Appeared at ICRA 2022 IEEE International Conference on Robotics and Automation
May 23-27 2022 Philadelphia (PA) USA

Failure is an option: Task and Motion Planning with Failing Executions

Tianyang Pan, Andrew M. Wells, Rahul Shome and Lydia E. Kavraki

***Abstract*— Future robotic deployments will require robots to be able to repeatedly solve a variety of tasks in application domains. Task and motion planning addresses complex robotic problems that combine discrete reasoning over states and actions and geometric interactions during action executions. Moving beyond deterministic settings, stochastic actions can be handled by modeling the problem as a Markov Decision Process. The underlying probabilities however are typically hard to model since failures might be caused by hardware imperfections, sensing noise, or physical interactions. We propose a framework to address a task and motion planning setting where actions can fail during execution. To achieve a task goal actions need to be computed and executed despite failures. The robot has to infer which actions are robust and for each new problem effectively choose a solution that reduces expected execution failures. The key idea is to continually recover and refine the underlying beliefs associated with actions across multiple different problems in the domain. Our proposed method can find solutions that reduce the expected number of discrete, executed actions. Results in physics-based simulation indicate that our method outperforms baseline replanning strategies to deal with failing executions.**

I. INTRODUCTION

In order to accomplish complex tasks in real-world scenarios, robots must be capable of choosing among available actions and computing motions for accomplishing a task objective. This is typically solved using Task and Motion Planning (TAMP) [1], [2], [3] which combines discrete, AI planning and geometric reasoning. A robot might be capable of servicing a variety of tasks in a setting. Each time a new task is provided to the robot, a structured specification is given to a TAMP solver module, and the resultant task and motion plan is obtained. This plan consists of a sequence of discrete actions and accompanying motions that can be executed by the robot. As shown in Fig. 1, during execution, an action may fail even if the computed trajectory itself is feasible and collision-free. Such a failure can result from unknown system errors, malfunctioning hardware, unmodeled physical interactions, or sensing uncertainty. Consider the example of a slippery object which might be easier to push than to pick up. While the robot is operating in the scene, it might come across many problems that require interactions with the same object. If certain actions are less robust than other actions, the robot can prioritize task and motion plans that are more likely to be executed successfully. The current work asks the question: can we use the information from failing executions towards efficiently solving a variety of TAMP problem instances in the domain with a minimum expected number of executed actions?

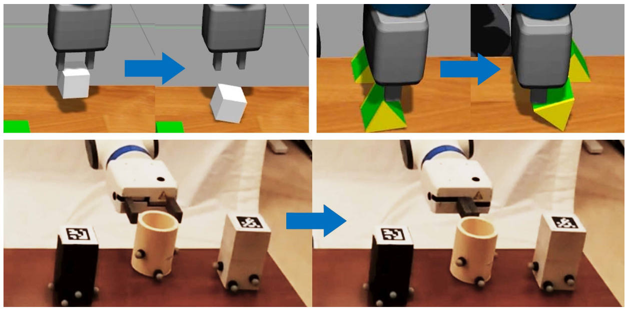


Fig. 1: Instances of actions that can fail during execution. *(top-left):* a pick failure, *(top-right):* a push failure, and *(bottom):* a pick failure in real world.

Since actions can fail during execution, the world cannot be modeled as deterministic. A stochastic version of the task and motion planning domain can be obtained by associating each action with the probability of execution failure. Markov Decision Process (MDP) is a commonly used tool to use as a way to find robust solutions. Some previous work [4] proposes a TAMP framework that is integrated with an MDP solver for stochastic environments where failure probabilities are known. This assumption about knowing the ground truth probabilities is often unrealistic. Consider again the example of a slippery object. Without a perfect model for the physical interactions, the likelihood of success for a pick versus a push cannot be exactly determined. Without any information about the underlying probabilities, the TAMP approach is, for example, incapable of discriminating between a robust push and an unreliable pick for a slippery object. If the specific task can admit either solution, such an uninformed TAMP solver exhibits the highly brittle behavior of computing the same solution again and again. Here the robot will stubbornly retry the failing actions. Since the intrinsic probabilities of the system are unknown, the problem can be formulated as a partially observable Markov Decision Process (POMDP) [5]. POMDPs are generally intractable for large problems, motivating the contribution of the current work - a computationally efficient framework for TAMP with failing executions.

Keeping track of the execution failures and updating the beliefs associated with the actions hold the promise of quickly identifying robust actions and achieving task objectives in fewer overall executed steps. Bayesian inference is a common way to keep incorporating new information to update action beliefs. To achieve good performance, we must balance between exploration and exploitation [6], for which we choose posterior sampling [7], [8]. Our proposed method

The authors are with the Department of Computer Science at Rice University. Emails: {tp36, andrew.wells, rahul.shome, kavraki}@rice.edu. The work was supported in part by NSF 1830549, NSF 2008720, and Rice University Funds. AMW was supported in part by NASA 80NSSC17K0162.

combines an MDP with a Beta-Binomial distribution model [9] and leverages Bayesian inference for updating the belief of the unknown probabilities. When we start with no prior knowledge of a TAMP domain, we initialize an MDP with uniform prior and let the robot execute the optimal policy. The posterior belief is updated given the observations after a failed execution. Then we sample the transition probabilities from the posterior, which can be used to construct an updated MDP. A key contribution of our work is that this information is maintained across all the instances encountered in the domain to aggregate prior knowledge across the domain that can be reused for task and motion planning. Another feature of the method is its general applicability to TAMP domains including high-dimensional robotic arms. Maintaining beliefs and updating the MDP proves efficient when the initial estimate of the model parameters is inaccurate.

