2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Congress Center Hamburg
Sept 28 - Oct 2, 2015. Hamburg, Germany

Robust Trajectory Selection for Rearrangement Planning as a Multi-Armed Bandit Problem

Michael C. Koval Jennifer E. King Nancy S. Pollard Siddhartha S. Srinivasa
{mkoval,jeking,nsp,siddh}@cs.cmu.edu
The Robotics Institute, Carnegie Mellon University

Abstract— **We present an algorithm for generating open-loop trajectories that solve the problem of rearrangement planning under uncertainty. We frame this as a selection problem where the goal is to choose the most robust trajectory from a finite set of candidates. We generate each candidate using a kinodynamic state space planner and evaluate it using noisy rollouts.**

Our key insight is we can formalize the selection problem as the "best arm" variant of the multi-armed bandit problem. We use the successive rejects algorithm to efficiently allocate rollouts between candidate trajectories given a rollout budget. We show that the successive rejects algorithm identifies the best candidate using fewer rollouts than a baseline algorithm in simulation. We also show that selecting a good candidate increases the likelihood of successful execution on a real robot.

I. Introduction

We explore the *rearrangement planning* problem [11] where a robot must rearrange several objects in a cluttered environment to achieve a goal (Fig.1-Top). Recent work has used simple physics models, like quasistatic pushing [8, 9, 22, 32, 39, 43], to quickly produce efficient and intricate plans. However, the intricacy of these plans makes them particularly sensitive to *uncertainty* in object pose, physics parameters, and trajectory execution.

We address this problem by generating *open-loop trajectories* that are robust to uncertainty. However, this is particularly hard for rearrangement planning.

First, the problem is set in a high-dimensional space with continuous actions. Second, contact causes physics to evolve in complex, non-linear ways and quickly leads to multi-modal and non-smooth distributions [25, 26, 34]. Third, finding good trajectories is inherently hard: most trajectories achieve success with zero probability.

As a consequence, this problem lies outside the domains of standard *conformant planning* [19, 37], *POMDP* [20] and *policy search* [7] algorithms.

In response to these challenges, we propose a domain-agnostic algorithm that only requires: (1) a stochastic method of generating trajectories, (2) the ability to forward-simulate the system's dynamics, and (3) the capability of testing whether an execution is successful.

Exploiting the fact that we can quickly generate feasible *state space* trajectories [22], we formulate rearrangement planning under uncertainty (sections III and IV) as a *selection problem* (Sec.V). Our goal is to choose the most robust trajectory from a finite set of state space trajectory *candidates* (Fig.1-Bottom) generated by this planner. We estimate the success probability of each candidate by performing noisy *rollouts*.

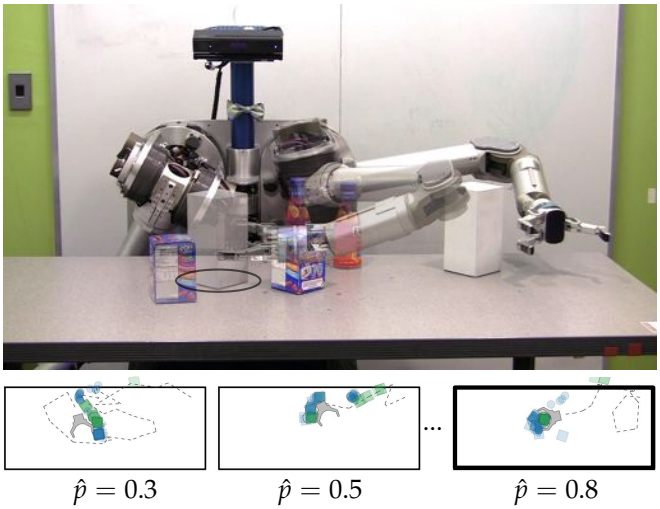


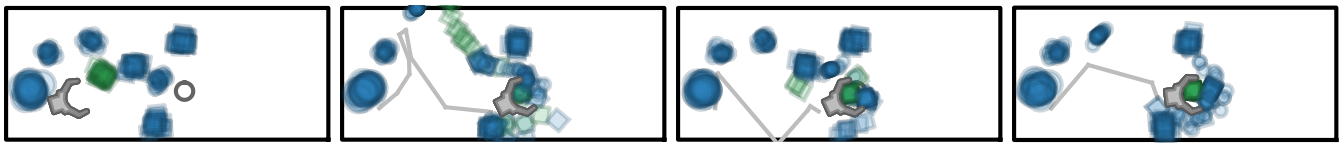
Fig. 1: (Top) HERB [38] is trying to push the white box into the circular goal region. One goal configuration is shown as a semi-transparent overlay. (Bottom) HERB estimates the success probability of several candidate trajectories and executes the trajectory with $\hat{p} = 0.8$.

