current planning session, our formulation takes into account that the robot will enforce the safety constraint when it plans in the future. This is very similar to the recursive cumulative expected constraint shown in [10]. However, our recursion is multiplicative. Further, in Section VI-C, we show rigorously that future dangerous actions participating in the objective can shift the expectations and change the action returned by the planner.

IV. Anytime Approach

In this section, we present our anytime safety approach. To invalidate the sample approximation of (10), it is sufficient that a single belief (propagated or posterior) in the belief tree fails to be safe and the corresponding indicator is zero. In our methodology, we leverage the classical iterative BMCTS-DPW scheme of descending down the search tree of histories and ascending back to the root (see Section II-C). Once on the way down the tree an unsafe belief is encountered, we know that the PC enforced from each predecessor belief node is violated. We delete such an action from the search tree and fix the relevant $\hat{Q}$ and the visitation counts above in the search tree. Let us delve into the details.

Suppose the BMCTS-DPW is configured to run without rollout. Would we construct the estimated counterpart of (20) from the belief tree constructed by BMCTS-DPW, the estimator of our PC would be such that

$$\Bigg(\mathbf{1}_{\Phi_0^\delta}(b_0)\sum_{i_1\in C(b_0a_0)}\frac{\mathbf{1}_{\Phi_1^\delta}(b_1^{i_1})}{|C(b_0a_0)|}$$
$$\sum_{a_1\in C(h_1^{i_1})}\frac{n(h_1^{i_1}a_1)}{n(h_1^{i_1})}\sum_{i_2\in C(h_1^{i_1}a_1)}\frac{\mathbf{1}_{\Phi_2^\delta}(b_2^{i_2})}{|C(h_1^{i_1}a_1)|}\dots$$
$$\dots\frac{\mathbf{1}_{\Phi_{L-1}^\delta}(b_{L-1}^{i_{L-1}})}{|C(h_{L-2}^{i_{L-2}}a_{L-2})|}\sum_{a_{L-1}\in C(h_{L-1}^{i_{L-1}})}\frac{n(h_{L-1}^{i_{L-1}}a_{L-1})}{n(h_{L-1}^{i_{L-1}})}$$
$$\sum_{i_L\in C(h_{L-1}^{i_{L-1}}a_{L-1})}\frac{\mathbf{1}_{\Phi_L^\delta}(b_L^{i_L})}{|C(h_{L-1}^{i_{L-1}}a_{L-1})|}\Bigg)=1. \quad (21)$$

Since our constraint is defined using an indicator, (21) translates to the fact that *each* lace defined by the actions and the observations on the way down the tree shall consist *only* of safe beliefs. Note that (21) approximates the condition (20) equals one and is enforced from each belief in the search tree.

To emphasize that each belief in the search tree has a single parent and the corresponding parent is attainable, let us introduce yet another notation $b_\ell^{i_\ell|i_{\ell-1}}$. This means that the belief $b_\ell^{i_\ell|i_{\ell-1}}$ has a global index i_ℓ and the parent belief has a global index $i_{\ell-1}$. On the way down the tree, we ensure that

$$\Big(\mathbf{1}_{\Phi_0^\delta}(b_0^1)\mathbf{1}_{\Phi_1^\delta}(b_1^{i_1|1})\mathbf{1}_{\Phi_2^\delta}(b_2^{i_2|i_1})\dots$$
$$\mathbf{1}_{\Phi_{L-1}^\delta}(b_{L-1}^{i_{L-1}|i_{L-2}})\mathbf{1}_{\Phi_L^\delta}(b_L^{i_L|i_{L-1}})\Big)=1 \quad (22)$$

where the actions along the lace are $a_1\in C(h_1^{i_1}), a_2\in C(h_2^{i_2}),\dots,a_{L-1}\in C(h_{L-1}^{i_{L-1}})$, and the beliefs are according to the observations indexed by $i_2\in C(h_1^{i_1}a_1), i_3\in C(h_2^{i_2}a_2),\dots,i_L\in C(h_{L-1}^{i_{L-1}}a_{L-1})$. In other words, we require that every propagated and posterior belief along the lace be safe. Condition (22) can be verified on the way down the tree as the search expands the lace. To do that, the algorithm must check each indicator while descending without requiring a look ahead.

Remark: The equivalence of (21) and the fact that every lace in the search tree shall be safe is a property of our PC formulation, e.g., this does not happen in the case of the popular chance constraint [15].

Note that in (21), we do not have a distributional shift due to PW of observations (and the fact that not in every tree query, a new belief node is introduced), as opposed to the objective (8), as explained in Section II-C2. This is because we do not take into account the statistics dictated by the visitation counts of the observations. Instead of explicitly calculating (21), we merely require that each indicator along each lace is one.

As we see from (21), the recursive form portrayed by (20) transfers to the BMCTS-DPW estimator. Let us denote the product of the indicators in the inner constraint (15) by c depending on the current and future beliefs. For example, at the root of the belief tree, we have $c(b_{0:L})=\mathbf{1}_{\Phi_0^\delta}(b_0)\prod_{\ell=1}^{L}\mathbf{1}_{\Phi_\ell^\delta}(b_\ell)$. By design, (20) and (21) are equal to one if and only if the PC starting from each belief action node ha in the tree is satisfied, namely, $\mathrm{P}(c=1|b(h),a,\pi)=1$. We now define the notion of a dangerous action in belief tree.

Remark: We call an action dangerous if it is *believed* to be dangerous. Meaning our notion of dangerous or safe actions is based on beliefs and not the possible POMDP states as in chance constraint [2].

Definition 1 (Dangerous action): A dangerous action is action a in a place h in a search tree that renders an estimator of (20) smaller than one, namely, $\hat{\mathrm{P}}(c=1|b_0,a_0,\pi)<1$, where the estimator is as in (21).

Note that the best stochastic future tree policy is dependent on the number of performed tree queries.

Corollary 1: Each action in a search tree can be dangerous or safe. We define *safe* action a (or a_0) to be the action that is *not* dangerous, namely, $\hat{\mathrm{P}}(c=1|b_0,a_0,\pi)=1$, and safe under the safe future tree policy, namely, $\hat{\mathrm{P}}(c=1|b(h),a,\pi)=1$ for arbitrary future history h as a result of the mentioned safe policy.

Let us reiterate that we build the search tree solely from the safe actions. Effectively, using our pruning and fixing the values and statistics maintained by the search, to be explained shortly, we assure preemptively that the sample approximation of (20)

defined by (21) using the beliefs from the search tree built by BMCTS-DPW equals one. To assure that (21) equals one, it is required that every indicator function within is one. This is our mechanism to assure that in *any* finite time, the search tree contains only *safe* actions, as opposed to duality-based methods, where the constraint is satisfied only at the convergence limit, namely, in infinite time (see Section IX-B). When a newly sampled belief renders the corresponding indicator equal to one, we add it to the belief tree. If the indicator is zero, then we develop a mechanism to delete an action and fix the search tree upward.

A. Pushing Forward in Time Only the Safe Trajectories

Even if (21) equals one, meaning every indicator inside equals one, when $\delta<1$ and the payoff operator as in (12), it is possible that there exist samples that are unsafe, e.g., falling inside an obstacle or a dangerous region. If the robot is operational, then it means the robot was safe before it commenced an action. Thus, we shall discard the unsafe portion of the belief before we update the belief with action and observation [barring the situation when $\delta=1$ and the payoff operator as in (12) and (13)]. We define $\bar{b}^{\text{safe}}$ as the belief constituted only by the safe particles, namely, conditioned on the history and the intersection of the events $\{x_i\in\mathcal{X}_i^{\text{safe}}\}$ for time indices $i=1\ldots\ell$. Such a belief is given by

$$\bar{b}_\ell^{\text{safe}}(x_\ell)=\mathbb{P}\left(x_\ell|b_0,a_{0:\ell-1},z_{1:\ell},\bigcap_{i=0}^{\ell}\{x_i\in\mathcal{X}_i^{\text{safe}}\}\right). \tag{23}$$

To convert $\bar{b}$ to $\bar{b}^{\text{safe}}$, we remove not safe particles and re-sample with replacement the safe ones to the initial size. This means that the beliefs and observations in (20) will not be as in objective (9) but as follows. We define $\bar{b}$ as the belief obtained by percolating forward in time belief that has been made safe sequentially, that is, similar to (14) $\bar{b}'=\psi(\bar{b}^{\text{safe}},a,z')=\psi^{\text{post}}(\psi^{\text{prop}}(\bar{b}^{\text{safe}},a),z')=\psi^{\text{post}}(\bar{b}'^{,-},z')$, where

$$\bar{b}_\ell(x_\ell)=\mathbb{P}\left(x_\ell|b_0,a_{0:\ell-1},z_{1:\ell},\bigcap_{i=0}^{\ell-1}\{x_i\in\mathcal{X}_i^{\text{safe}}\}\right) \tag{24}$$

and the belief propagated only with action and without an observation

$$\bar{b}_\ell^-(x_\ell)=\mathbb{P}\left(x_\ell|b_0,a_{0:\ell-1},z_{1:\ell-1},\bigcap_{i=0}^{\ell-1}\{x_i\in\mathcal{X}_i^{\text{safe}}\}\right). \tag{25}$$

Both beliefs are generally unsafe. Our BMDP tuple is now augmented with another space of beliefs $\bar{\mathcal{B}}$ defined by (24). We have now

$$\langle\mathcal{B},\underbrace{\bar{\mathcal{B}}}_{\text{space of the beliefs defined by (24)}},\mathcal{A},\mathrm{T}_b,\underbrace{\rho}_{\text{reward}},\underbrace{\phi}_{\text{payoff}},\gamma,b_0\rangle.$$

At this point, we need to define another safe set

$$\bar{\Phi}_\ell^\delta=\{\bar{b}_\ell^-,\bar{b}_\ell:\bar{b}_\ell^-\in\bar{\mathcal{B}}_\ell^-,\bar{b}_\ell\in\bar{\mathcal{B}}_\ell,\phi(\bar{b}_\ell^-)\geq\delta,\phi(\bar{b}_\ell)\geq\delta\}.$$

Here, the $\bar{\mathcal{B}}_\ell^-$ and $\bar{\mathcal{B}}_\ell$ are reachable from b_0 spaces of beliefs defined by (24) and (25). Only at time 0, the set $\Phi_0^\delta=\bar{\Phi}_0^\delta$. This is because in inference we know that the robot is still operational. We always make safe the actual robot belief b_0. In planning, the belief is rendering the observation in the next time step [see Fig. 4(a)]. Thus, in both CC and PC in time $\ell+1$, the observation PDF reads $\mathbb{P}(z_{\ell+1}|\bar{b}_\ell^{\text{safe}},a_\ell)$, whereas in the objective, we have $\mathbb{P}(z_{\ell+1}|b_\ell,a_\ell)$. We sample from the latter, and the normalized ratios of these likelihoods $w_{\ell+1}^{a_\ell}\propto{}^{\mathbb{P}(z_{\ell+1}|\bar{b}_\ell^{\text{safe}},a_\ell)}/_{\mathbb{P}(z_{\ell+1}|b_\ell,a_\ell)}$ are the weights in (27). Using importance sampling in such a way, we construct a single belief tree. However, for the constraint calculation, we use $\bar{b}$ corresponding to the belief b; see Fig. 4(a). Equation (15) transforms into

$$\mathrm{P}\left(\left(\mathbf{1}_{\Phi_0^\delta}(b_0)\prod_{\ell=1}^{L}\mathbf{1}_{\bar{\Phi}_\ell^\delta}(\bar{b}_\ell)\right)=1|b_0,a_0,\pi\right). \tag{26}$$

Let us reiterate, on the way down the tree, we ensure that every belief along the lace lightens up its indicator. Similar to (21), we ensure that under the stochastic policy approximated by the MCTS, the PC is satisfied. We have that

$$\left(\mathbf{1}_{\bar{\Phi}_0^\delta}(b_0)\sum_{i_1\in C(b_0a_0)}w_1^{a_0,i_1}\mathbf{1}_{\bar{\Phi}_1^\delta}(\bar{b}_1^{i_1})\sum_{a_1\in C(h_1^{i_1})}\frac{n(h_1^{i_1}a_1)}{n(h_1^{i_1})}\right.$$
$$\sum_{i_2\in C(h_1^{i_1}a_1)}w_2^{a_1,i_2}\mathbf{1}_{\bar{\Phi}_2^\delta}(\bar{b}_2^{i_2})\cdots$$
$$\cdots\mathbf{1}_{\bar{\Phi}_{L-1}^\delta}(\bar{b}_{L-1}^{i_{L-1}})\sum_{a_{L-1}\in C(h_{L-1}^{i_{L-1}})}\frac{n(h_{L-1}^{i_{L-1}}a_{L-1})}{n(h_{L-1}^{i_{L-1}})}$$
$$\left.\sum_{i_L\in C(h_{L-1}^{i_{L-1}}a_{L-1})}w_L^{a_{L-1},i_L}\mathbf{1}_{\bar{\Phi}_L^\delta}(\bar{b}_L^{i_L})\right)=1. \tag{27}$$

Further in this article, we assume that the observation model $\mathrm{O}(z,x)$ has infinite support to rephrase that $\{z\in\mathcal{Z},x\in\mathcal{X}:\mathrm{O}(z,x)>0\}=\mathcal{Z}\times\mathcal{X}$. This assumption ensures that there are no nullified weights in (27). If the weights in (27) are normalized and all of them are nonzero, then even a single weight missing because the inner constraint started from some belief in the search tree is violated, renders (27) smaller than one. This means that the constraint with respect to the root b_0 of the belief tree is not satisfied. Since the weights are self-normalized per action, to verify that (27) equals one, we do not need to calculate weights at all. In fact, we never check the whole PC approximation. In contrast, as we already mentioned, we only verify that each indicator equals one on the way down the tree *without any lookahead*.