The performance of the proposed approach was empirically validated. Experiments were first performed for sets of TAMP problems with manipulation actions with execution failures sampled from simulated ground truths. The proposed approach is empirically reported to be comparable in performance to an optimal baseline (which is aware of ground truth probabilities), and our method is better than replanning or retry-action baselines. A further set of benchmarks are presented in Gazebo [10] using physics-based simulation of the object interactions, where the ground truth is unknown. Our method shows better performance in terms of average number of actions to solve problem instances by correctly differentiating easy to pick objects, and objects where pick or push actions are more reliable. In settings where real-world executions can fail and the underlying probabilities are unknown, our proposed work in modeling beliefs over parameters to a stochastic TAMP solver shows promise and can help to address richer real-world interactions and failures.

II. RELATED WORK

Task and Motion Planning: Previous work [1] in TAMP combines a general satisfiability modulo theories (SMT) solver for task planning and standard motion planners. Some other approaches use heuristic search for high-level planning [2], [3]. TAMP has also been formulated as an optimization problem [11], addressed through guided multi-modal search [12], and extended to multi-robot instances [13]. However, these works usually focus on deterministic and fully-observable environments.

Some works focus on partially observable problems where the robot is uncertain about the current state of the world [14]. The problem is formulated as a POMDP and a regression based algorithm is proposed. A POMDP-based method with online replanning [15] was proposed to gain efficiency. The stochastic TAMP problem can be formulated as an MDP [4], where the authors propose a framework that combines MDP solvers and motion planners. These works need known transition probabilities. There has been some recent interest on closed-loop tasks with real sensing data. These can relax the necessity of exact models but address specific problem instances [16], simplify motion feasibility [17], or rely on learned controllers [18]. In contrast our method provides a general closed-loop TAMP framework to address unknown probabilities of execution failures.

Failure Recovery in Robotics: Extensive research has been carried out on the bin-picking problem [19], [20] where grasping failure is considered. These works usually provide Markov policies for picking-up objects from cluttered environment using deep neural networks. A Markov policy decides which action to take at each state, regardless of the history of how the state is reached. A recent work [21] proposes non-Markov policies for such problems as it leverages the history of execution and observation. These works [19], [20], [21] focus on a specific set of problems, while our framework has the ability to deal with different high-level tasks with abstract specifications. One previous work [22] learns grasping points using beta-binomial distribution based Bayesian techniques. But it does not compute a policy to handle grasping failures. Some works leverage learning from demonstration techniques to enable the robot to recover from errors [23], [24]. Our framework can deal with problems where human prior knowledge is insufficient. A framework is proposed [25] to combine Bayesian inference and deal with failure recovery. However, they consider mobile robots rather than manipulators. Moreover, the above works are mostly restricted to a single policy, while we keep providing new policies as we incorporate new observations and update the MDP model, recovering efficiency even when the initial estimate of the MDP parameters is inaccurate. Recent work [26] learns a feasibility model of task plans to deal with uncertainties in real-world executions. The model parameters are learned via a Bayesian approach. An exploration phase is necessary to collect data and learn the feasibility model, before using it. We do not need a dedicated data collection phase, which is expensive for real robots.

Relation to Reinforcement Learning: Standard reinforcement learning assumes the environment can be modeled as an MDP or POMDP. To avoid the computation complexity of operating in belief space, some approaches leverage posterior sampling to automatically adapt to the change in model estimates [27], [28]. The Bayes-adaptive MDP (BAMDP) is proposed to address uncertainty in the MDP model [29]. BAMDP is mathematically equivalent to POMDP under structural assumptions of the uncertainty in the model [29], [30]. Some recent reinforcement learning work applies policy optimization on BAMDPs to address model uncertainty [31]. However, these works do not tackle problems from TAMP domain. They do not consider motion planning with high DoF robotic manipulators. In addition, they focus on learning to solve a single problem, while we consider a TAMP domain that can have a variety of problem instances. Lastly, the focus of our work is to find a way for the robot to finish tasks as early as possible, rather than learning the whole domain. Reinforcement learning also usually require a large number of interactions between the agent and the environment. The learning procedure is feasible in simulation, but can be inefficient and costly for real robots [32] due to the the well-known “gap” between simulation and reality [33].

III. PROBLEM STATEMENT

A. *Preliminaries*

Motion Planning: Consider a high degree of freedom (DoF) manipulator with multiple objects in the workspace. For a d DoF robot, $\mathcal{C} \subset \mathbb{R}^d$ is its $d-$dimensional configuration space. We denote the free configuration space as $\mathcal{C}_{\text{free}} \subseteq \mathcal{C}$. A trajectory is a time parameterized curve $\pi : [0, t] \rightarrow \mathcal{C}_{\text{free}}$, which is called a solution to a motion planning problem between a start $q_0 \in \mathcal{C}$ and a goal $q_1 \in \mathcal{C}$ if $\pi(0) = q_0, \pi(t) = q_1$. The motion planning domain is D_{MP}.