Thus, given a set of k trajectories, we could perform a fixed sufficient number n of rollouts on each trajectory, requiring a total of nk rollouts.

However, rollouts are computationally expensive, requiring the evaluation of a full physics simulation. Thus, we seek an algorithm that can *efficiently* choose the best trajectory given a small rollout budget.

Our key insight is that we can formalize this selection problem as an instance of the "best arm" variant [31] of the k-armed bandit problem [35]. In our formulation (Sec.VI), each candidate trajectory is an "arm" and the goal is to identify the best arm given a fixed budget of rollouts. We use the *successive rejects* algorithm [2] to select the best candidate.

We evaluate the proposed algorithm on the rearrangement planning problem [11] in simulation and on a real robot (Sec.VII). First, we validate that a state space



(a) Initial State (b) $\hat{p} = 0.24$ (c) $\hat{p} = 0.58$ (d) $\hat{p} = 1.0$

Fig. 2: (a) The initial state. Start poses for each object in the scene are drawn from a Gaussian distribution with $\mu = \vec{0}, \Sigma^{1/2} = \text{diag}[2\text{ cm}, 2\text{ cm}, .1\text{ rad}]$. The robot must push the green box into the goal region indicated by the gray circle. (b) A candidate with low success probability. Here, several rollouts lead to the target object failing to end up in the goal. (c) A more robust candidate. Here many of the noisy rollouts achieve the goal. (d) A trajectory with all noisy rollouts achieving the goal.

planner can successfully generate candidate trajectories with varying success probabilities (P1, Sec.VII-B). Next, we show that the successive rejects algorithm identifies the best candidate using fewer rollouts than a baseline (H1, Sec.VII-D). Finally, we demonstrate that selecting a good candidate trajectory increases the likelihood of successful execution on a real robot (H2, Sec.VII-E).

The proposed algorithm is applicable to planning under uncertainty in *any domain* where the ability to perform rollouts is available. We believe that this algorithm is most useful when the standard assumptions made by belief space planners are not satisfied. We describe these assumptions in Sec.II.

II. Related Work

Recent work has formulated rearrangement planning [39] as a *kinodynamic motion planning problem* [13] in the joint configuration space of the robot and movable objects in the environment [15, 22]. This formulation naturally copes with the continuous state and action spaces. Unfortunately, these algorithms do not consider uncertainty.

One method of coping with uncertainty is to formulate rearrangement planning as a *partially observable Markov decision process* (POMDP) [20]. POMDP solvers reason about uncertainty and can integrate observations, but do not easily generalize to continuous action spaces or non-additive reward functions. Most applications of POMDPs to manipulation rely on discretization [17, 18, 26] or *receding horizon planning* [34, 41].

In this paper, we consider planning *open-loop trajectories*. In this case, it is natural to formulate rearrangement planning as a *conformant planning* problem where the goal is to find an open-loop trajectory that succeeds despite non-deterministic uncertainty [6, 37]. *Conformant probabilistic planning* relaxes this requirement by maximizing the probability of achieving the goal under probabilistic uncertainty [19, 29]. Early work in manipulation applied conformant planning to the *part alignment problem* by carefully converting the continuous configuration space to a discrete graph [4, 5, 14, 16, 18, 33]. More recently, probabilistic conformant planning has been used to search of discrete sets of primitive actions [10, 12, 30].

In this paper, we introduce *a conformant probabilistic planning algorithm that searches for solutions in the infinite-dimensional space of trajectories*. This proposed algorithm can be thought of as a *policy search* algorithm [7]. Policy search has been successfully used in reinforcement learning [3, 24] and, more recently, applied to motion planning in the form of *trajectory optimization* [21, 36, 44].

Applying policy search to the rearrangement planning problem is difficult because the set of all policies Π is infinite-dimensional and most policies achieve the goal with probability zero. As a result, little-to-no information is available to guide a local optimizer. We subvert this problem by using a motion planner to generate a set of *candidate trajectories* that achieve the goal with non-zero probability. Then, we select the best candidate from the set.