Remark: The assumption that the observation model has infinite support is not mandatory. If some weight is zero, then it means that the value of the corresponding indicator does not matter. However, our algorithm would still require that this value be one.

B. Bounded Support Motion Model

Suppose $\bar{b}^{\text{safe}}$ is represented by the finite set of particles, and belief update ψ is a PF. If the motion model $\mathrm{T}(x',a,x)$ has a support encapsulating the whole space $\mathbb{R}^d$, where d is a

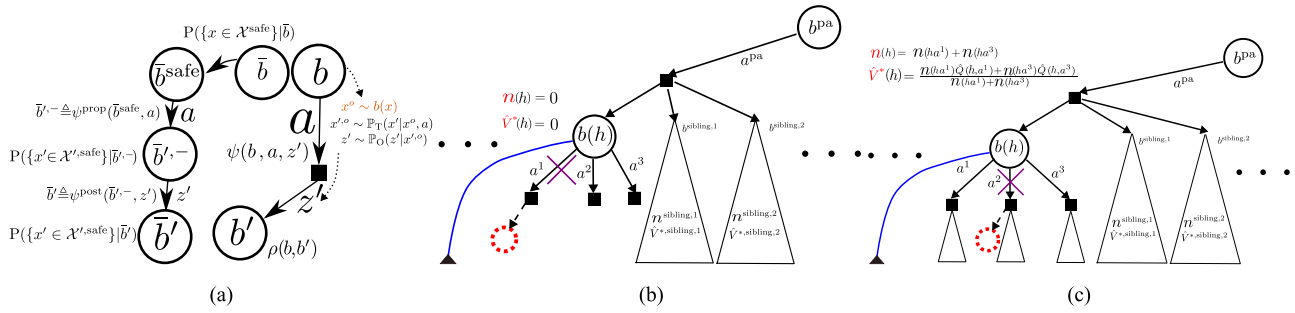


Fig. 4. (a) Conceptual illustration of maintaining a pair of posterior beliefs, b and b' are used for the reward operator, while $\bar{b}$, $\bar{b}'^{,-}$ and $\bar{b}'$ for the safety operator. We create observation using belief b highlighted by the brown color. To this end, we first sample x^o from $b(x)$. We then sample subsequent state $x'^{,o}$ using motion model $\mathbb{P}_{\mathrm{T}}(x'|x^o,a)$ and then the observation z' from $\mathbb{P}_{\mathrm{O}}(z'|x'^{,o})$. In the subfigures (b) and (c), we visualize the cleaning of the belief tree in case that the new belief violated the inner constraint. By blue color, we mark elements related to *optional* rollout. By the thick red dashed circle, we mark a newly expanded belief node. (b) In this scenario, the belief node is a first child expanded from the first selected action a^1. The expanded belief violates the inner constraint. Thus, we need to prune action a^1. Because we have not updated the visitation count of b yet, we only need to delete action a^1. (c) Harder scenario for cleaning the tree. Here, we need to perform appropriate fixes to the action-value-estimates after we prune a^2. Note that UCB or PUCT tries each action from each belief infinitely many times.

dimension, eventually, at the limit of BMCTS-DPW, for every tried action, it will be sampled unsafe belief (unsafe set of particles). However, we know that a robot cannot teleport, and truncation of the motion model is, therefore, natural. In fact, it is assumed often times to be Gaussian to bring infinite support to the table to alleviate the complexity of the solution. Without truncation for every action a, it is possible that the propagated state sample intended for observation creation will be unsafe and render the next in time posterior belief also unsafe.

Using our further presented method, we build a tree solely from safe actions. Do note that all our algorithms can be run with various belief-dependent operators. It is customary to maintain a pair of posterior beliefs $\bar{b}$ and b, as visualized in Fig. 4(a) or just maintain a single belief b. Further, we stick to the former scheme, as in Fig. 4(a).

V. Algorithms

This section focuses on our actual algorithms that overview our suggested approach. Before we begin this section, we must clarify that from now on we slightly change the notations in text and the algorithms. The sets $C(b), C(h), C(b),$ and $C(ba)$ contain now actions, beliefs, and histories and not global in tree indices. We will clarify this aspect for each algorithm.

A. PFPUCT Algorithm Clarification

For the sake of completeness, we present the PFT-DPW algorithm [19] with modifications to satisfy proof of the convergence in probability with exponential rate to optimal action-value at each belief-action node, namely, Algorithm 1. This algorithm will be needed to prove the convergence of our probabilistically constrained approach. Let us specify the modifications. The first modification is the polynomial PW for both actions (line 22) and the observations (line 10) with the depth-dependent parameters. Thus, we name Algorithm 1 particle filter polynomial upper confidence tree (PFPUCT). Purely for the clarity of the exposition, in Algorithm 1, we stick to the PF as a belief update and index places in the search tree by the beliefs instead of histories, and our reward depends only on a pair of consecutive in time beliefs. Any other belief update operator would be suitable. The only complication would be the indexing with histories instead of beliefs. The second modification is that the algorithm selects already sampled posterior belief according to minimal among children visitation count (line 14) instead of uniform sampling. The third modification is that the rollout is canceled. To the best of the authors' knowledge, we are first to present such a modified PFT-DPW. The set of actions that are children of each

Algorithm 1: PFPUCT.

1: **procedure** PLAN(b)
2: **for** m iterations or timeout **do**
3: SIMULATE($b, d_{\max}$)
4: **end for**
5: **return** $\arg\max_{a\in C(b)} \hat{Q}(ba)$
6: **end procedure**
7: **procedure** SIMULATE(b, d)
8: **If** $d == 0$ **then return** 0 **end If**
9: $a \leftarrow$ ACTIONPROGWIDEN(b)
10: **if** $\lfloor n(ba)^{\alpha_{o,d}} \rfloor > \lfloor (n(ba)-1)^{\alpha_{o,d}} \rfloor$ **then**
11: $b' \sim \mathrm{T}_b(b,a,b')$, $r' \leftarrow \rho(b,b')$
12: $C(ba) \leftarrow C(ba) \cup \{(b', r')\}$
13: **else**
14: $(r', b') \leftarrow \arg\min_{\{(r',b')\}\in C(ba)} n(b')$
15: **end if**
16: $r^{\text{lace}} \leftarrow r' + \gamma$SIMULATE($b', d-1$)
17: $n(b) \leftarrow n(b)+1$, $n(ba) \leftarrow n(ba)+1$
18: $\hat{Q}(ba) \leftarrow \hat{Q}(ba) + {}^{r^{\text{lace}}-\hat{Q}(ba)}/_{n(ba)}$
19: **return** r^{lace}
20: **end procedure**
21: **procedure** ActionProgWiden(b)
22: **if** $\lfloor n(b)^{\alpha_{a,d}} \rfloor > \lfloor (n(b)-1)^{\alpha_{a,d}} \rfloor$ **then**
23: $a \leftarrow$ NextAction(b),
24: $C(b) \leftarrow C(b) \cup \{a\}$
25: **end if**
26: **return** $\arg\max_{a\in C(b)} \hat{Q}(ba) + \sqrt{{}^{n(b)^{e_d}}/_{n(ba)}}$
27: **end procedure**

Listing 2: Common Procedures for Algorithms 3 and 6.

1: **procedure** PLAN(belief: b_0, horizon: L)
2: **for** m iterations or timeout **do**
3: SIMULATE(b_0, b_0, $\{b_0\}$, L) ▷ A single tree query
4: **end for**
6: **return** ACTIONSELECTION(b_0, $\{b_0\}$, 0)
7: **end procedure**

belief b, we denote by $C(b)$. Similarly, the set $C(ba)$ contains the pairs (b',r'). The robot calls the algorithm by initiating the `Plan` routine. Each call to `Simulate` defines descending down the tree by cycling through the BMDP loop depicted in Fig. 3(a) while selecting the action by `ActionProgWiden`. DPW for observation (line 10) decides either to sample a new belief from T_b or select an already present belief in the search tree in order to improve the action-value estimates at the deeper level of the search tree. The initial values of visitation counts $n(b), n(ba)$, and $\hat{Q}(ba)$ [q^{init} in Fig. 2(b)] are zero.

B. Detailed Description of PC-SBMCTS-DPW and PC-SBMCTS-PUCT

This section describes Algorithm 3 summarizing our main result. We name Algorithm 3 probabilistically constrained safe beliefs MCTS-DPW (PC-SBMCTS-DPW). Similar to [19], we present a provable modified variant recapped by Algorithm 6. As we describe in Section IV, our method can be utilized with safe beliefs and without. We call the algorithm maintaining a single set of beliefs defined by (1) and (2) probabilistically constrained belief MCTS-DPW (PC-BMCTS-DPW).

The entry point of both these algorithms, listed in Listing 2, is a loop over the tree queries. The difference between the algorithms is in the `Simulate` function. We name a single call to `Simulate` a *tree query*. In each tree query, both algorithms descend with the lace of observations and actions intermittently, calculate the beliefs and rewards along the way, and ascend back to the root of the belief tree. Once, on the way down the tree, the unsafe belief is encountered, we clean such action from the search tree and fix the action value estimates of all ancestor belief action nodes. Similar to the classical BMCTS-DPW, our approach can be run with rollout or without. In addition, if we do not want to use safe beliefs for the constraint, then we only need to remove parts marked by the brown color in Algorithms 3 and 6 and use regular belief instead. We also present a polynomial variant of our approach, Algorithm 6, which we named PCSBMCTS with polynomial upper confidence tree (PCSBMCTS-PUCT). In the next section, we prove the convergence with an exponential rate of Algorithm 6 in probability. Note that we cache the values of the summation of the cumulative reward for all belief nodes for both algorithms. This happens in line 32 of Algorithm 3 and line 27 of Algorithm 6, where we denote $S(h) \triangleq \hat{V}^*(h) \cdot n(h)$. It is noteworthy that in very large, countably infinite or continuous action spaces, the action sampler is the function `NextAction`. It must be designed such that some safe action is sampled first, e.g., zero action $\mathbf{0}$.

Algorithm 3: PC-SBMCTS-DPW.