Task and Motion Planning (TAMP): Task planning is defined over a task planning domain D_{TP} consisting of a finite set of states S, a finite set of actions A that transitions the states. An instance of a TAMP problem includes an initial state s^0 and a set of goal states S_{goal}. Each state-action pair (s, a) is associated with a single result state s^*. TAMP is the problem of finding a feasible n-step task and motion plan $\mathbf{T} = (\langle a^0, \pi^0\rangle, ..., \langle a^n, \pi^n\rangle)$, where $a^0, ..., a^n$ is the task plan that satisfies the task planning domain and successfully transitions to a goal state $s^n \in S_{goal}$. Each discrete action a^i transitions from state s^i to the result state s^{i+1} and corresponds to a feasible motion plan π^i.

When actions executions can fail, i.e., there is a non-zero probability that an action does not reach the desired state, such stochastic TAMP domains can be modeled by an MDP.

Markov Decision Process (MDP): An MDP is a tuple $\mathcal{M} = (S, A, P, R)$,

- S is a finite set of states,
- A is a finite set of actions A_s is the set of actions available from state s,
- $P : S \times A \times S \rightarrow [0, 1]$ is the transition probability function where $\forall s \in S, \forall a \in A_s, \Sigma_{s' \in S} P(s, a, s') = 1$,
- $R : S \times A \times S \rightarrow \mathbb{R}$ is the reward function.

The precise formulation of the MDP in TAMP follows previous work [34], [35]. The state space S and action space A of the MDP belong to the TAMP task planning domain D_{TP}. The transition probability $P(s, a, s')$ denotes the probability of executing action a at state s leading to the state s'. The reward function can be defined based on a notion of cost of such state-action-state tuples.

Path through MDP: A path ξ through an MDP is a state followed by a sequence of action-state pairs:

$$\xi = s^0 \langle a^0 s^1 \rangle ... \langle a^{n-1} s^n \rangle ...$$

ξ is determined by the path in the MDP which has the most cumulative reward from s^0 to a goal state $s^n \in S_{goal}$. This task plan arises from the optimal policy. For solving the TAMP problem, motion planning computes the π^i for each a^i, using this information to update the task domain [1].

In real world problems where executions may fail, the ground truth of the transition probabilities in the MDP are usually unknown. These probabilities can be thought of as parameters that determine the performance of the MDP. We denote Θ as the unknown transition probabilities and P_θ as the transition function parameterized by Θ. When these parameter values are unknown, we have to use probability distributions as beliefs over these parameters Φ. Beta-Binomial distribution model is frequently used in statistics [9], [36], especially in cases where the variable of interest lies in $[0, 1]$.

Beta-Binomial Distribution Model: Assume that the execution of an action either succeeds or fails, with the success probability being θ. Then, if this action is executed L times, the probability of k executions succeeded in the L trials is $\binom{L}{k}\theta^k(1-\theta)^{L-k}$, which follows a Binomial distribution. When the probability θ is unknown, the Bayesian approach to maintain and update the belief of θ is to use the Beta distribution (i.e., $\phi \sim \text{Beta}(\alpha, \beta)$). Since the Beta distribution and the Binomial distribution form a conjugate pair, the posterior after L executions and k successes can be updated [36] by:

$$(\alpha, \beta) \leftarrow (\alpha + k, \beta + L - k)$$

In other words, the number of success and failure can be accumulated to conveniently update the parameters of the Beta distribution in accounting for failing executions.

B. *Problem Formulation*

Let S be the set of all abstract states, A be the set of high-level actions, P_θ be the transition probability function parameterized by the set of parameters Θ, and R be the reward function. $\mathcal{M}_\theta = \{S, A, P_\theta, R\}$ is the MDP parameterized by Θ. We now introduce what parameterization of the MDP we use to model a stochastic TAMP problem.

As we only focus on whether an execution succeeds, for each state-action pair (s, a), we assume there exists only one result state s^* that is defined as the success state as represented in D_{TP}. If the execution of a at s led to any other state that is not s^*, we say that the execution failed. Therefore, the observation model (or likelihood) for a single execution becomes Bernoulli distribution, i.e., the result of the execution of an action a is either success with probability $P_\theta(s, a, s^*)$ or failure with probability $\Sigma_{s' \in S, s' \neq s^*} P_\theta(s, a, s') = 1 - P_\theta(s, a, s^*)$. Denote the success probability to be $P_\theta(s, a, s^*) = \theta_{sas^*}$. An element θ in the parameter space Θ parameterizes P_θ.