Similar methods have solved motion planning under the linear-quadratic-Gaussian (LQG) assumptions [42]. This work uses a RRT [28] to generate candidate trajectories and uses the LQG assumptions to quickly evaluate their robustness. The rearrangement planning problem does not satisfy the LQG assumptions. We formulate the selection problem as a multi-armed bandit and use the successive rejects algorithm [2] to allocate Monte Carlo rollouts between candidates. Recent work has applied the same algorithm to select a high quality grasp under object shape uncertainty [27].

III. The Rearrangement Planning Problem

Suppose we have a robot with configuration space $\mathcal{C}^{\mathcal{R}}$. The world is populated with a set of m objects that the robot is allowed to manipulate. Each object has a configuration space C^i. Additionally, there is a set of obstacles that the robot is forbidden to contact.

We define the state space X as the Cartesian product space of the configurations spaces of the robot and objects $X = C^R \times C^1 \times \cdots \times C^m$. The free space $X_{\text{free}} \subseteq X$ is the set of states where the robot and all objects are not penetrating themselves, an obstacle, or each other. Note that this formulation allows for non-penetrating contact between entities, which is critical for manipulation.

The state $x \in X$ evolves nonlinearly based on the physics of the manipulation, i.e. the motion of the

Algorithm 1 Fixed n Selection

Input: candidates $\Pi_{\text{can}} = \{\pi_1, \ldots, \pi_k\}$
Output: selection $\hat{\pi}^* \in \Pi_{\text{can}}$

1: **for** $i = 1, \ldots, k$ **do**
2: $\quad s_i \leftarrow 0$
3: $\quad$ **for** $j = 1, \ldots, n$ **do** ▷ Perform n rollouts
4: $\quad\quad x_j \sim p(x_{\text{s}})$
5: $\quad\quad \xi_j \leftarrow \text{Rollout}(x_j, \pi_i)$
6: $\quad\quad s_i \leftarrow s_i + \Lambda[\xi_j]$
7: $\hat{\pi}^* \leftarrow \arg\max_{\pi_i \in \Pi_{\text{can}}} s_i$ ▷ Choose the best $\hat{p}_{\pi_i}$

objects is governed by the characteristics of the contact between the objects and the manipulator. We describe this evolution as a non-holonomic constraint:

$$\dot{x} = f(x, u)$$

where $u \in \mathcal{U}$ is an instantaneous control input. The function f encodes the physics of the environment.

The task of *rearrangement planning* is to find a feasible trajectory $\xi : \mathbb{R}^{\geq 0} \to X_{\text{free}}$ starting from a state $\xi(0) = x_{\text{s}} \in X_{\text{free}}$ and ending in a goal region $\xi(T_\xi) \in X_{\text{g}} \subseteq X_{\text{free}}$ at some time $T_\xi \geq 0$. A trajectory ξ is feasible if there exists a mapping $\pi : \mathbb{R}^{\geq 0} \to \mathcal{U}$ such that $\dot{\xi}(t) = f(\xi(t), \pi(t))$ for all $t \geq 0$. This requirement ensures we can satisfy the constraint f while following ξ by executing the controls dictated by π.

IV. Conformant Probabilistic Planning

Our formulation of the rearrangement planning problem assumes that the initial state x_{s} is known and the non-holonomic constraint f is deterministic. This assumption rarely holds in practice and, as a result, a sequence of control inputs π that nominally solves the problem posed in Sec.III is unlikely to succeed when executed on the robot.

Instead, we frame rearrangement planning as a *conformant probabilistic planning* problem [19]. We begin in an initial *belief state* $p(x_{\text{s}})$ that is a probability distribution over the start state x_{s}.

Our state evolves as a stochastic non-holonomic system. We use $p(\xi|\pi)$ to denote the distribution of state space trajectories that results from starting in $x_{\text{s}} \sim p(x_{\text{s}})$, then executing the control inputs π under the stochastic transition dynamics.

Our goal is find a sequence of control inputs π that maximizes the probability $p_\pi = \Pr[\Lambda[\xi] = 1]$ of satisfying the *success functional* $\Lambda : \Xi \to \{0, 1\}$ where Ξ is the set of all trajectories.