1: **procedure** SIMULATE(b, $\bar{b}$, h, d)
2: **If** $d == 0$ **then return** 0 **end If**
3: $\bar{b}^{\text{safe}} \leftarrow$ MAKEBELIEFSAFE($\bar{b}$)
4: SafeActionFlag $\leftarrow$ false; NewNodeFlag $\leftarrow$ false; SampledActionFlag $\leftarrow$ false
5: **while** not(SafeActionFlag) **do**
6: $a \leftarrow$ ACTIONSELECTION(h, c)
7: $b'^{-} \leftarrow \psi^{\text{prop}}(b, a)$, $\bar{b}'^{,-} \leftarrow \psi^{\text{prop}}(\bar{b}^{\text{safe}}, a)$
8: **if** $|C(ha)| \leq k_o n(ha)^{\alpha_o}$ **then** ▷ obs. PW
9: $z' \sim \mathbb{P}_{\mathrm{O}}(z'|x'^{,o}); x'^{,o} \sim b'^{-}$
10: $\bar{b}' \leftarrow \psi(\bar{b}^{\text{safe}}, a, z')$, (can be $\bar{b}' \leftarrow \psi^{\text{post}}(\bar{b}'^{,-} z')$)
11: **if** $\mathbf{1}_{\{\phi(\bar{b}'^{,-}) \geq \delta, \phi(\bar{b}') \geq \delta\}}(\bar{b}'^{,-}, \bar{b}') == 0$ **then**
12: CLEANTREE(h, a) ▷ Algorithm 5.
13: **Continue** ▷ Jump to line 14
14: **else**
15: SafeActionFlag $\leftarrow$ true
16: **end if**
17: $b' \leftarrow \psi^{\text{post}}(b'^{,-}, z')$, $r' \leftarrow \rho(b, a, z', b')$
18: $C(ha) \leftarrow C(ha) \cup \{(z', r', b')\}$
19: NewNodeFlag $\leftarrow$ true
20: **else**
21: SafeActionFlag $\leftarrow$ true
22: $\{(z', r', b')\} \leftarrow$ sample uniformly from $C(ha)$
23: **end if**
24: **end while**
25: **if** NewNodeFlag and RolloutFlag **then**
26: $r^{\text{lace}} \leftarrow r' + \gamma$SAFEROLLOUT($b'$, $d-1$)
27: **else**
28: $r^{\text{lace}} \leftarrow r' + \gamma$SIMULATE($b', \bar{b}', haz', d-1$)
29: **end if**
30: $n(h) \leftarrow n(h) + 1$, $n(ha) \leftarrow n(ha) + 1$
31: $\hat{Q}(ha) \leftarrow \hat{Q}(ha) + \frac{r^{\text{lace}} - \hat{Q}(ha)}{n(ha)}$
32: $S(h) \leftarrow S(h) + r^{\text{lace}}$
33: **return** r^{lace}
34: **end procedure**
35: **procedure** ACTIONSELECTION (b, h, c)
36: **if** $|C(h)| \leq k_a n(h)^{\alpha_a}$ and (SampledActionFlag is false) **then** ▷ action PW
37: $a \leftarrow$ NEXTACTION(h)
38: $C(h) \leftarrow C(h) \cup \{a\}$,SampledActionFlag $\leftarrow$ true
39: **end if**
40: return $\arg\max_{a \in C(h)} \hat{Q}(ha) + \kappa \sqrt{\log n(h) / n(ha)}$
41: **end procedure**

a) Safe rollout: The rollout is not necessary for applying BMCTS. Without rollout, upon opening a new belief node, the BMCTS would call the `Simulate`. Nevertheless, the rollout helps to provide better results in finite time, and apparently accelerates convergence (We did not find any rigorous analysis for that. It is not clear to what theoretical values such a converge tends). With this motivation in mind, we present the `SafeRolloutPolicy` routine for our approach summarized by Algorithm 4. It selects action randomly, which myopically fulfills the sample approximation of myopic PC based on m

Algorithm 4: Myopically Safe Rollout Action Selection.

procedure SAFEROLLOUTPOLICY(b, $\mathcal{A}$)
  $\mathcal{A} \leftarrow$ shuffle($\mathcal{A}$). $V^* \leftarrow -\infty$
  for $a \in \mathcal{A}$ **do**
    for m iterations **do**
      Calculate propagated belief b'^- from b and a
      $b' \leftarrow \psi(b, a, z')$; $z' \sim \mathbb{P}_{\mathrm{O}}(z'|x'^{,o})$; $x'^{,o} \sim b'^-$
      Calculate $\mathbf{1}_{\{\phi(b'^-)\geq\delta, \phi(b')\geq\delta\}}(b'^-, b')$
    end for
    $\hat{V}^{(m)} \leftarrow \frac{1}{m}\sum_{i=1}^{m} \mathbf{1}_{\{\phi(b'^-)\geq\delta, \phi(b')\geq\delta\}}(b'^{,i,-}, b'^{,i})$
    if $\hat{V}^{(m)} \geq 1-\epsilon$ **then** ▷ Note that we added ϵ here
      return a
    else if $\hat{V}^{(m)} > \hat{V}_*^{(m)}$ **then**
      $V^* \leftarrow V$, $a^* \leftarrow a$
    end if
  end for
  return a^*
end procedure

Algorithm 5: Cleaning Belief Tree to be Safe.

procedure CLEANTREE(h, a)
  Delete all children b' of ba belief-action node and delete ba itself $C(h) \leftarrow C^{\text{intree}}(h) \setminus \{a\}$
  If $n^{\text{intree}}(h) == 0$ **return end if**
  $n(h) \leftarrow n^{\text{intree}}(h) - n^{\text{intree}}(ha)$
  if b is root **then**
    $C^{\text{intree}}(h) \leftarrow C(h)$, $n^{\text{intree}}(h) \leftarrow n(h)$,
    return
  else
    Assemble $\hat{V}^*(h)$ using (29)
  end if
  while true **do**
    if $b(h)$ is root **then**
      $n^{\text{intree}}(h) \leftarrow n(h)$,
      return
    end if
    Identify a^{pa} which is parent to $b(h)$
    Identify b^{pa} such that $b^{\text{pa}}a^{\text{pa}}$ is a belief action node which is parent of b
    $n(h^{\text{pa}}a^{\text{pa}}) \leftarrow n^{\text{intree}}(h^{\text{pa}}a^{\text{pa}}) - n^{\text{intree}}(h) + n(h)$
    Reconstruct $\hat{Q}(h^{\text{pa}}a^{\text{pa}})$ using (30) and put $n^{\text{intree}}(h) \leftarrow n(h)$ and $S^{\text{intree}}(h) \leftarrow S(h)$.
    $n(h^{\text{pa}}) \leftarrow n^{\text{intree}}(h^{\text{pa}}) - n^{\text{intree}}(h^{\text{pa}}a^{\text{pa}}) + n(h^{\text{pa}}a^{\text{pa}})$,
    Assemble $\hat{V}^*(h^{\text{pa}})$ using (31) and put $\hat{Q}^{\text{intree}}(h^{\text{pa}}a^{\text{pa}}) \leftarrow \hat{Q}(h^{\text{pa}}a^{\text{pa}})$ and $n^{\text{intree}}(h^{\text{pa}}a^{\text{pa}}) \leftarrow n(h^{\text{pa}}a^{\text{pa}})$.
    $b(h) \leftarrow b^{\text{pa}}(h^{\text{pa}})$, $h \leftarrow h^{\text{pa}}$.
  end while
end procedure

samples. If no feasible action exists (which is not possible with our method since we always have an action "do not do anything"), then we select an action maximizing the sample approximation mentioned before. Note that the action space must be discretized for the safe rollout outlined by Algorithm 4. For clarity, safe rollout does not maintain two sets of beliefs. Our rollout maintains only the beliefs defined by (1) and (2).

C. Constraint Violation and Efficient Tree Cleaning

We now explain how we prune all dangerous actions (see Definition 1) from the search tree and thereby our search tree always contains only the safe actions (see Corollary 1). Actions at the root and the tree future policy, which is stochastic due to exploration, are such that the PC is fulfilled starting from each belief node in the search tree. Purely for the sake of clarity, we set $\gamma = 1$ in this section. Extension to general γ does not pose any problem. Suppose that our PC-SBMTCS-DPW, Algorithm 3 (or Algorithm 6 without breaking the ties), is currently at a belief node $b(h)$ in the belief tree, with a corresponding history h defining the unique place h in the belief tree. The algorithm selects an action according to (7), and suppose it creates a new belief. Every time we create a new belief node to be added to the search tree, we obtain b' for the reward calculation and corresponding $\bar{b}'^{,-}$ and $\bar{b}'$ for the constraint. We then check if $\phi(\bar{b}'^{,-}) \geq \delta$ and $\phi(\bar{b}') \geq \delta$, and if both inequalities are satisfied, then we add the newly created node to the belief tree. If $\phi(\bar{b}'^{,-}) < \delta$ or $\phi(\bar{b}') < \delta$, then we shall prune an action leading to this belief node from the belief tree and fix the $\hat{Q}$ upward since the laces emanating from the cleaned action participate in $\hat{Q}$ of every ancestor belief action node. Due to the fact that we assemble $\hat{Q}$ at each belief-action node from laces, at node h, it holds that $\hat{V}^*(h) = \sum_{a\in C(h)} \frac{n(ha)\hat{Q}(ha)}{n(h)}$ and $n(h) = \sum_{a\in C(h)} n(ha)$. The $\hat{Q}$ of the parent reads

$$\hat{Q}(h^{\text{pa}}a^{\text{pa}}) = \frac{(n(h)+1)(\rho(b^{\text{pa}}, a^{\text{pa}}, b) + \hat{V}^*(h) + \text{roll}(h)) + \dots}{n(h^{\text{pa}}a^{\text{pa}})} \tag{28}$$

where the summation over all the sibling subtrees and the visitation count appears as $n(h^{\text{pa}}a^{\text{pa}}) = n(h)+1+n^{\text{sibling},1}+1\dots$, where by the red color, we denote values of the current belief node, and by the blue color, we denote optional values related to the activation of the rollout. We now turn to an explanation of how to clean the tree efficiently using subtraction and adding operations instead of assembling action-value estimates and visitation counts from scratch using updated values down the tree. Suppose the action leading to the newly added belief does not have sibling subtrees corresponding to another actions, and this belief is the first child of such an action, as depicted in Fig. 4(b). In this case, the visitation counts $n(h)$ and $\hat{V}^*(h)$ are not present yet within $\hat{Q}(h^{\text{pa}}a^{\text{pa}})$. This is because the only rollout was commenced from $b(h)$. We can just delete the action leading to the newly created belief node.

We, now, focus on a more interesting setting depicted in Fig. 4(c). After we deleted the subtree defined by the belief node $b(h)$ and action a [a^2 in Fig. 4(c)], we need to update the visitation count $n(h)$ as such: $n(h) \leftarrow n^{\text{intree}}(h) - n^{\text{intree}}(ha)$, where we denote values that are currently in the belief tree by $\square^{\text{intree}}$. As shown in Fig. 4(c), we have that $\hat{V}^{*,\text{intree}}(h) = \frac{\sum_{a\in C^{\text{intree}}(h)} n^{\text{intree}}(ha)\hat{Q}^{\text{intree}}(ha)}{\sum_{a\in C^{\text{intree}}(h)} n^{\text{intree}}(ha)}$. At this point, we have everything to calculate the updated value function at belief $b(h)$

indexed by history h as such

$$S(h) \leftarrow S^{\text{intree}}(h) - \hat{Q}^{\text{intree}}(ha) n^{\text{intree}}(ha). \tag{29}$$

Recall that $S(h) \triangleq \hat{V}^*(h) \cdot n(h)$, $S^{\text{intree}}(h) \triangleq \hat{V}^{*,\text{intree}}(h) \cdot n^{\text{intree}}(h)$. For an efficient update by (29), we need to cache the sum $S^{\text{intree}}(h)$ of the cumulative reward laces for each belief node h. In addition, we need to subtract the deleted action a from the set of children of $b(h)$, namely $C(h) \leftarrow C^{\text{intree}}(h) \backslash \{a\}$. If $b(h)$ is a root node, we just update its visitation count $n(h)$ to a new value. We do not need to store $S(h)$ for a root belief. Else, we identify a parent action and node of $b(h)$ marked a^{pa} and b^{pa}, respectively. We need to calculate $n(h^{\text{pa}}a^{\text{pa}})$ as such, $n(h^{\text{pa}}a^{\text{pa}}) \leftarrow n^{\text{intree}}(h^{\text{pa}}a^{\text{pa}}) - n^{\text{intree}}(h) + n(h)$. We then shall update $\hat{Q}(h^{\text{pa}}a^{\text{pa}})$ as such

$$\begin{aligned}
&\hat{Q}(h^{\text{pa}}a^{\text{pa}}) \cdot n(h^{\text{pa}}a^{\text{pa}}) \leftarrow \hat{Q}^{\text{intree}}(h^{\text{pa}}a^{\text{pa}}) \cdot n^{\text{intree}}(h^{\text{pa}}a^{\text{pa}}) - \\
&\left(n^{\text{intree}}(h)+1\right) \left(\rho(b^{\text{pa}}a^{\text{pa}}b(h)) + \hat{V}^{*,\text{intree}}(h) + \text{roll}^{\text{intree}}(h)\right) \\
&+ \left(n(h)+1\right) \left(\rho(b^{\text{pa}}a^{\text{pa}}b(h)) + \hat{V}^{*}(h) + \text{roll}^{\text{intree}}(h)\right).
\end{aligned} \tag{30}$$

Note that (30) encompasses both cases. The case when the subtree is deleted (29) and the case when only the visitation counts down the tree and the $\hat{Q}$ are updated (upper levels of the belief search tree). The value of $\hat{V}^*(h)$ subsumes both cases. In the latter case, its update reads as such

$$S(h) \leftarrow S^{\text{intree}}(h) - \hat{Q}^{\text{intree}}(ha) n^{\text{intree}}(ha) + \hat{Q}(ha) n(ha). \tag{31}$$

We now calculate a new visitation count of b^{pa} using $n(h^{\text{pa}}) \leftarrow n^{\text{intree}}(h^{\text{pa}}) - n^{\text{intree}}(h^{\text{pa}}a^{\text{pa}}) + n(h^{\text{pa}}a^{\text{pa}})$, and $S(h^{\text{pa}})$ using (31) and renaming there the history from h to h^{pa} and the action from a to a^{pa}. Now, we can treat $b^{\text{pa}}(h^{\text{pa}})$ similarly as we treated $b(h)$. We outlined the tree cleaning procedure in Algorithm 5. To conclude, this way our search tree always consists of only safe actions (not dangerous with respect to Definition 1).