Let $\phi^t_{sas^*}$ be the prior distribution of θ_{sas^*} at time step t, which is a Beta distribution $\text{Beta}(\alpha^t_{\theta_{sas^*}}, \beta^t_{\theta_{sas^*}})$. If we execute a from s at time step t for L times repeatedly, the probability of the occurrence of the event where k out of L executions succeeded is $\binom{L}{k}\theta^k_{sas^*}(1-\theta_{sas^*})^{L-k}$, being a Binomial distribution. After $t' = t + L$ executions and k successes, the posterior distribution $\phi^{t'}_{sas^*}$ can be updated by the following rule:

$$(\alpha^{t'}_{\theta_{sas^*}}, \beta^{t'}_{\theta_{sas^*}}) \leftarrow (\alpha^t_{\theta_{sas^*}} + k, \beta^t_{\theta_{sas^*}} + L - k)$$

TAMP with Failing Executions: Given an MDP $\mathcal{M}_\theta = (S, A, P_\theta, R)$ constructed from D_{TP} with unknown transition probabilities Θ, a set of goal states $S_{goal} \subset S$, and the prior distributions Φ of each parameter $\theta \in \Theta$, minimize the expected number of actions executed to solve problem instances in the stochastic task and motion planning domain.

IV. METHOD

We propose the following algorithm that combines MDP and posterior sampling to maintain and update the belief over the unknown transition probabilities.

Algorithm 1: HIGHLEVELLOOP

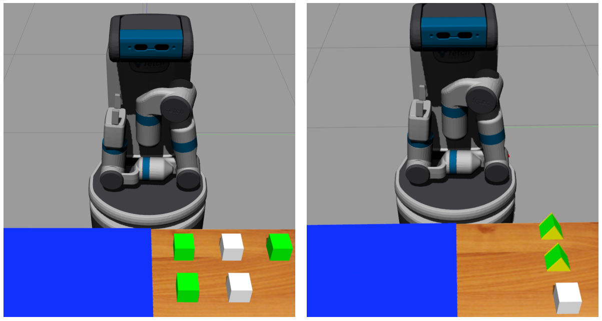
```
   Input: $D_{\mathrm{TP}}, D_{\mathrm{MP}}, \Phi, \mathcal{P}$
   // $\mathcal{P}$ is a sequence of TAMP instances
1  while True do
2      $S, A, P, R, \phi, s_{init}, S_{goal} \leftarrow$ GETNEXTPROBLEM($\mathcal{P}$);   // Get the next incoming problem instance
3      $\mathcal{M} \leftarrow \langle S, A, P, R \rangle$;
4      $\phi \leftarrow$ SOLVEINSTANCE($\mathcal{M}, \phi, s_{init}, S_{goal}, D_{\mathrm{TP}}, D_{\mathrm{MP}}$)
       // Solve & execute the problem instance
5      $\Phi \leftarrow$ BOOKKEEP($\Phi, \phi$);
```

Algorithm 2: SOLVEINSTANCE

```
    Input: $\mathcal{M}, \phi, s_{init}, S_{goal}, D_{\mathrm{TP}}, D_{\mathrm{MP}}$
1   $s \leftarrow s_{init}$;
2   $\psi \leftarrow \emptyset$ ;   // Execution outcome observations
3   while $s \notin S_{goal}$ do
4       $\phi \leftarrow$ UPDATEDISTRIBUTIONS($\phi, \psi$);   // Use action outcomes to update distributions
5       $\Theta \leftarrow$ GETPARAMETERS($\mathcal{M}$);
6       for $\theta \in \Theta$ do
7           $\theta \leftarrow$ BETA($\phi$);   // Sample the transition probabilities from Beta distributions
8       $\mathcal{M}_\theta \leftarrow \langle S, A, P_\theta, R \rangle$;   // MDP parameterized by sampled probabilities
9       $\mathbf{T} \leftarrow$ SOLVETAMP($\mathcal{M}_\theta, s, S_{goal}, D_{\mathrm{TP}}, D_{\mathrm{MP}}$);
10      $s, \psi \leftarrow$ EXECUTEACTIONS($\mathbf{T}$);   // Execute actions till failure and record outcomes
11  return $\phi$;   // Return the updated distributions
```

Given the full TAMP domain, Alg. 1 shows how we maintain the belief of all the transition probabilities of the full domain when we keep solving a sequence of problem instances and update the beliefs. Alg. 2 solves an instance that can be modeled from the domain using an MDP $\mathcal{M}$.

Alg. 1 is responsible for book-keeping of the full TAMP domain. The input to the algorithm is the task planning domain D_{TP}, the motion planning domain D_{MP}, a set of parameters Φ (initialized with prior information) that we use to maintain the distribution of the unknown probabilities, and a set $\mathcal{P}$ of problem instances. Alg. 1 can be seen as a lifelong high-level loop that keeps solving the problem instances within the domain, and updating the belief over the parameters along the way. $\mathcal{P}$ is a sequence of TAMP instances in this domain, specified as a user input when used in real world. The focus is on effectively using information from solving and executing problems in $\mathcal{P}$. Note that if the instances lead to every transition in the MDP being attempted infinitely often, the beliefs will converge. We are instead interested in reducing the expected number of steps executed for each instance, for which exact estimation of the ground truths is not necessary. This is supported empirically. Whenever a new TAMP instance arrives (Alg. 1 line 2, GETNEXTPROBLEM), we use Alg. 2 to solve it (Alg. 1 line 4, SOLVEINSTANCE). After each instance is fully solved and executed, we keep track of the updated distributions (Alg. 1 line 5, BOOKKEEP).