In the case of rearrangement planning, we define the success functional

$$\Lambda[\xi] = \begin{cases} 1 & : \xi(T_\xi) \in X_{\text{g}} \\ 0 & : \text{otherwise} \end{cases} \tag{1}$$

where T_ξ is the duration of trajectory ξ.

V. Trajectory Selection

Our goal is to find the most robust sequence of control inputs

$$\pi_{\text{best}} = \arg\max_{\pi \in \Pi} p_\pi. \tag{2}$$

where Π is the set of all sequences of control inputs. Computing π_{best} exactly is not possible. In the following sections, we outline a method of finding an approximate solution and provide intuition about the accuracy of the approximation.

A. Approximating Success Probability

We write p_π as the expectation

$$p_\pi = E_{\xi \sim p(\xi|\pi)}\Big[\Lambda[\xi]\Big] = \int_\Xi \Lambda[\xi] p(\xi|\pi) d\xi$$

over the space of trajectories Ξ. Directly computing this integral requires the ability to evaluate $p(\xi|\pi)$. In the case of rearrangement planning, this not available.

Instead, we approximate this expectation as the mean

$$\hat{p}_\pi = \frac{1}{n} \sum_{j=1}^{n} \Lambda[\xi_j] \tag{3}$$

of Λ over n rollouts $\xi_1, \ldots, \xi_n \in \Xi$. The law of large numbers guarantees that $\lim_{n \to \infty} \hat{p}_\pi = p_\pi$; i.e. our approximation $\hat{p}_\pi$ approaches the true success probability p_π as the number of samples n increases.

Each rollout is an independent sample from the distribution $\xi_j \sim p(\xi|\pi)$. We generate rollout ξ_j by sampling an initial state $x_j \sim p(x_{\text{s}})$, then forward-propagating x_j through the stochastic dynamics while executing the control inputs dictated by π.

B. Trajectory Selection

Solving Eq.(2) requires finding the global optimum of the infinite-dimensional set of sequences of control inputs Π. To gain tractability, we first *generate* a candidate set of control sequences $\Pi_{\text{can}} \subseteq \Pi$ and then *select* the most robust candidate

$$\pi^* = \arg\max_{\pi \in \Pi_{\text{can}}} p_\pi$$

from this finite set.

We populate Π_{can} by drawing samples from a distribution over Π. In the rearrangement planning problem, we generate candidates by repeatedly calling a kinodynamic RRT [22] in the state space X with start x_{s} and goal X_{g}. Our intuition is that the RRT will generate a diverse set of candidates that achieve the goal with varying success probabilities.

Given Π_{can}, the simplest selection algorithm is to choose

$$\hat{\pi}^* = \arg\max_{\pi \in \Pi_{\text{can}}} \hat{p}_\pi.$$

using our approximation $\hat{p}_\pi$ of p_π.

Alg.1 outlines this algorithm. For each candidate π_i, we perform n rollouts (line 1) and count the number of successes, s_i. Then $\hat{p}_{\pi_i} = s_i / n$.

Algorithm 2 Successive Rejects

Input: candidates $\Pi_{\text{can}} = \{\pi_1, \ldots, \pi_k\}$
Output: selection $\hat{\pi}^* \in \Pi_{\text{can}}$

1: $A = \{1, \ldots, k\}$
2: $s_i \leftarrow 0$ for all $i \in A$
3: $n_0 \leftarrow 0$
4: **for** $l = 1, \ldots, k-1$ **do**
5: $\quad n \leftarrow n_l - n_{l-1}$ ▷ See Eq.(6) for def'n
6: $\quad$ **for all** $i \in A$ **do**
7: $\quad\quad$ **for** $j = 1, \ldots, n$ **do** ▷ Perform n rollouts
8: $\quad\quad\quad x_j \sim p(x_{\text{s}})$
9: $\quad\quad\quad \xi_j \leftarrow \text{Rollout}(x_j, \pi_i)$
10: $\quad\quad\quad s_i \leftarrow s_i + \Lambda[\xi_j]$
11: $\quad i_{\text{worst}} \leftarrow \arg\min_{i \in A}(s_i / n_l)$
12: $\quad A \leftarrow A \setminus \{i_{\text{worst}}\}$ ▷ Reject the worst $\hat{p}_\pi$
13: $\{\hat{\pi}^*\} \leftarrow A$

C. Effect of Approximation Error

Approximating p_π with $\hat{p}_\pi$ comes at a cost: we may incorrectly select a sub-optimal candidate $\hat{\pi}^* \neq \pi^*$ due to error. An error occurs when a candidate π_i performs well on the n Monte Carlo rollouts used to estimate p_{π_i}, but performs poorly on the underlying distribution. This phenomenon parallels the concept of *overfitting*.