VI. Guarantees

We now turn to guarantees of our approach. We first show that Algorithm 1 fulfills the proof innovated by Auger et al. [18]. We then do the reduction of Algorithm 6 to Algorithm 1. To this end, we first must show that sampled beliefs b' from $\text{T}_b(b, a, b')$ are independent identically distributed (i.i.d.) given b.

A. Sampling From BMDP Transition Model T_b

PFT-DPW represents all beliefs in the search tree by a fixed and predefined number of weighted particles. PF update first promotes each particle of b with the motion model to create propagated belief $b'^{,-}$. It, then, samples the state $x'^{,o} \sim b'^{,-}$. Further, it samples an observation from the observation model $z' \sim \mathbb{P}_{\text{O}}(z'|x'^{,o})$ given sampled from the propagated belief state, updates the weights of weighted samples of $b'^{,-}$, and finally performs resampling. See Fig. 5. Such a process matches (6) as follows. It approximates the observation likelihood $\mathbb{P}(z'|b,a)$ in (6) by first approximating $\mathbb{P}(x'|b,a)$ by $|C(ha)|$ i.i.d. samples (number of children of ha, where h corresponds to $b(h)$) of

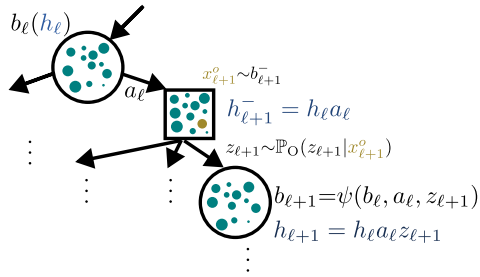


Fig. 5. Sampling from T_b. The state $x^o_{\ell+1}$ is sampled from propagated belief $b^-_{\ell+1}$. It then renders an observation $z_{\ell+1}$. The belief $b^-_{\ell+1}$ is then updated to $b_{\ell+1}$.

states $x'^{,o}$. This is because to sample a single sample from each propagated belief is equivalent to sample $x^o \sim b(h)$ and pass the sample through motion model T. The $C(ha)$ is the set of sampled observations such that

$$\begin{aligned}
\mathbb{P}(x'|b(h),a) &= \int_x \mathbb{P}_{\text{T}}(x'|x,a)\mathbb{P}(x|b(h))\text{d}x \approx \\
&1/|C(ha)| \sum_{i=1}^{|C(ha)|} \delta(x' - x'^{,o,i}).
\end{aligned} \tag{32}$$

A single $x'^{,o,i}$ is sampled from each created propagated belief at each arrival to h when the new observation is going to be created. Recall that dropping corresponding x^o sample after passing them through T is equivalent to marginalization. The samples $x'^{,o,i}$ stay independent and identically distributed. To clarify, (32) is not the propagated belief. This distribution is constructed from i.i.d. samples and used solely to approximate the observation likelihood portrayed by (33).

Utilizing marginalization followed by the chain rule, the observation likelihood is then

$$\begin{aligned}
\mathbb{P}(z'|b,a) &= \int_{x' \in \mathcal{X}'} \mathbb{P}(z', x'|b, a)\text{d}x' = \\
&= \int_{x' \in \mathcal{X}'} \mathbb{P}_{\text{O}}(z'|x')\mathbb{P}(x'|b,a)\text{d}x' \\
&\approx \int_{x' \in \mathcal{X}'} \mathbb{P}_{\text{O}}(z'|x') 1/|C(ha)| \sum_{i=1}^{|C(ha)|} \delta(x' - x'^{,o,i})\text{d}x' \\
&\approx 1/|C(ha)| \sum_{i=1}^{|C(ha)|} \mathbb{P}_{\text{O}}(z'|x'^{,o,i}).
\end{aligned} \tag{33}$$

From each $\mathbb{P}_{\text{O}}(z'|x'^{,o,i})$ for $i = 1 \dots |C(ha)|$, BMCTS-DPW samples a single observation making $\mathbb{P}(z'|b,a) \approx 1/|C(ha)| \sum_{z'^{,i} \in C(ha)} \delta(z' - z'^{,i})$. Note that indices $i \in [1 \dots |C(ha)|]$ are distributed uniformly according to $1/|C(ha)|$. The BMDP motion model is then $\text{T}_b(b,a,b') \approx 1/|C(ha)| \sum_{z'^{,i} \in C(ha)} \delta(b' - \psi(b,a,z'^{,i}))$. We showed the visualization of this process in Fig. 5. Such a sampling process produces i.i.d. samples of the posterior beliefs b' from the correct model T_b with a single discrepancy that belief update ψ is deterministic,

namely, $\mathbb{P}(b'|b,a,z')=\delta(b'-\psi(b,a,z'))$. We neglect this aspect in this article and leave it for future research. See Section VIII.

Remark: If the belief is parametric, e.g., Gaussian, then nothing changes. We can either sample $x^o_\ell \sim b_\ell$ and pass through the motion model T to create $x^o_{\ell+1}$ and then observation, or propagate belief parameterically first and then sample $x^o_{\ell+1}\sim b^-_{\ell+1}$. This is similar to the situation with PF, where $x^o_{\ell+1}$ can be a particle of $b^-_{\ell+1}$ or a sampled b_ℓ particle and passed through T. This is because we assume that PF is a deterministic function of the belief, the performed action, and the observation received when the action is completed.

B. Provable BMCTS-DPW (Algorithm 1)

The paper [18] presents its proof for the MDP setting. We are not aware of any paper that explicitly posed that Algorithm 1 fulfills the proof innovated in [18]. Thus, we show it as lemma.

Lemma 2: Disregarding the error introduced by the stochastic belief update, Algorithm 1 with $\alpha_{a,d}, \alpha_{o,d}$, and e_d defined as described in [18] satisfies the proof presented in [18]. To specify, at each belief-action node, it converges in probability and with an exponential rate to the optimal action-value function.

The proof of Lemma 2 is rather simple. Following the explanation in Section II-B, Algorithm 1 samples i.i.d. posterior beliefs from the valid BMDP transition model T_b. The proof is finished.

C. Necessity of the Cleaning Routine

When the action space $\mathcal{A}$ is continuous, the actions are uncountable. In this section, we show the necessity of our cleaning routine in the first place. In continuous domains, our algorithms *endlessly* sample new actions; see line 23 in Algorithm 1, line 37 in Algorithm 3, and line 32 in Algorithm 6. Suppose, at the node h, we have an action a sampler on top of the continuous probability space $(\mathcal{A},\mathcal{F},\mathrm{P})$, where $\mathcal{A}$ is the outcomes space, $\mathcal{F}$ is the events space, and P is the probability. Suppose the set $D\in\mathcal{A}$ is dangerous such that $\mathrm{P}(D|h)>0$. Note that the probability here depends on the history (belief) we focus on. Time marches to infinity in countable steps so that the samples are countable.

We now show that the cleaning tree routine is necessary due to the fact that we will sample an infinite amount of unsafe actions, and this can shift the expectations (at the limit of convergence) with respect to actions in values maintained in the search tree.

Without our pruning, we will obtain infinitely many unsafe actions in each unsafe set D. It can be seen using the second Borel–Cantelli lemma. Toward this end, let us define the event $E^i \triangleq \{a^i\sim\mathbb{P}(a|h):a^i\in D\}$, where sampled action i is a member of set D. The events E^i are independent since we sample actions independently. The series $\sum_{i=1}^{\infty}\mathrm{P}(E^i|h)$ are divergent since $\mathrm{P}(E^i|h)=\mathrm{P}(D|h)>0$ by definition. Thus, $\mathrm{P}(\bigcap_{j=1}^{\infty}\bigcup_{i=j}^{\infty}E^i|h)=1$, namely, the event that the sampled action is a member of set D occurs infinitely often times. To rephrase that, without our pruning, we would have sampled infinitely many dangerous actions and, therefore, the expectations can undergo a shift, and even if at the root of the belief tree we select an optimal safe action, the influence of unsafe future actions can be substantial.

D. Convergence Guarantees

All three algorithms we present in this article, namely, Algorithms 1, 3, and 6, sample actions and observations. The number of samples of both is marching to infinity in discrete steps, although the action state and observation spaces are continuous. Thus, we focus on convergence in probability. Recall that we neglect the belief representation error and treat particle belief as the true belief. Our goal in this section is to prove that Algorithm 6 converges in probability to the following objective at each history node h:

$$Q^{\pi^*}(b(h),a;\boldsymbol{\rho}) \text{ subject to} \tag{34}$$

$$\mathrm{P}\left(c{=}1|\bar{b}(h),a,\pi^*\right){=}1 \tag{35}$$

while the convergence of (34) is with exponential rate with respect to tree queries. In (34), we omitted the time indices. Alas, we cannot say anything about the convergence rate of (35). Let us give an intuition. The main problem with convergence in continuous domains is that Algorithms 1, 3, and 6 endlessly sample new actions and observations. Thus, to ensure convergence in probability, the algorithm shall provide two things: 1) sample new actions and observations with a slowing rate to ensure that previous samples defining histories h have enough time to approximate the objectives $\hat{Q}^{\pi^*}(b(h),a;\boldsymbol{\rho})$ better and better. 2) Moreover, the algorithm must select already sampled actions and observations such that each action and observation defined objective $\hat{Q}^{\pi^*}(b(h),a;\boldsymbol{\rho})$ receives a sufficient amount of samples. In other words, it shall distribute samples fairly. The first aspect already happens with DPW from Algorithm 3. Before we plunge into the proof, behold in the following lemma that the rate of sampling actions and observations is not so different in the classical DPW [19] used in Algorithm 3 and the polynomial variant [18] used in Algorithm 6.

Lemma 3: Fix belief node $b(h)$ in belief tree and belief action node ha, $k_a{=}k_o{=}1$ in Algorithm 3 and select in Algorithms 3 and 6 same $\alpha_{o,d}$ and $\alpha_{a,d}\in(0,1)$ in both algorithms (can be depth dependent). The condition $|C(ha)|{\leq}n(ha)^{\alpha_{o,d}}$ is equivalent to $\lfloor n(ha)^{\alpha_{o,d}}\rfloor{>}\lfloor(n(ha){-}1)^{\alpha_{o,d}}\rfloor$. In a similar manner, $|C(h)|{\leq}n(h)^{\alpha_a,d}$ is equivalent to $\lfloor n(h)^{\alpha_{a,d}}\rfloor > \lfloor(n(h)-1)^{\alpha_{a,d}}\rfloor$.

Proof sketch : It is sufficient to prove that the first claim is identical since both have identical structure. Let us focus on $|C(ha)|\leq n(ha)^{\alpha_{o,d}}$. The new child is added if and only if the visitation $n(ha)^{\alpha_{o,d}}$ passes the subsequent integer at some visitation of node ha. This happens if and only if $\lfloor n(ha)^{\alpha_{o,d}}\rfloor > \lfloor(n(ha)-1)^{\alpha_{o,d}}\rfloor$. This is because $n(ha){-}1$ is the visitation count at the previous visit of node ha. ■

To guarantee the second aspect in the list, for each h in the search tree, we shall change the way Algorithm 3 selects already sampled history–action nodes ha that are children of h and the way Algorithm 3 selects already sampled history nodes haz' that are children of h. Proper selection of ha nodes is achieved by PUCT, line 35 in Algorithm 6 similar to line 26 in Algorithm 1 instead of UCB (line 40 in Algorithm 3). The polynomial variant of DPW used in Algorithm 6 allows each history–action node to be visited a sufficient amount of times before the new action is introduced. Correct selection of the haz' nodes is obtained by

selecting the child with the smallest visitation count, line 21 in Algorithm 6 similar to line 14 in Algorithm 1 instead of uniform selection in line 22 of Algorithm 3. Furthermore, we must cancel the rollout.

In our proof, we will show that Algorithm 6 is equivalent to Algorithm 1. This, in turn, will imply that the proof for Algorithm 1 applies.

We now turn to the proof of the convergence in probability with an exponential rate of the PUCT version of our approach (see Algorithm 6). For clarity, we list down the changes between Algorithms 3 and 6.

1) In Algorithm 6, we have polynomial double progressive widening with depth-dependent parameters defined in [18].
2) The rollout in Algorithm 6 is missing.
3) If Algorithm 3 decided not to open a new branch in terms of observation, then it samples the triple $\{z', r', b'\}$ uniformly from $C(ha)$ (line 22 highlighted by the blue color). In contrast, Algorithm 6 selects the child with a minimal visitation count (line 21 highlighted by the blue color).

The following theorem provides the soundness of Algorithm 6.