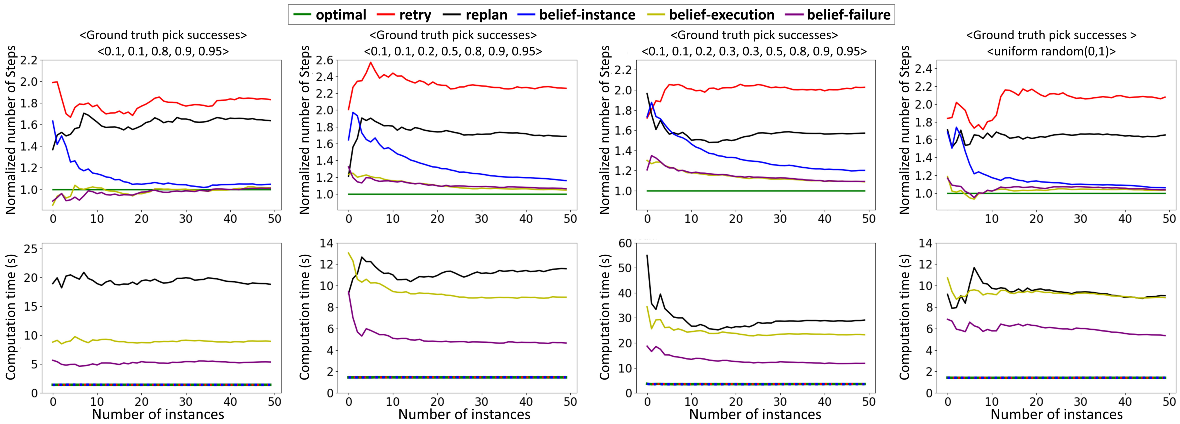


(a) Pick and Place Benchmark: Move n out of 5 objects (b) Pick versus Push Benchmark: Move n out of 3 objects

Fig. 2: The screenshots of the benchmark problem setup in Gazebo showing the initial states. The goal region is marked in blue color. The white objects have much less friction than the green ones, making them difficult to be picked up, but easy to be pushed. The triangular objects in the Pick vs. Push benchmark cannot be pushed to the goal region with a single action as it topples over because of its shape and friction coefficients. The sides of the triangular object are also designed such that picks can fail sometimes.

The input to Alg. 2 is an MDP $\mathcal{M}_\theta$ with unknown transition probabilities P_θ, the initial state, a set of the goal states, and the current belief distribution ϕ of the unknown parameters Θ. The algorithm starts with randomly sampling the transition probabilities from the distribution (Alg. 2 line 7) and builds an MDP (Alg. 2 line 8). We solve the TAMP problem using a stochastic TAMP solver (Alg. 2 line 9, SOLVETAMP), where we use PRISM model checker [37] to solve the MDP, and use RRT-Connect [38] to solve the motion planning problems. Then, the TAMP plan is executed from the current state s, and observations ψ in the form of execution successes or failures are made and collected (Alg. 2 line 10, EXECUTEACTIONS). The distributions are updated using the collected information (Alg. 2 line 4). We then get the set of transition probabilities from the MDP by GETPARAMETERS (Alg. 2 line 5), and sample new transition probabilities from updated distributions (Alg. 2 line 7) to build a new MDP.

Note that we choose to let the robot execute the TAMP plan until a failed execution is observed (Alg. 2 line 10), which means the observation information is collected along the way and updated to the belief together upon failures. Other update strategies could also apply, such as updating after each action, or after the whole task is executed, where motion plans from the failed state can be computed according to the existing policy.



Pick and Place: N = 5 Pick and Place: N =7 Pick and Place: N = 9 Random ground truth: N = 5

Fig. 3: The Pick and Place benchmark is designed for $N = 5, 7$, and 9 objects. One instance of the problem can require any $1 < n < N$ objects to be moved. A sequence of 50 problem instances defines a single run over which the performance of approaches can be measured. 20 such runs are sampled and recorded. The data at the instance $t \in [1 \cdots 50]$ is reported as an average over instances 1 up to t, across 20 runs. (Top:) Number of steps executed to finish each instance normalized over the optimal behavior. (Bottom:) Computation time of each instance. The ground truth probabilities for a successful pick of N objects is shown above each plot. The rightmost column uses randomly sampled ground truths for each of 20 runs.

V. EXPERIMENTS

A. Benchmark Problems

We evaluate the performance of each method in a tabletop domain. It is assumed that the real execution of each picking or pushing action has a success probability. There is a single robot that must finish the given problem instances.

Pick and Place Benchmark: As shown in Fig. 2a, there are N objects on the table, which contains an initial region and a goal region. We assume a region can support all N objects. Initially, all the objects are in the initial region. The available actions are pick and place. A problem instance needs to move n objects to the goal region, where $1 \leq n < N$. To be efficient, less pickable objects should to be avoided.

Pick vs. Push Benchmark: Similarly, we have N objects in the initial region of the table (see Fig. 2b). The goal is to move any n of the objects to the goal region. The available actions are pick, place, and push. An instance of this TAMP domain is the task of moving n objects to the goal region, where $1 \leq n \leq N$. The expected behavior of the proposed method is to distinguish if pushing or picking works better for each object, and to distinguish between objects.