The number of successful rollouts s_i is a Binomial random variable. The magnitude of the error $|\hat{p}_\pi - p_\pi|$ is unbounded for any finite number n of rollouts. However, we can use a Binomial confidence interval to bound the probability

$$\Pr\left(|\hat{p}_\pi - p_\pi| > \delta\right) < \alpha, \tag{4}$$

of an error with magnitude δ occurring. When the central limit theorem holds, the Wald interval states

$$\delta = z_{1-\alpha/2}\sqrt{\frac{1}{n}\hat{p}_\pi(1-\hat{p}_\pi)} \tag{5}$$

where $z_{1-\alpha/2} = \Phi^{-1}(1-\alpha/2)$ is the $(1-\alpha/2)$-th percentile of the Gaussian distribution.

Given a desired δ and α, we can solve for the number of samples n required to satisfy Eq.(4). The value δ is related to the minimum difference between two trajectories that we can reliably detect. Ideally, we would drive $\delta \to 0$, allowing us to differentiate trajectories with similar success rates. From Eq.(5) we see this requires a prohibitively large value of n; e.g. reducing δ by half requires increasing n by a factor of four.

VI. Multi-Armed Bandit Formulation

The approach described in Sec.V assumes that we need to perform the *same number of rollouts on all candidates*. Our analysis in Sec.V-C suggests that this is wasteful: we can use fewer samples to differentiate between two candidates that have vastly different success probabilities.

We formalize this intuition by framing the trajectory selection problem as a variant of the *multi-armed bandit problem* [35] (Sec.VI-A). This enables us to use the *successive rejects* algorithm [2] to efficiently identify the best candidate (Sec.VI-B).

A. Multi-Armed Bandit Formulation

A *multi-armed bandit problem* [35] is a sequential process where an agent is presented with k arms and at each timestep must choose only one arm to pull. After pulling an arm, the agent receives a reward. The goal of the agent is to maximize expected sum of reward. Since the agent does not know the distribution of rewards, it must trade off between *exploring* different arms and *exploiting* its estimate of the best arm.

Trajectory selection is an instance of the multi-armed bandit algorithm where each candidate $\pi_i \in \Pi_{\text{can}}$ is an arm. Pulling the i-th arm corresponds to performing one rollout, ξ_j, of π_i. We receive a binary reward $\Lambda[\xi_j] \sim \text{Bernoulli}[p_{\pi_i}]$ depending upon whether the rollout, $\xi_j \sim p(\xi|\pi_i)$, achieves the goal.

Unlike the canonical bandit problem [35], the goal of trajectory selection is not to maximize the expected sum of reward across all rollouts. Instead, after exhausting a budget of b rollouts, we choose the single best candidate $\hat{\pi}^* = \arg\max_{\pi \in \Pi_{\text{can}}} \hat{p}_\pi$ and execute $\hat{\pi}^*$ on the real robot. Our goal is to optimally allocate the b rollouts amongst the k candidates to maximize the success probability $p_{\hat{\pi}^*}$ of our selection $\hat{\pi}^*$. This is known as the *best arm* or *pure exploration* variant of the bandit problem [2, 31].

B. Successive Rejects Algorithm

The *successive rejects* algorithm (Alg.2) is a principled method of solving the best arm problem [2]. The intuition behind the algorithm is to partition the b rollouts between several phases (line 4). A set $A \subseteq \Pi_{\text{can}}$ is repeatedly shrunk until it contains the single best candidate. In each phase, we perform an equal number of rollouts n (line 5) on each remaining candidate $\pi \in A$ and remove the candidate $\pi_{\text{worst}} = \arg\min_{\pi \in A} \hat{p}_\pi$ with the lowest estimated success probability from A (line 12). This repeats until we have completed $k-1$ phases. At this point, we return the remaining candidate $\hat{\pi}^*$ in A as our selection (line 13).