Theorem 3 (Convergence with Exponential Rate in Probability): Every belief h and belief action node ha of Algorithm 6, equipped with our pruning mechanism from Section V-C and summarized by Algorithm 5, converges in probability to the optimal value function $V^*(b(h))$ and action-value function $Q(b(h)a)$, respectively, while satisfying the PC starting from the belief action node ha, namely, $\mathrm{P}(c{=}1|\bar{b}(h), a, \pi^*){=}1$. Furthermore, the convergence rate is exponential in terms of tree queries with respect to objective (34).

Next, we provide the proof under rather mild assumptions. To be specific, we must assume that the reward lies in a bounded interval and that the sampling of actions covers the entire space with an arbitrary precision. For a more precise definition, see Definition 2. Moreover, we assume that, if the sample approximations (21) and (27) of (35) are converging in probability to some value, then they are converging to the right value, namely, to (35). We leave proving this claim to future research. Our proof is valid for both approaches, namely, with making belief safe before pushing forward in time with action and observation and without (in this case, the constraint at each belief-action node is $\mathrm{P}(c{=}1|b(h), a, \pi^*){=}1$). Similar to Sunberg and Kochenderfer [19], we leverage the proof by Auger et al. [18].

The following claim is required to understand [18], and we now give an informal proof missing in [18].

Lemma 4: The kth child of node ha in Algorithm 6 is added on visit $n(ha) = \lceil k^{\frac{1}{\alpha}} \rceil \triangleq n_k(ha)$.

Observe that the left-hand side of $\lfloor n(ha)^{\alpha_{o,d}} \rfloor > \lfloor (n(ha){-}1)^{\alpha_{o,d}} \rfloor$ jumped $\lfloor n(ha)^{\alpha_{o,d}} \rfloor$ times and the right-hand side lagged exactly by a single visitation. So, the inequality is fulfilled exactly $\lfloor n(ha)^{\alpha_{o,d}} \rfloor$ times. Moreover, the *first* $n(ha)$ such that $n(ha)^{\alpha_{o,d}}$ passes a subsequent integer assures the jump. Meaning, we have two cases. The first case is $n(ha)^{\alpha_{o,d}}{=}\lfloor n(ha)^{\alpha_{o,d}} \rfloor{=}k$, and taking $k^{\frac{1}{\alpha_{o,d}}}{=}\lceil k^{\frac{1}{\alpha_{o,d}}} \rceil$ is returning us back to $n(ha)$. The second case is $\lfloor n(ha)^{\alpha_{o,d}} \rfloor{+}1{>}n(ha)^{\alpha_{o,d}}{>}\lfloor n(ha)^{\alpha_{o,d}} \rfloor{=}k$. However, we know that if $k^{\frac{1}{\alpha}}$ were an integer, then it would be the previous case with a smaller $n(ha)$. $k^{\frac{1}{\alpha}}$ has to be slightly larger than an integer, so the ceil operator returns the right natural $n(ha)$. Similarly, the number of actions expanded from node h is $\lfloor n(h)^{\alpha_a, d} \rfloor$. ■

Algorithm 6: PC-SB-BMCTS-PUCT.

1: **procedure** SIMULATE(b, $\bar{b}$, h, d)
2: **If** $d == 0$ **then return** 0 **end If**
3: $\bar{b}^{\text{safe}} \leftarrow$ MAKEBELIEFSAFE($\bar{b}$)
4: SafeActionFlag $\leftarrow$ false, SampledActionFlag $\leftarrow$ false
5: **while** not(SafeActionFlag) **do**
6: $a \leftarrow$ ACTIONSELECTION(h, c)
7: $b'^{-} \leftarrow \psi^{\text{prop}}(b, a)$ and $\bar{b}'^{,-} \leftarrow \psi^{\text{prop}}(\bar{b}^{\text{safe}}, a)$
8: **if** $\lfloor n(ha)^{\alpha_{o,d}} \rfloor > \lfloor (n(ha)-1)^{\alpha_{o,d}} \rfloor$ **then**
9: $z' \sim \mathbb{P}_{\mathrm{O}}(z'|x'^{,o}); x'^{,o} \sim b'^{-}$
10: $\bar{b}' \leftarrow \psi(\bar{b}^{\text{safe}}, a, z')$,
11: **if** $\mathbf{1}_{\{\phi(\bar{b}'^{,-}) \geq \delta, \phi(\bar{b}') \geq \delta\}}(\bar{b}'^{,-}, \bar{b}') == 0$ **then**
12: CLEANTREE(h, a) ▷ Algorithm 5.
13: **Continue** ▷ Jump to line 14
14: **else**
15: SafeActionFlag $\leftarrow$ true
16: **end if**
17: $b' \leftarrow \psi^{\text{post}}(b'^{,-}, z')$, $r' \leftarrow \rho(b,a,z',b')$
18: $C(ha) \leftarrow C(ha) \cup \{(z', r', b')\}$
19: **else**
20: SafeActionFlag $\leftarrow$ true
21: $\{(z',r',b')\} \leftarrow \arg\min_{\{(z',r',b')\} \in C(ha)} n(haz')$
22: **end if**
23: **end while**
24: $r^{\text{lace}} \leftarrow r' + \gamma$Simulate($b', \bar{b}', haz', d{-}1$)
25: $n(h) \leftarrow n(h) + 1, n(ha) \leftarrow n(ha) + 1$
26: $\hat{Q}(ha) \leftarrow \hat{Q}(ha) + \frac{r^{\text{lace}} - \hat{Q}(ha)}{n(ha)}$
27: $S(h) \leftarrow S(h) + r^{\text{lace}}$ ▷ Initialized to zero
29: **return** r^{lace}
30: **end procedure**
31: **procedure** ActionSelection(b, h, c)
32: **if** $\lfloor n(h)^{\alpha_{a,d}} \rfloor > \lfloor (n(h){-}1)^{\alpha_{a,d}} \rfloor$ and (SampledActionFlag is false) **then**
33: $a \leftarrow$ NEXTACTION(h)
34: $C(h) \leftarrow C(h) \cup \{a\}$, SampledActionFlag $\leftarrow$ true
35: **end if**
36: return $\arg\max_{a \in C(h)} \hat{Q}(ha) + \sqrt{n(h)^{e_d}/n(ha)}$ ▷ PUCT
37: **end procedure**

Our cleaning routine prunes only the dangerous actions and fixes the affected visitation counts, so the proof by Auger et al. [18] is *not broken*. The new actions and the observations are sampled *endlessly*. Thus, the sample approximation of (35) converges (in probability) to the theoretical PC defined by (35).

Further, we establish the definitions and the assumptions from [18] in order to assure the validity of the proof. Some of them we take directly from [18] and [19].

Definition 2 (Regularity Hypothesis): The regularity hypothesis is the assumption that for any $\Delta{>}0$, there is a nonzero

probability to sample an action that is optimal with precision Δ. More precisely, there is a $\theta>0$ and a $p>1$ (which remain the same during the planning) such that for all $\Delta>0$

$$Q(ha)\geq V^*(h)-\Delta \text{ with a probobaility of at least } \min(1,\theta\Delta^p). \tag{36}$$

Definition 3 (Exponentially sure in n): We say that some property depending on an integer n is exponentially sure in n if there exist positive constants C, h, and η such that the probability that the property holds is at least $1-C\exp(-hn^\eta)$. Also, we need to assume that the belief-dependent reward is bounded from below and above, namely, it lies in the closed interval $[\rho^{\min},\rho^{\max}]$. Instead of $\rho:\mathcal{B}\times\mathcal{A}\times\mathcal{Z}\times\mathcal{B}\mapsto\mathbb{R}$, we require the mapping to be $\rho:\mathcal{B}\times\mathcal{A}\times\mathcal{Z}\times\mathcal{B}\mapsto[\rho^{\min},\rho^{\max}]$. Under these assumptions, the convergence result of Algorithm 6 summarized by Theorem 3 holds. Crucially, both assumptions, the regularity hypothesis (see Definition 2) and the finite support of the reward, are used merely to prove convergence with exponential rate. It has no impact on the practical applicability of the proposed method.

VII. Complexity Analysis

Let us provide the detailed analysis of the computational complexity to shed light on the scalability of the proposed method. Without our imposed safety and the safe rollout, the complexity of the BMCTS-DPW search is $O(n(b_0))$. This is because the BMCTS-DPW applies on the level of BMDP, as described in Section II-B, runs down the tree and up. In each such errand (a tree query), at most L rewards are calculated. Since with our approach a pair of posterior belief sets is maintained instead of one, each tree query $2L$ beliefs are updated. Importantly, since the number of belief particles is constant, both belief update and reward calculation consume a constant amount of time. Thus, each tree query running time is bounded from above by constant time and the overall complexity is $O(n(b_0))$. We conclude that maintaining the pair of belief sets does not contribute to asymptotic complexity.

We now turn to the analysis of the complexity added by our safety. If the action space is continuous, then the new action is sampled, using the `NextAction` routine, only if the DPW decides to sample a new action. We enforce this behavior using SampledActionFlag defined in line 4 of Algorithm 3 and Algorithm 6. Even if the appropriate visitation count leads to the decision by DPW to sample a new action, it will happen only once in the safety loop in line 5 of Algorithm 3 and Algorithm 6. The cleaning routine is initiated at most $|C(h)|-1$ times. In the worst-case scenario, it will clean all the actions up until only zero action is left. Suppose, without losing generality, that the current tree query ends at h_L. Overall, the search complexity with the cleaning tree included is $O(n(b_0)\max_{h_L\in\mathbb{T}}\sum_{h\subseteq h_L}|C(h)|)$, where $\mathbb{T}$ is the search tree so far, and by $h\subseteq h_L$, we denote each history included in h_L from the root and until h_L itself, inclusive. We now include the rollout in the development. In rollout, we use the `shuffle` method. For the rollout, the entire action space $\mathcal{A}$ must be discretized before. Therefore, the rollout does not influence the asymptotic complexity analysis. This is because selecting action in rollout is $O(1)$. To summarize this section, the asymptotical complexity of our approach is $O(n(b_0)\max_{h_L\in\mathbb{T}}\{\sum_{h\subseteq h_L}|C(h)|\})$.

VIII. Limitations and Drawbacks

We ignore the stochasticity of the belief update operator throughout this article, particularly in our convergence analysis. We leave this part for later research. Nevertheless, we tried to increase the amount of different beliefs verified for safety by running the belief update from scratch instead of updating the already present propagated belief. See line 10 in Algorithm 3.

IX. SOTA Continuous Constrained MCTS

We now firm up the loose ends and turn to the description of the existing constrained POMDP considered in an anytime setting, which will serve as our baseline.

A. Expectation Constrained Belief-Dependent POMDPs

The averaged constraint formulated with a *payoff* operator and including the propagated beliefs would be

$$\mathbb{E}\left[\sum_{\ell=0}^{L}\phi(b_\ell^-,b_\ell)\Big|b_0,\pi\right]\geq\delta. \tag{37}$$

One possibility is to define $\phi(b_\ell^-,b_\ell)=\phi(b_\ell^-)+\phi(b_\ell)$. Clearly, the cumulative averaged formulation (37) is not suitable for safety since it permits deviations of the individual safety operators ϕ. It can happen that with the low probability of future observation, the resulting posterior belief will be *extremely* unsafe. However, sometimes, the operator ϕ is naturally bounded from above. It holds that $\mathrm{P}(\{x_\ell\in\mathcal{X}_\ell^{\text{safe}}\}|\square)\leq 1$ for $\square$ being any belief (b,b^-,$\bar{b}$ or $\bar{b}^-$). Thus, if we select an operator ϕ as in (12) and $\delta=2L$, it is sufficient to ensure safety. If $\delta<2L$, then it permits deviations of the individual belief-dependent operators. Therefore, with respect to observations episodes and stochastic policy, the averaged cumulative constraint is not sufficient to assure safety. The works [6] and [7] impose the averaged cumulative constraint at the root of a belief tree as

$$V^\pi(b_0;\boldsymbol{\theta}_0)\triangleq\left[\sum_{\ell=0}^{L}\theta(b_\ell^-,b_\ell)\Big|b_0,\pi\right]\leq\delta^\theta. \tag{38}$$

We introduced the optional dependence on b^- of the cost operator θ (emphasized by turquoise color), e.g., $\theta(b_\ell^-,b_\ell)=\theta(b_\ell^-)+\theta(b_\ell)$

$$\theta(\square)=1-\mathrm{P}\left(\{x_\ell\in\mathcal{X}_\ell^{\text{safe}}\}\big|\square\right)=\mathrm{P}\left(\{x_\ell\notin\mathcal{X}_\ell^{\text{safe}}\}\big|\square\right) \tag{39}$$

with values in $\square$ being substituted by b_ℓ^- and b_ℓ, respectively. Similar to the behavior of the bounded payoff operator, here we can assure safety if $\delta^\theta=0$. This will assure that (38) is satisfied if and only if *all* $\theta(b_\ell^-,b_\ell)$ inside are zero. This is because $\mathrm{P}(\{x_\ell\notin\mathcal{X}_\ell^{\text{safe}}\}|\square)\geq 0$. In the light of the discussion about deviation of the cost values, further in this article, we assume that $\delta^\theta=0$. Now, if we set $\delta=1-\delta^\theta=1$ in our PC (10) and payoff as in (12), then two formulations are equivalent. Yet, this will happen solely with the payoff operator being as in (12),

cost as in (39), and $\delta=1$. Another option is to set cost in (38) as

$$\theta_\ell(b_\ell^-, b_\ell) \triangleq 1 - \mathbf{1}_{\Phi_\ell^\delta}(b_\ell^-, b_\ell) \tag{40}$$

with $\delta^\theta=0$ and Φ_ℓ^δ as in (11). We obtain that (38) is satisfied if and only if our PC (10) is satisfied and, in both formulations, we have the freedom to select operator ϕ and δ (we still need to assure that δ is the same in both formulations). Importantly, unlike the cost from (39), the cost from (40) cannot be represented as expectation over the state-dependent cost. This cost is general belief-dependent operator even if the payoff inside is as (12).