B. Experiment Setting

Non-physics simulation: In this setting, the ground truth of all the probabilities are given to the simulator. The simulator then randomly generates the outcome for each action execution. We keep running the methods for 50 randomly sampled instances from the TAMP domain.

Physics simulator: We use Gazebo with the Open Dynamics Engine (ODE) [39] as a physics simulator and the Robowflex [40] motion planning framework. To simulate the real world in Gazebo we tune the mass and the friction terms of each object to make them have different probabilities of being picked successfully. Note here the ground truth of such probabilities are unknown. To introduce uncertainty in pushing, a triangular object (Fig. 2b) topples when pushed.

C. Methods To Compare

Optimal (Ground truth MDP): This baseline knows the ground truth probabilities. Its performance is therefore the optimal. Note that we can design ground truths for non-physics simulation. The ground truths become unknown when using physics simulator or in the real world.

Retry: This method assumes no prior knowledge of the ground truth of the parameters. Therefore, it assumes the probability of successfully executing any action to be 1. It builds an MDP with such an assumption and keeps executing the policy even if any real execution fails.

Replan: When an execution fails, this method will create a new MDP with the failed transition blocked to make sure a new policy is computed. If no policy can be computed, it will relax all the blocked transitions and start over again. Note that this baseline maintains the information across different problem instances. If it observes that an action fails for one instance, it keeps blocking this action from the state when solving the successive instances.

Belief (Proposed): This is the proposed method shown in Alg. 1 and 2 which updates the belief when an execution fails ("*belief-failure*"). We also compare against alternate frequencies of updating the belief. The "*belief-execution*" update rule will update the belief after each execution during each task. The "*belief-instance*" update rule only updates the belief once after a problem instance is fully completed.

D. Results

Simulation results: Fig. 3 shows the results of the *Pick and Place* benchmark, with increasing total number of objects N in the TAMP domain. The top row shows the normalized number of steps over the result of the ground truth baseline. Note the plot is smoothed so that each data point is averaged from the beginning. From the top row in Fig. 3, we can see that the *retry* baseline reaches the goal state taking twice as many steps as the optimal behavior on average because it is not leveraging the observation. The *replan* baseline has a better performance than the *retry* one, but still far from optimal. Since all the executions have some probability of failure, the *replan* method will block all the actions and has to start over again from time to time. All three variants of the proposed method converge to the optimal behavior eventually, often within only the first few instances.

Some problem instances can require longer solutions than others regardless of the uncertainty. By averaging over runs and instances this noise is smoothed out in the data and a clear trend emerges where the proposed method performs close to the optimal, while *retry* and *replan* are worse.

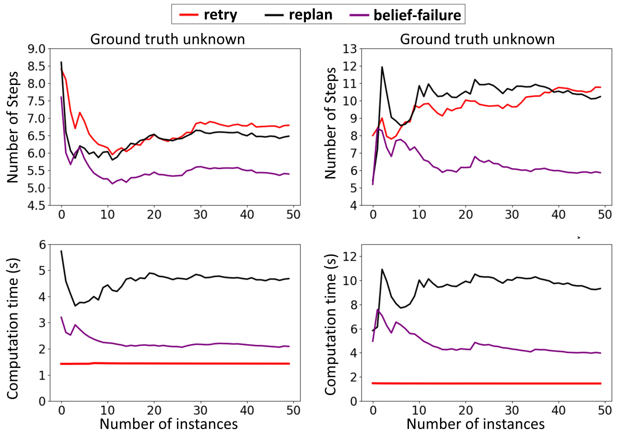
The bottom row in Fig. 3 shows the corresponding computation time per-instance. The *retry* baseline only computes policy once for each task and blindly follow the policy. The *belief-instance* variant of the proposed method also computes policy once per-instance but it updates the belief after an instance is finished. The *replan* baseline needs to compute a new policy whenever it updates the MDP model by blocking an action. From Fig. 3, the proposed *belief-failure* performs much better than the other methods and needs less computation than the *replan* or *belief-execution*.

Note: There is a trade-off between computation time and performance for the three variants of the proposed method. The performance of *belief-failure* and *belief-execution* are similar, but *belief-failure* needs much less computation. The *belief-instance* computes the policy even less, but performs much worse for the first several instances. Based on the trade-off, one strategy when applying the proposed method into real-world problems is that when the robot solves and executes a new problem, we use the *belief-failure* variant to begin with. If the robot becomes more confident about the probabilities we switch to the *instance* variant to save policy computation time for the later problem instances.

The rightmost column of Fig. 3 shows the result of the methods averaged over 20 runs. In each run, the ground truth is randomly sampled in [0,1]. There is a similar trend where the proposed method shows a near-optimal performance with less computation time than competing methods.

Remark: We also checked the *Pick and Place* benchmark with the probabilities all set to 0.9 and all set to 0.1. We observed no performance difference among the methods. This highlights that the proposed approach is useful when actions need to be distinguished (are not all good or bad).