The key component of the successive rejects algorithm is how to select the number of rollouts to perform in phase l. If we have k candidates and a total budget b of rollouts, then we choose

$$n_l = \left\lceil \frac{1}{\overline{\log}\ k} \cdot \frac{b-k}{k+1-l} \right\rceil \tag{6}$$

where $\overline{\log}\ k = 1/2 + \sum_{i=2}^{k} 1/i$ and $\lceil \cdot \rceil$ denotes the ceiling operator [2]. n_l is the *total number* of rollouts performed across all phases on each candidate remaining in phase l. Only $n = n_l - n_{l-1}$ of these rollouts are performed in phase l.

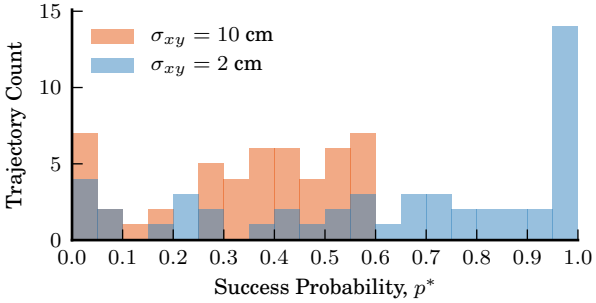


Fig. 3: Histogram of the success probabilities p^* achieved by 50 candidate control sequences under low (blue) and high (orange) levels of uncertainty. This data was generated from the scene shown in Fig.2.

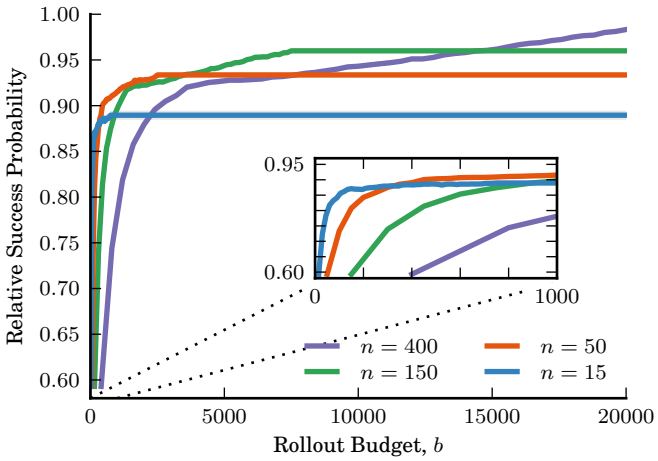


Fig. 4: Success probability relative to π^* of the fixed rollout algorithm. Small values of n quickly find solutions given a small budget (inset). However, large values of n find better solutions as the budget increases. This data was generated from the scene shown in Fig.2.

Given the choice of Eq.(6) for n_l, prior work [2] shows that the probability ϵ of Alg.2 making an error is bounded by

$$\epsilon \leq \frac{k(k-1)}{2} \exp\left[-\frac{b-k}{H_2 \overline{\log}\ k}\right]$$

where $H_2 = \max_{2 \leq i \leq k}(i\Delta_{(i)}^{-2})$ and $\Delta_{(i)} = p_{\pi^*} - p_{\pi_{(i)}}$ is the gap between the best candidate and the i-th best candidate $\pi_{(i)}$. We can additionally bound $H_2 \leq H_1$ with $H_1 = \sum_{i=2}^{k} \Delta_{(i)}^{-2}$.

The quantities H_1 and H_2 formalize the difficulty of the problem [2]. Since $\Delta_{(i)}$ is in the denominator of H_1, a problem is more difficult if the gaps $\Delta_{(1)}, \ldots \Delta_{(k)}$ are small. This confirms our analysis from Sec.V-C that it is difficult to differentiate between two candidates with similar success probabilities.

VII. Experimental Results

First, we verify that the following two properties hold in our test environment:

- **P1** The state space planner can generate candidate trajectories with varying success probabilities.
- **P2** Increasing the number of rollouts per trajectory improves the selected trajectory.

Finally, we test two hypotheses:

- **H1** The successive rejects algorithm requires fewer rollouts to find the best trajectory than a baseline that uses a fixed number of rollouts.
- **H2** Selecting a good trajectory increases the likelihood of successful execution on a real robot.