B. Duality-Based Approach

We now turn to the discussion about the duality-based approach in continuous spaces suggested in [5]. Suppose that $\delta^\theta=0$ in (38). The iterative scheme of the duality-based approach subsumes two steps iteratively solving the following objective:

$$\max_{\pi} \min_{\lambda \geq 0} \left(V^\pi(b_0; \boldsymbol{\rho}_1) - V^\pi(b_0; \boldsymbol{\theta}_0) \right) \tag{41}$$

where one step minimizes for λ and another maximizes for execution stochastic policy π. Here, $\boldsymbol{\theta}_0$ is a vector of cost operators (starting from time 0). The dual ascend goes toward $V^\pi(b_0; \boldsymbol{\theta}_0)=0$. The policy is feasible only in this case. In the λ minimization step, since $V^\pi(b_0; \boldsymbol{\theta}_0) \geq 0$, the larger λ will yield a smaller objective. Thus, this part of the objective is becoming increasingly important with the iterations of the step of the minimization of λ. In practice, (41) is approximated by the BMCTS-DPW estimator for the frozen λ. Many different suboptimal actions participate within every $\hat{Q}$ in the search tree. This is a direct result of the exploration–exploitation tradeoff portrayed by (7) and the assembling of each $\hat{Q}$ from the laces (e.g., if the search tree rooted at b_0, then the corresponding action value is $\hat{Q}(b_0 a_0) = {}^1/_{n(b_0 a_0)} \sum_j q(b_{0:L}^j)$). Another possibility would be on the way up the tree to take the maximum of the previously calculated $\hat{Q}(b(h), a)$ with respect to actions with visitation count $n(ha)>0$. We need to exclude the actions with $n(ha)=0$ in order not to take the initial values q^{init} [see Fig. 2(b)]. If we do that, on the way up the tree, instead of completing the lace with future cumulative reward, we will complete it with the result of the maximum. Still, the problem of many actions participating in the $\hat{Q}$ remains. Because the result of the maximum is changing as MCTS progresses, still suboptimal actions are participating in each $\hat{Q}$ besides the leaves. This aspect is detrimental to online planning under the safety constraint. If the safety is formulated as in (38) and (39) with $\delta^\theta=0$ and the objective is (41), then the robot will prefer to depart from unsafe regions as far as possible to ensure that the all expanded actions with at least a single posterior are safe at as many beliefs as possible in the search tree. The importance of all the actions being safe increases closer to the root because closer to the root actions participate within more laces. Our approach does not suffer from such a problem since we prune the dangerous actions in the first place.

X. Simulations and Results

We are now eager to demonstrate our findings in simulations. First, note that upholding the two sets of beliefs is not mandatory in our method. An alternative approach would be maintaining a single set of beliefs defined by (1) and (2), which are unsafe in general. We name such a variant of our approach PC-PFT-DPW. Let us remind the difference in the inner constraint as such. In PC-PFT-DPW, the payoff operator is defined by (12) and (13), whereas in PC-SB-PFT-DPW (we added SB to indicate "safe beliefs")

$$\begin{aligned}\phi(\bar{b}_\ell) &= \mathrm{P}\left(\{x_\ell \in \mathcal{X}_\ell^{\text{safe}}\} \middle| \bar{b}_\ell\right) \\ &= \mathrm{P}\left(\{x_\ell \in \mathcal{X}_\ell^{\text{safe}}\} \middle| b_0, a_{0:\ell-1}, z_{1:\ell}, \bigcap_{i=0}^{\ell-1} \{x_i \in \mathcal{X}_i^{\text{safe}}\}\right)\end{aligned} \tag{42}$$

$$\begin{aligned}\phi(\bar{b}_\ell^-) &= \mathrm{P}\left(\{x_\ell \in \mathcal{X}_\ell^{\text{safe}}\} \middle| \bar{b}_\ell^-\right) = \\ &\mathrm{P}\left(\{x_\ell \in \mathcal{X}_\ell^{\text{safe}}\} \middle| b_0, a_{0:\ell-1}, z_{1:\ell-1}, \bigcap_{i=0}^{\ell-1} \{x_i \in \mathcal{X}_i^{\text{safe}}\}\right).\end{aligned} \tag{43}$$

We always simulate the trials of several AL cycles. A single cycle is depicted in Fig. 3(b). Within PF, we parallelized passing each particle through motion model T and calculating the weights. For resampling, we use a low-variance resampler [23].

A. Problems Composition

We now specify our problems under consideration.

a) Safe lidar roomba: Roomba is a robotic vacuum cleaner that attempts to localize itself in a familiar room and reach the target region. The POMDP state is the position of the agent x, its orientation angle θ, and the status. The status is a binary variable and it tells whether the robot has reached the goal state or stairs. The Roomba action space is defined as $\mathcal{A} \triangleq \{a^1, a^2, a^3, a^4, a^5, a^6\}$. Each Roomba action is a pair (v, ω^v). It comprises a velocity v and a corresponding angular velocity $\omega^v = {}^{\mathrm{d}\theta}/_{\mathrm{d}t}$. We discretized the velocities and the angular velocities such that our actions are $a^1=(0,-\pi/2)$, $a^2=(0,0)$, $a^3=(0,\pi/2)$, $a^4=(5,-\pi/2)$, $a^5=(5,0)$, and $a^6=(5,\pi/2)$. We also have $v^{\text{noise\_coeff}}=0.2$ and $\omega^{\text{noise\_coeff}}=0.05$ such that $v^{\max}=5+0.5\cdot v^{\text{noise\_coeff}}$ and $\omega^{\max}=\pi/2+0.5\cdot\omega^{\text{noise\_coeff}}$. In our simulations, we selected $\mathrm{d}t=0.5$ s. We set $\sigma_{\text{ray}}=0.01$, $\text{rl}_{\min}=0.001$, the stairs penalty is -10000, the goal reward is 10 000, and the time penalty is -1000. The robot motion is deterministic with a predefined time step $\mathrm{d}t$, but the action is noisy. When we apply PF, each particle is propagated with a noisy action. The velocity noise is drawn from a uniform distribution over the interval $(-0.5v^{\text{noise\_coeff}}, 0.5v^{\text{noise\_coeff}})$. Similarly, the angular velocity noise is uniform over the interval $(-0.5\omega^{\text{noise\_coeff}}, 0.5\omega^{\text{noise\_coeff}})$. We draw the noise for each particle and add to the action from $\mathcal{A}$ before we apply the motion model. To do so, we first clamp velocity v in the interval $[0, v^{\max}]$. We then clamp ω^v in the interval $[-\omega^{\max}, \omega^{\max}]$. The next $\theta'=\theta+\omega^v\cdot\mathrm{d}t$ is wrapped to the interval $(-\pi, \pi]$. After the turn, next position of the agent is $x'=x+v\cdot\mathrm{d}t\cdot(\cos(\theta), \sin(\theta))^T$. If the robot hits the wall, then it stops. The status becomes 1 if the robot hits the goal wall [green color in Fig. 6(a)] and -1 if the robot hits a stairs wall [red color in Fig. 6(a)]. At the end of the motion step, the status is updated and the agent takes an observation. It first determines the ray length rl using the known workspace

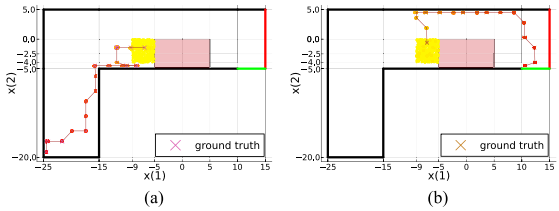


Fig. 6. Plot of one of the trials of the execution of the actions from planning in Lidar Roomba problem. Illustration of departing from the unsafe region problem in Lagrangian based methods. The yellow rectangle represents 500 particles of b_0 sampled from a uniform distribution, and the pink rectangle is the unsafe region to avoid. The green line is the exit area and the red line is the stairs. On both figures, we plotted the ground truth robot positions and the beliefs that transit from yellow to red as time indexes progress. (a) CPFT departs as far as possible from the unsafe region. (b) Our method behaves as expected: the agent goes to the green area while avoiding the unsafe region.

(room) and the position and heading direction $(\cos(\theta), \sin(\theta))^T$ of the robot. The distribution of the observation conditioned on the robot pose is then Gaussian $\mathcal{N}(\text{rl}, \sigma(\text{rl}))$ truncated from the left at zero, where $\sigma(\text{rl}) = \sigma_{\text{ray}} \max(\text{rl}, \text{rl}_{\min})$. To introduce the safety aspect, similar to [6], we add a rectangular avoid region (see Fig. 6). The reward is the expectation over the state reward that is a large reward for reaching the goal, large penalty for reaching the stairs and for each time instance.

b) Dangerous Light Dark: We take inspiration from the problem described in [5]. The agent lives in a 1-D space. We reach versatility of action space by the length of actions, such that $\mathcal{A} \triangleq \{0, \pm 0.5, \pm 1, \pm 1.5, \pm 2, \pm 2.5, \pm 6\}$. The agent's reward is the multiobjective and subsumes the expected state-dependent reward and the belief-dependent reward to localize itself $\rho_{\ell+1}(b_\ell, a_\ell, b_{\ell+1}) = \mathbb{E}_{x_\ell \sim b_\ell}[r(x_\ell, a_\ell)] - \text{tr}(\Sigma(b_{\ell+1}))$, where $\Sigma(b_{\ell+1})$ is the covariance matrix of $b_{\ell+1}$. The agent's state-dependent goal is to get to the location defined by the interval $[-0.75, 0.75]$ as fast as possible and execute the action 0 to stay there. Executing it within the interval $[-0.75, 0.75]$ will give the agent a reward of 100, and executing it outside the radius will yield a negative reward of -100. For all other actions, the state-dependent reward function is $-\text{abs}(x)$. The agent's motion model T is specified as

$$x_{\ell+1} = x_\ell + a_\ell + w_\ell \tag{44}$$

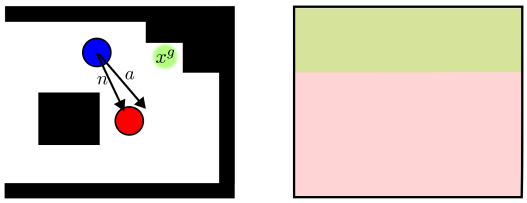
where w_ℓ follows truncated Gaussian $\mathcal{N}(0, \sigma)$ with $\sigma = 0.1$ and truncation with $\Delta = \pm 0.5$ around zero. The light region is located at $x = 2$ and the observation model is $z_\ell = x_\ell + v_\ell$, where $v_\ell \sim \mathcal{N}(0, \sigma(x_\ell))$ and $\sigma(x) = \mathbf{1}_{\{x:|x-2|\leq 1\}}(x) \cdot 10^{-10} + \mathbf{1}_{\{x:|x-2|>1\}}(x) \cdot |x-2|$. At $x = -0.75$, there is a cliff such that if the agent falls, then it crashes. In addition, around the light source, there is a pit. The safe space is $\mathcal{X}^{\text{safe}} = \{-0.75 < x < 1\} \cap \{x > 3\}$. The prior belief $b_0(x_0)$ is Gaussian $\mathcal{N}(7, 20)$ truncated such that its support is $[6, 8]$.

c) Simultaneous Localization and Mapping (SLAM) with Certain and Uncertain Obstacles: Our action space comprises motion primitives and zero action, $\mathcal{A} = \{\rightarrow, \nearrow, \uparrow, \nwarrow, \leftarrow, \swarrow, \downarrow, \searrow, \mathbf{0}\}$. If robot selected zero action $\mathbf{0}$, we do not apply motion model to each particles but do resampling to take into account the received observation. This allows the robot to not move if it is too dangerous. In this problem, the agent and the uncertain obstacles (landmarks) have circular form. The motion model T for the agent is

$$x_{\ell+1} = x_\ell + a_\ell + w_\ell \tag{45}$$

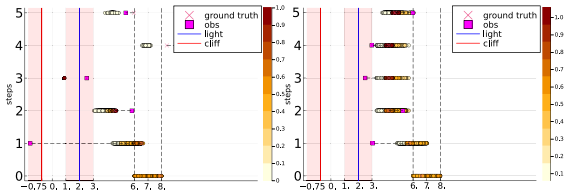
where $w_\ell \sim \mathcal{N}(0, 0.05)$ is truncated at $\Delta = \pm 0.5$. Our goal is to epitomize the importance of safe state trajectories (making beliefs safe) versus solely safe beliefs trajectory. Toward this end, we draw randomly many tiny obstacles. Our observation model is bearing range with the noise inversely proportional to the distance to the uncertain obstacle, the landmark l. The motion model for the landmark is $l_{\ell+1} = l_\ell$. We maintain belief over the last robot pose and the landmark. The observation model reads $z_\ell = x_\ell - l_\ell + v_\ell$, where $v_\ell \sim \mathcal{N}(0, \Sigma(x_\ell - l_\ell))$. $\Sigma(x_\ell - l_\ell)$ is a diagonal matrix with the main diagonal filled with $\sigma^2(x_\ell - l_\ell)) = \|x_\ell - l_\ell\|_2$. The reward is $\rho_\ell(b_\ell) = -\mathbb{E}_{x_\ell \sim b_\ell}[\|x_\ell - x^g\|_2] - \text{tr}(\Sigma(b_\ell))$.