Physics simulator results: Fig. 4 shows the results of the two benchmark problems (see Sec. V-A) in Gazebo. The trends are similar to what is shown in Fig. 3. For the result of the *Pick and Place* benchmark, we can see that the *retry* and *replan* baselines are still worse than our method. The results of the *Pick vs. Push* in Fig. 4 show that the average number of steps executed of our method is much less than the baselines. *Replan* is usually worse than *retry* because the *replan* method keeps blocking failed pushing and picking actions and can run out of options and reset from scratch.



Pick and Place: $N = 5$ Pick-Place vs. Push: $N = 3$

Fig. 4: Gazebo simulator results. Top: Number of steps executed to finish each task. Bottom: Computation time of each task. All data are smoothed to be the average from beginning. All the results are collected from Gazebo simulation, averaged over 5 runs. Left: The *Pick and Place* benchmark with 5 objects in total. Right: The *Pick-Place vs. Push* benchmark with 3 objects in total.

The results indicate performance benefits within a few problem instances. This happens even without precise estimates of the transition probabilities since *good enough* estimates can *discriminate* robust actions from ones that fail.

Real-world demonstration: In the demonstration[1], the robot is tasked with three problem instances with the objects shown in the bottom figure of Fig. 1. The proposed method starts to prefer more robust actions quickly.

VI. CONCLUSION

In this work we proposed an efficient framework to solve TAMP problems where action executions can fail. To address realistic problems where ground truth probabilities of successfully executing actions are not known, our method combines MDP with Bayesian inference and posterior sampling techniques. Our method reduces the expected number of steps executed to finish the problem instances in a TAMP domain. The results show that given a stochastic TAMP domain with unknown ground truth probabilities, the proposed method achieves much better performance than competing baselines. In simulated failure benchmarks our performance converges to an optimal method (aware of the underlying ground truths). In physics simulator benchmarks where the ground truth is unknown, the proposed method outperforms replanning and retry baselines. Future efforts will be devoted to finding more efficient ways to solve the task planning problems, reasoning over different outcomes of an execution including the consideration of irreversible actions, efficient inference of diverse and granular causes of failures, and other sources of uncertainty. This work proposes a promising step towards enabling TAMP methods to address the challenges posed when robots solve tasks in the real world.

[1] Real demonstration video: `https://youtu.be/gg_1Xa3v6dQ`

Appeared at ICRA 2022 IEEE International Conference on Robotics and Automation
May 23-27 2022 Philadelphia (PA) USA

References

[1] N. T. Dantam, Z. K. Kingston, S. Chaudhuri, and L. E. Kavraki, "An incremental constraint-based framework for task and motion planning," *The International Journal of Robotics Research*, vol. 37, no. 10, pp. 1134–1151, 2018.

[2] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling, "Ffrob: Leveraging symbolic planning for efficient task and motion planning," *The International Journal of Robotics Research*, vol. 37, no. 1, pp. 104–136, 2018.

[3] S. Srivastava, E. Fang, L. Riano, R. Chitnis, S. Russell, and P. Abbeel, "Combined task and motion planning through an extensible planner-independent interface layer," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 639–646.

[4] N. Shah, D. K. Vasudevan, K. Kumar, P. Kamojjhala, and S. Srivastava, "Anytime integrated task and motion policies for stochastic environments," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 9285–9291.

[5] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial Intelligence*, vol. 101, no. 1, pp. 99–134, 1998. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S000437029800023X

[6] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[7] O. Chapelle and L. Li, "An empirical evaluation of thompson sampling," *Advances in neural information processing systems*, vol. 24, pp. 2249–2257, 2011.

[8] D. Russo, B. Van Roy, A. Kazerouni, I. Osband, and Z. Wen, *A Tutorial on Thompson Sampling*, ser. Foundations and Trends in Machine Learning Series. Now Publishers, 2018. [Online]. Available: https://books.google.com/books?id=nXx6uQEACAAJ

[9] N. L. Johnson, A. W. Kemp, and S. Kotz, *Univariate discrete distributions*. John Wiley & Sons, 2005, vol. 444.

[10] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sendai, Japan, Sep 2004, pp. 2149–2154.

[11] M. Toussaint, "Logic-geometric programming: An optimization-based approach to combined task and motion planning." in *IJCAI*, 2015, pp. 1930–1936.

[12] Z. Kingston, A. M. Wells, M. Moll, and L. E. Kavraki, "Informing multi-modal planning with synergistic discrete leads," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. Paris, France: IEEE, May 2020, pp. 3199–3205. [Online]. Available: http://dx.doi.org/10.1109/icra40945.2020.9197545

[13] T. Pan, A. M. Wells, R. Shome, and L. E. Kavraki, "A general task and motion planning framework for multiple manipulators," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Prague, Czech Republic: IEEE, Sep. 2021. [Online]. Available: http://dx.doi.org/10.1109/iros51168.2021.9636119

[14] L. P. Kaelbling and T. Lozano-Pérez, "Integrated task and motion planning in belief space," *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1194–1227, 2013.

[15] C. R. Garrett, C. Paxton, T. Lozano-Pérez, L. P. Kaelbling, and D. Fox, "Online replanning in belief space for partially observable task and motion problems," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 5678–5684.

[16] C. Mitash, R. Shome, B. Wen, A. Boularias, and K. Bekris, "Task-driven perception and manipulation for constrained placement of unknown objects," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5605–5612, 2020.

[17] K. Kase, C. Paxton, H. Mazhar, T. Ogata, and D. Fox, "Transferable task execution from pixels through deep planning domain learning," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 10 459–10 465.

[18] A. Qureshi, A. Mousavian, C. Paxton, M. Yip, and D. Fox, "Nerp: Neural rearrangement planning for unknown objects," in *Proceedings of Robotics: Science and Systems*, 2021.

[19] J. Mahler and K. Goldberg, "Learning deep policies for robot bin picking by simulating robust grasping sequences," in *Conference on robot learning*. PMLR, 2017, pp. 515–524.

[20] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *The International Journal of Robotics Research*, vol. 37, no. 4-5, pp. 421–436, 2018.

[21] K. Sanders, M. Danielczuk, J. Mahler, A. Tanwani, and K. Goldberg, "Non-markov policies to reduce sequential failures in robot bin picking," in *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*, 2020, pp. 1141–1148.

[22] L. Montesano and M. Lopes, "Active learning of visual descriptors for grasping using non-parametric smoothed beta distributions," *Robotics and Autonomous Systems*, vol. 60, no. 3, pp. 452–462, 2012, autonomous Grasping. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S092188901100145X

[23] A. S. Wang and O. Kroemer, "Learning robust manipulation strategies with multimodal state transition models and recovery heuristics," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 1309–1315.

[24] S. Luo, H. Wu, S. Duan, Y. Lin, and J. Rojas, "Endowing robots with longer-term autonomy by recovering from external disturbances in manipulation through grounded anomaly classification and recovery policies," *Journal of Intelligent & Robotic Systems*, vol. 101, no. 3, p. 51, Feb 2021. [Online]. Available: https://doi.org/10.1007/s10846-021-01312-6

[25] J. C. Hammond, J. Biswas, and A. Guha, "Automatic failure recovery for end-user programs on service mobile robots," *CoRR*, vol. abs/1909.02778, 2019. [Online]. Available: http://arxiv.org/abs/1909.02778

[26] M. Noseworthy, I. Brand, C. Moses, S. Castro, L. Kaelbling, T. Lozano-Perez, and N. Roy, "Active Learning of Abstract Plan Feasibility," in *Proceedings of Robotics: Science and Systems*, Virtual, July 2021.

[27] I. Osband, D. Russo, and B. Roy, "(more) efficient reinforcement learning via posterior sampling," *Advances in Neural Information Processing Systems*, 06 2013.

[28] A. Gopalan and S. Mannor, "Thompson sampling for learning parameterized markov decision processes," in *Conference on Learning Theory*. PMLR, 2015, pp. 861–898.

[29] M. O. Duff, "Optimal learning: Computational procedures for bayes-adaptive markov decision processes." Ph.D. dissertation, University of Massachusetts Amherst, 2003.

[30] M. Ghavamzadeh, S. Mannor, J. Pineau, and A. Tamar, "Bayesian reinforcement learning: A survey," *Found. Trends Mach. Learn.*, vol. 8, no. 5–6, p. 359–483, Nov. 2015. [Online]. Available: https://doi.org/10.1561/2200000049

[31] G. Lee, B. Hou, A. Mandalika, J. Lee, and S. S. Srinivasa, "Bayesian policy optimization for model uncertainty," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=SJGvns0qK7

[32] W. Zhou, S. Bajracharya, and D. Held, "Latent action space for offline reinforcement learning," in *Conference on Robot Learning*, 2020.

[33] S. James, P. Wohlhart, M. Kalakrishnan, D. Kalashnikov, A. Irpan, J. Ibarz, S. Levine, R. Hadsell, and K. Bousmalis, "Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks," 06 2019, pp. 12 619–12 629.

[34] A. M. Wells, M. Lahijanian, L. E. Kavraki, and M. Y. Vardi, "LTLf synthesis on probabilistic systems," *Electronic Proceedings in Theoretical Computer Science*, vol. 326, pp. 166–181, Sep. 2020. [Online]. Available: http://dx.doi.org/10.4204/eptcs.326.11

[35] M. Wells, Z. Kingston, M. Lahijanian, L. E. Kavraki, and M. Y. Vardi, "Finite-horizon synthesis for probabilistic manipulation domains," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 6336–6342.

[36] T. Lattimore and C. Szepesvári, *Bandit algorithms*. Cambridge University Press, 2020.

[37] M. Kwiatkowska, G. Norman, and D. Parker, "PRISM 4.0: Verification of probabilistic real-time systems," in *Proc. 23rd International Conference on Computer Aided Verification (CAV'11)*, ser. LNCS, G. Gopalakrishnan and S. Qadeer, Eds., vol. 6806. Springer, 2011, pp. 585–591.

[38] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, vol. 2. IEEE, 2000, pp. 995–1001.

[39] R. Smith, "Open dynamics engine," 2005. [Online]. Available: http://www.ode.org/

[40] Z. K. Kingston and L. E. Kavraki, "Robowflex: Robot motion planning with moveit made easy," *ArXiv*, vol. abs/2103.12826, 2021.