2) Pushbox2D Problem: In this section, we first describe our variation to continuous domains in terms of observations of the PushBox2D problem with soft safety presented in [24]. We then transfer the soft safety to our formulation described in Section III. Clearly, soft safety is not good enough. In cases where there is no feasible solution, we do not want the robot to do any operations. It is desirable that the robot decides that the goal is not achievable. With soft safety, this is not possible. For the sake of completeness, we turn now to the problem description, closely following [24] with our mere extension to continuous domains in terms of observations. The original version was continuous in terms of states and actions. A disk-shaped robot (blue disk) must bump against a disk-shaped puck (red disk) to push it into a goal area (green circle) while avoiding colliding of itself and the puck with the black regions. The state space consists of the xy-locations of both the robot and the puck, i.e., $\mathcal{X} = \mathbb{R}^4$, whereas the action space is defined by motion primitives of unit length. The action Null is terminal. If the robot does not make contact with the puck during a move, then the POMDP state progresses as follows: $x' = (f(x, a) + w, (x^p, y^p))^T, w \sim \mathcal{N}(0, W), f(x, a) = (x^r + a^x, y^r + a^y)^T$, where (x^r, y^r) and (x^p, y^p) represent the robot and puck's xy coordinates for state x, while (a^x, a^y) embody the displacement vector for action a. If the robot collides with the puck while applying action a, then the puck moves from (x^p, y^p) to (x'^p, y'^p) in accord to $\binom{x'^p}{y'^p} = \binom{x^p}{y^p} + 5r_s \left(\binom{a^x}{a^y} \cdot n\right) \left(n + \binom{r^x}{r^y}\right)$, where n is the unit directional vector from the center of the robot to the center of the puck at the moment of contact, and r_s is a random variable drawn from a truncated Gaussian distribution $\mathcal{N}(\mu, \sigma^2, l, u)$, which is the Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$ truncated to the interval $[l, u]$, where $\mu = 2$, $\sigma^2 = 0.001$, $l = 1.99$, and $u = 2.02$. The variables r_x and r_y are random variables drawn from a truncated Gaussian distribution $\mathcal{N}(0.0001, 0.00001, 0, 0.0002)$. The prior belief $b_0(x_0)$ is a Gaussian over the robot position and deterministic over the puck position. The robot is equipped with a noisy bearing sensor for localizing itself relative to the puck, and a noise-free collision sensor that detects contacts between the robot and the puck. Specifically, given a state $x \in \mathcal{X}$, an observation $z \triangleq (o_c, o_b)$ consists of a binary component o_c, which indicates whether or not a contact between the robot and the puck occurred, and a bearing range component o^{br} calculated as



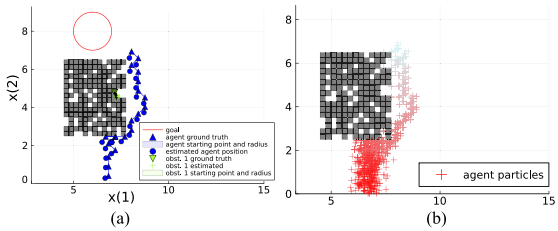
(a) (b)

Fig. 7. (a) Conceptual visualization of the PushBox2D problem. The agent is the blue circle. The puck is the red circle. The goal is the green circle. (b)Observation noise intensity map. Light green color denotes the lower noise intensity.



(a) (b)

Fig. 8. Faulty scenario of CPFT as opposed to our approach. (a) CPFT. (b) Our Algorithm 3.



(a) (b)

Fig. 9. Simulation setup is associated with Table III. Here, we plot one of the trials given in Table III. (a) Goal, agent ground truth, estimated agent positions, and the obstacles. (b) Belief particles with the colors symbolizing the time instant.

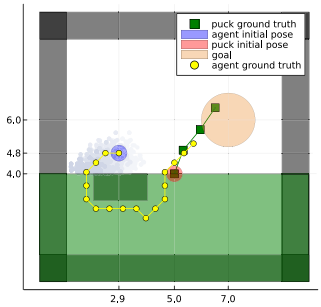


Fig. 10. PushBox2D simulation, δ=0.7.

$o^{\text{br}}=h(x)+v(x)$, $h(x)=(x^r-x^p, y^r-y^p)^T$, where x^r and y^r, and x^p and y^p are the xy-coordinates of the robot and the puck corresponding to the state $x=(x^r x^p, y^r y^p)^T$, respectively. The noise $v(x)$ follows Gaussian distribution with the magnitude of the variance dependent on the position on the map of the robot, as in Fig. 7(b). The reward for the MCTS baseline is the distance to goal of the puck with the boundary region and other obstacles

$$\rho(z,b)=-\mathbb{E}_{x\sim b}[\|x^p-x^g\|_2]-1000\cdot\mathbf{1}_{\{o_c==1\}}(o_c) + s\mathrm{P}(\{x\in\mathcal{X}^{\text{safe}}\}|b)-\mathrm{tr}(\Sigma(b)). \quad (46)$$

TABLE I
LIDAR ROOMBA PROBLEM WITH $n_x=500$ BELIEF PARTICLES

Model	tree queries	$\hat{V}^*(b_0;\boldsymbol{\rho}_1)$	$\hat{\mathbb{E}}[\|x^t-x^g\|_2^2]\pm$std	$\hat{\mathrm{P}}(S\|b_0)$	$n_{\text{coll}}/n_{\text{trials}}$
CPFT-DPW	1000	-46500.0 ± 136.31	829.78 ± 525.88	1	0/70
PCPFT-DPW	1000	-28086 ± 143.99	56.86 ± 215.12	1	0/70

TABLE II
LIGHT DARK PROBLEM WITH n_x=500 BELIEF PARTICLES

Model	tree queries	$\hat{V}^*(b_0;\boldsymbol{\rho}_1)$	$\hat{\mathrm{P}}(S\|b_0)$	$n_{\text{coll}}/n_{\text{trials}}$
CPFT-DPW	15	-75.67 ± 57.66	0.77	16/70
PCPFT-DPW	15	-115.27 ± 94.28	1	0/70

TABLE III
FIFTY TRIALS OF AT MOST 20 CYCLES OF AL WHERE PLANNING SESSIONS IMPLEMENTED BY ALGORITHM PC-SB-PFT-DPW

Alg.	$\hat{\mathrm{P}}(S\|b_0)$	n_{coll}	mean cum. rew. $\pm$ std
PC-SB-PFT-DPW	0.64	18/50	-106.37 ± 12.37

The PC-PFT-DPW crashed in some trial due to empty action space at some node within search tree constructed in planning session. Same seed in both algorithms. This problem is the SLAM described in Section X-Ac) in scenario shown in Fig. 9. The inner threshold $\delta=0.8$. The rollout is ON.

Here, we have a soft safety emphasized by the red color. It is not clear how to select safety importance parameter s. In the case of our approach, we shift $\mathrm{P}(\{x\in\mathcal{X}^{\text{safe}}\}|b)$ to our PC.

B. Experiments

We benchmarked our approach using the Lidar Roomba problem, the famous Light Dark problem, active SLAM problem, and PushBox2D problem. We have shown the issue described in Section IX-B on Lidar Roomba and the satisfiability of the constraint solely at the limit of BMCTS-DPW convergence situation on a Light Dark problem. We compare our approach (Algorithm 3 with CPFT-DPW suggested in [5]) with our modifications in terms of constraining propagated beliefs as described in Section IX-A. For our approach named PC-PFT-DPW, we select the payoff operators ϕ as in (12) and (13) and simulate for δ=1. Our baseline follows the averaged constraint formulation in the cost form as in (38) with cost operator as in (40), and payoff operators ϕ and δ inside (40) identical to the one used in PC-PFT-DPW. As described in Section IX-A, we set δ^θ=0. In PC-PFT-DPW, we use our safe rollout (see Algorithm 4), whereas in CPFT-DPW, the rollout is set per problem. Recall that in case of δ=1, maintaining two sets of beliefs is redundant, and operators (12) and (13) overlap with (42) and (43). Considering an active SLAM problem, we visualized the importance of making belief safe in planning, as described in Section IV-A. With PushBox2D problem, we verified the importance of constraining the propagated belief and simulated for several values of δ and compared our approach with classical PFT-DPW with soft safety portrayed by the red element in (46). In the Lidar Roomba problem, the robot performs at most 50 cycles of AL. In the Light Dark problem, the robot performs five cycles of AL. We do 70 trials of each such scenario and approximate $\mathrm{P}(\tau_0^{\text{gt}}\in S|b_0,\pi^{\text{planner}})\approx\hat{\mathrm{P}}(S|b_0,\pi^{\text{planner}})=\sum_{i=1}^{70}\mathbf{1}_S(\tau_0^{\text{gt},i})/70$ using the simulated trajectories. The event $S=\{\tau_0:$

TABLE IV
TWENTY TRIALS OF AT MOST 20 AL CYCLES OF ALGORITHM PC-SB-PFT-DPW VERSUS PC-PFT-DPW

Parameters				$\hat{\mathrm{P}}(S\|b_0,\pi^{\text{planner}})$ in accord to (47)		Est. prob. of puck reaching the goal		mean cum. rew. $\pm$ std	
n_x	δ	propagated constr.	rollout	PC-SB-PFT-DPW	PC-PFT-DPW	PC-SB-PFT-DPW	PC-PFT-DPW	PC-SB-PFT-DPW	PC-PFT-DPW
500	0.0	Yes	Yes	0.1	0.0	1.0	1.0	-20.39 ± 6.12	-20.73 ± 4.32
500	0.3	Yes	Yes	0.45	0.3	0.85	0.85	-37.33 ± 11.93	-32.31 ± 11.58
500	0.7	Yes	Yes	0.65	0.7	0.65	0.8	-43.04 ± 6.82	-41.56 ± 9.31
500	0.7	No	Yes	0.2	0.3	0.9	0.9	-38.38 ± 9.69	-34.07 ± 9.20
500	0.7	Yes	No	0.65	0.85	0.55	0.6	-46.71 ± 7.71	-44.26 ± 8.81
1000	0.7	Yes	No	0.65	0.5	0.65	0.7	-39.09 ± 10.88	-42.68 ± 8.32
500	1.0	Yes	Yes	-	1.0	-	0.0	-	-60.18 ± 0.18
500	1.0	No	Yes	-	0.2	-	0.9	-	-35.81 ± 7.59

Same seed in both algorithms. This problem is the PushBox2D described in Section X-A1. When the rollout is switched OFF, we shuffle $\mathcal{A}$. Each planning session the planner does 1500 tree queries. The horizon is $L = 20$.

TABLE V
TWENTY TRIALS OF AT MOST 20 AL CYCLES OF ALGORITHM PC-SB-PFT-DPW VERSUS PC-PFT-DPW AND **PUSHBOX2D** PROBLEM

	n_x	planning time [s]	Rollout
PC-SB-PFT-DPW	500	110.12 ± 27.15	Yes
PC-PFT-DPW	500	95.10 ± 33.91	Yes
PC-SB-PFT-DPW	500	39.35 ± 12.64	No
PC-PFT-DPW	500	23.27 ± 9.92	No
PC-SB-PFT-DPW	1000	65.39 ± 31.49	No
PC-PFT-DPW	1000	51.37 ± 17.62	No

	s	planning time[s]	$\hat{\mathrm{P}}(S\|b_0)$	Est. prob of puck reaching the goal	mean cum.rew. $\pm$ std	Rollout
PC-SB-PFT-DPW	-	110.12 ± 27.15	0.65	0.65	-43.04 ± 6.82	Yes
PFT-DPW	1000	26.67 ± 0.47	1.0	0.0	19939.45 ± 1.34	Yes
PFT-DPW	1000	22.48 ± 0.40	1.0	0.0	19939.81 ± 0.19	No
PFT-DPW	500	22.38 ± 0.34	1.0	0.0	9939.82 ± 0.18	No

Same seed in both algorithms, 1500 tree queries, δ=0.7, L=20, propagated belief constraint is activated. At each trial, we calculate averaged across AL cycles planning time of single session. Using these 20 values, we calculate averaged planning time and std.

$\tau_0 \in \times_{\ell=0}^{L} \mathcal{X}_\ell^{\text{safe}}\}$, where $\tau_0 = x_{0:L}$ means each state in the actual robot trajectory starting at time 0 was safe. $\hat{V}^*(b_0;\boldsymbol{\rho}_1) = \frac{1}{70}\sum_{i=1}^{70}\sum_{\ell=0}^{L(i)} \rho_{\ell+1}(b_\ell^i, a_\ell^i, b_{\ell+1}^i)$, where in Roomba $L(i) \leq 50$ since we have a terminal state and in Light Dark $L(i) \equiv 5$. In the SLAM problem, the robot makes 50 trials of at most 20 cycles of AL. In the PushBox2D problem, the robot performs 20 trials of at most 20 cycles of AL.

C. Discussion and Results Interpretation

Before we interpret the results, note that the number of collisions and the approximated probability that the trajectory is safe in the relevant tables are connected as follows:

$$\hat{\mathrm{P}}(\tau_0 \in \times_{\ell=0}^{L} \mathcal{X}_\ell^{\text{safe}} | b_0, \pi^{\text{planner}}) = \hat{\mathrm{P}}(S|b_0, \pi^{\text{planner}}) = 1 - \frac{n_{\text{coll}}}{n_{\text{trials}}} \tag{47}$$

where n_{coll} is the number of collisions in n_{trials}.

a) Roomba: Table I corresponds to the Roomba problem. From Table I, we behold that the cumulative reward yielded by CPFT is much lower than our method. Using CPFT, the Roomba never reaches the goal and not stairs as opposed to our PC-PFT-DPW. We also calculate an empirical mean of the distance between the terminal Roomba position and the middle of the goal region. As we see, in stark contrast to our approach, CPFT makes Roomba depart from the obstacle as far as possible. See Fig. 6.

b) Light Dark: Table II corresponds to the Light Dark problem. In Table II, we see that with a small number of MCTS iterations, CPFT makes 16 collisions from 70 trials in contrast to 0 collisions with our technique. We illustrate the scenario in Fig. 8. In this problem, it is dangerous for the agent to jump to the desired interval. This is because the width of the belief b_0 is larger than the desired area, and the robot can fall off the cliff or into the pit [assuming the motion model as in (44) and without the stochastic noise w_ℓ]. Our approach prevents the robot from jumping to the desired area since any belief particle can be the ground truth.

c) Simultaneous Localization and Mapping: For the SLAM problem, we study the influence of making posterior belief safe before pushing forward in time to evaluate the payoff operator (see line 3 of Algorithms 3 and 6). Making posterior belief safe in time instance ℓ means that, in this time instance, the robot knows that it is online and operational. Table III presents the results for SLAM in our setup with tiny obstacles. We have a rectangular area where we randomly sow tiny rectangular obstacles without replacement. It means if we randomly sow the number of tiny obstacles equal to the number of cells within the large rectangle, then we will obtain a complete large rectangle. We randomly sow the tiny obstacles in each trial. With drawing 80% from a full rectangle of tiny obstacles (see Fig. 9), the PC-PFT-DPW, without making belief safe and maintaining a pair of the beliefs, the scenario in Fig. 9 reached the belief node where *all the actions were claimed unsafe and pruned*, even the **0** action. As we have seen in the simulation, **0** action was pruned the last (no shuffling of $\mathcal{A}$ has been done), and this is a direct result of the fact that unsafe belief particles were propagated with **0** action and updated with received observation. When we do the same operation previously making belief safe, we obtain again the safe belief since particles were propagated with **0** action and, therefore, stay at the same places. Our prior b_0 in SLAM problem is Gaussian with diagonal variances of 0.1.

d) PushBox2D: We constructed a challenging scenario where evading the obstacle significantly complicates putting the puck into the hole (the goal). Please see Fig. 10 for visualization of single trial. We presented the results in Tables IV and V. We selected m=10 and ϵ=0 in rollout (see Algorithm 4). Constraining the propagated belief significantly improves safety while preserving reaching the goal by the puck. We report time spent on maintaining the second set of beliefs defined by (24) and (25) in Table V. We also compare our approach with soft safety highlighted by the red color in (46). In Table V, we behold that maintaining two sets of beliefs only slightly increases the running time (maximum by a factor of two). The soft safety

yields the departure of the agent as far as possible from the obstacle due to the participation of the dangerous actions in the objectives within the search tree.

XI. Conclusion

We introduced an anytime online approach to perform safe and risk aware belief space planning in continuous domains in terms of states, actions, and observations. We rigorously analyzed our approach in terms of convergence. Our prominent novelty is assuring safety with respect to the belief tree expanded so far. As opposed to SOTA in continuous domains, we are not mixing safe and dangerous actions in the search tree. Our belief tree is safe with respect to our PC and consists solely of the safe actions. Moreover, when our PC is satisfied, it is satisfied starting from each belief action node, ensuring a match in a current planning session and future planning sessions. We corroborated our theoretical development by simulating *four* problems in continuous domains. Each problem exhibited a different phenomenon caught by our methodology.

Acknowledgment

The authors would like to thank Ron Benchetrit for fruitful conversations.

References

[1] O. Madani, S. Hanks, and A. Condon, "On the undecidability of probabilistic planning and related stochastic optimization problems," *Artif. Intell.*, vol. 147, no. 1-2, pp. 5–34, 2003.
[2] P. Santana, S. Thiébaux, and B. Williams, "RAO*: An algorithm for chance-constrained POMDP's," in *Proc. AAAI Conf. Artif. Intell.*, 2016.
[3] A. Zhitnikov and V. Indelman, "Risk aware adaptive belief-dependent probabilistically constrained continuous POMDP planning," 2022, *arXiv:2209.02679*.
[4] A. Zhitnikov and V. Indelman, "Simplified continuous high dimensional belief space planning with adaptive probabilistic belief-dependent constraints," *IEEE Trans. Robot.*, vol. 40, pp. 1684–1705, 2024.
[5] A. Jamgochian, A. Corso, and M. J. Kochenderfer, "Online planning for constrained POMDPs with continuous spaces through dual ascent," in *Proc. Int. Conf. Automated Plan. Scheduling*, 2023, pp. 198–202.
[6] A. Jamgochian, H. Buurmeijer, K. H. Wray, A. Corso, and M. J. Kochenderfer, "Constrained hierarchical Monte Carlo belief-state planning," 2023, *arXiv:2310.20054*.
[7] J. Lee, G.-H. Kim, P. Poupart, and K.-E. Kim, "Monte-Carlo tree search for constrained POMDPs," in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 7934–7943.
[8] E. Altman, *Constrained Markov Decision Processes*. Boca Raton, FL, USA: CRC Press, 1999.
[9] A. Beck, *First-Order Methods in Optimization*. Philadelphia, PA, USA: SIAM, 2017.
[10] Q. H. Ho et al., "Recursively-constrained partially observable Markov decision processes," 2023, *arXiv:2310.09688*.
[11] R. Munos, "From bandits to Monte-Carlo tree search: The optimistic principle applied to optimization and planning," 2014.
[12] M. Ajdarów, Š. Brlej, and P. Novotn, "Shielding in resource-constrained goal POMDPs," in *Proc. AAAI Conf. Artif. Intell.*, 2023, pp. 14674–14682.
[13] G. Mazzi, A. Castellini, and A. Farinelli, "Risk-aware shielding of partially observable Monte Carlo planning policies," *Artif. Intell.*, vol. 324, 2023, Art. no. 103987.
[14] D. Silver and J. Veness, "Monte-Carlo planning in large POMDPs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2010, pp. 2164–2172.
[15] R. J. Moss, A. Jamgochian, J. Fischer, A. Corso, and M. J. Kochenderfer, "ConstrainedZero: Chance-constrained POMDP planning using learned probabilistic failure surrogates and adaptive safety constraints," 2024, *arXiv:2405.00644*.
[16] G. Chou, N. Ozay, and D. Berenson, "Safe output feedback motion planning from images via learned perception modules and contraction theory," in *Proc. Int. Workshop Algorithmic Found. Robot.*, Springer, 2022, pp. 349–367.
[17] S. Dean, A. Taylor, R. Cosner, B. Recht, and A. Ames, "Guaranteeing safety of learned perception modules via measurement-robust control barrier functions," in *Proc. Conf. Robot Learn.*, 2021, pp. 654–670.
[18] D. Auger, A. Couetoux, and O. Teytaud, "Continuous upper confidence trees with polynomial exploration–consistency," in *Proc. Mach. Learn. Knowl. Discov. Databases: Eur. Conf.*, ECML PKDD 2013, Prague, Czech Republic, Sep. 23-27, 2013, Proc., Part I 13, 2013, pp. 194–209.
[19] Z. Sunberg and M. Kochenderfer, "Online algorithms for POMDPs with continuous state, action, and observation spaces," in *Proc. Int. Conf. Automated Plan. Scheduling*, 2018, pp. 259–263.
[20] L. Kocsis and C. Szepesvári, "Bandit based Monte-Carlo planning," in *Proc. Eur. Conf. Mach. Learn.*, 2006, pp. 282–293.
[21] A. Zhitnikov and V. Indelman, "Simplified risk aware decision making with belief dependent rewards in partially observable domains," *Artif. Intell.*, vol. 312, 2022, Art. no. 103775.
[22] M. H. Lim, T. J. Becker, M. J. Kochenderfer, C. J. Tomlin, and Z. N. Sunberg, "Optimality guarantees for particle belief approximation of POMDPs," *J. Artif. Intell. Res.*, vol. 77, pp. 1591–1636, 2023.
[23] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Cambridge, MA, USA: MIT Press, 2005.
[24] M. Hoerger, H. Kurniawati, D. Kroese, and N. Ye, "Adaptive discretization using Voronoi trees for continuous POMDPs," *Int. J. Robot. Res.*, vol. 43, no. 9, pp. 1283–1298, 2024.

Andrey Zhitnikov received the B.Sc. degree in electrical engineering from the School of Electrical Engineering, Tel Aviv University, Tel Aviv, Israel, in 2014, the M.Sc. degree in electrical and computer engineering from Technion, Haifa, Israel, in 2018, and the Ph.D. degree in robotics from Technion Autonomous Systems Program, Haifa, in 2024.

Previously, he was a one-year Postdoctoral Fellow with the Autonomous Navigation and Perception Lab, Technion. He is currently a Postdoctoral Scholar with the Multi-Robot Systems Lab, Technion. His current research interests include planning under uncertainty in the robotics context, robot safety under uncertainty, kinodynamic motion planning, and game-theoretic multirobot planning.

Vadim Indelman (Member, IEEE) received the B.A. degree in computer science and the B.Sc. degree in aerospace engineering, and the Ph.D. degree in aerospace engineering from Technion, Haifa, Israel, in 2002 and 2011, respectively.

Between 2012 and 2014, prior to joining the Technion as a Faculty Member, he was a Postdoctoral Fellow with the Institute of Robotics and Intelligent Machines, Georgia Institute of Technology. He is currently an Associate Professor with the Stephen B. Klein Faculty of Aerospace Engineering and with the Faculty of Data and Decision Sciences (Secondary Appointment), Technion—Israel Institute of Technology. He is also a Member of the Technion Autonomous Systems Program, the Technion Artificial Intelligence Hub (TechAI), and the Israeli Smart Transportation Research Center. In addition, he is a Member of the European Laboratory for Learning and Intelligent Systems. He is currently leading the Robotics vertical at TechAI, which promotes and facilitates research activities and projects within the Technion and collaboration with industry in areas related to AI and robotics. He is the Founder and Head of the Autonomous Navigation and Perception Lab, which performs research related to single- and multirobot autonomous perception, navigation, and planning under uncertainty in the context of mobile robotics. His current research interests include planning under uncertainty, probabilistic inference, semantic perception, and simultaneous localization and mapping in single- and multirobot systems.

Dr. Indelman is an Associate Editor for the *International Journal Of Robotics Research* and *Journal of Artificial Intelligence Research*. Previously, he was an Associate Editor for IEEE Robotics and Automation Letters, an Editor for IEEE/RSJ International Conference on Intelligent Robots and Systems, and a Co-chair of the IEEE Robotics and Automation Society Technical Committee on Algorithms for the Planning and Control of Robot Motion.