A. Experimental Setup

We evaluate our algorithm on a dataset of seven randomly generated scenes. In each test, the goal is to push a *target object* into a 5 cm radius goal region with fixed location across all scenes. Each scene contains between one and seven movable objects in the reachable workspace of the robot. The starting pose of the robot and all movable objects are selected uniformly at random from X_{free}.

We plan for a hand pushing objects in the plane; i.e. $C^R = C^1 = \cdots = C^M = SE(2)$. This enables us to use Box2D [1] as our physics simulator. We simulate quasistatic interactions by forcing small object velocities to zero at the end of each action and invalidating actions which result in large object velocities.

We use an implementation of a kinodynamic RRT [22] implemented in the Open Motion Planning Library (OMPL) framework [40] to generate candidate trajectories.

B. Robustness to Uncertainty

We test **P1** by using the state space planner to generate 50 candidate trajectories Π_{can} for each scene. We execute 400 noisy rollouts of each candidate $\pi_i \in \Pi_{\text{can}}$ and count the number of rollouts s_i that achieve the goal to compute $\hat{p}_{\pi_i} = s_i/400$. Using $n = 400$ rollouts gives us 95% confidence that our estimate $\hat{p}_{\pi_i}$ is within 5% of the true success probability p_{π_i}.

Fig.3 shows two distributions of success probabilities for Π_{can} on the scene shown in Fig.2. The first distribution (orange) samples the initial pose of each object from a Gaussian distribution with zero mean and $\Sigma^{1/2} = \text{diag}[10\text{ cm}, 10\text{ cm}, 0.1\text{ rad}]$. The results show that all trajectories lie in the range $0 \leq \hat{p}_{\pi_i} \leq 0.6$.

For the second distribution (blue), we sample the initial pose of the objects from a narrower Gaussian distribution with zero mean and $\Sigma^{1/2} = \text{diag}[2\text{ cm}, 2\text{ cm}, 0.1\text{ rad}]$. Sampling from this tighter distribution results in several trajectories that achieve perfect or near perfect probability of success.

These results show that we can easily generate different trajectories with our state space planner. More importantly, this confirms that the trajectories produced by our planner differ in robustness to uncertainty.

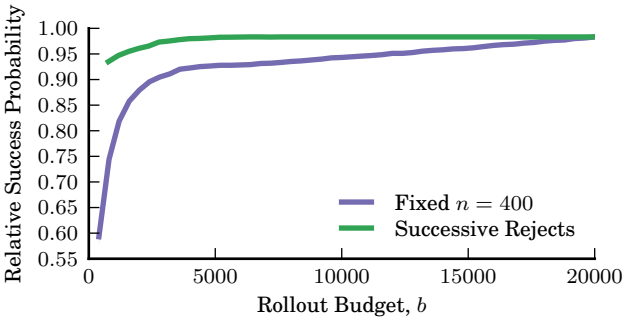


Fig. 5: Success probability relative to π^* of the fixed rollout algorithm ($n = 400$) and the successive rejects algorithm. The successive rejects algorithm finds better solutions given the same budget. This data was generated from the scene shown in Fig.2.

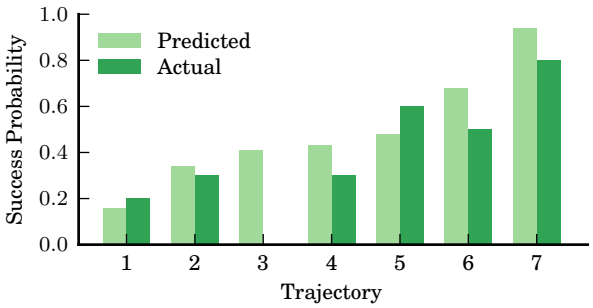


Fig. 6: Comparison of the success probability estimated through Monte Carlo rollouts (predicted) and observed during ten executions on the real-robot (actual). Trajectories with a high estimated probability of success tend to succeed more often when executed on the real-robot.

C. Analysis of the Fixed Rollout Method

Next, we verify **P2** with a baseline selection algorithm that uses a fixed number n of rollouts to evaluate each trajectory. We compare multiple values of $n \in \{15, 50, 150, 400\}$ using the same set of candidate trajectories as in Sec.VII-B. We use the calculated $\hat{p}_{\pi_i}$ values as the ground truth success probabilities $p^*_{\pi_i}$ for each trajectory. We discard these rollouts and generate 400 new rollouts to test the selection algorithm.

Fig.4 shows the ground truth success probability p^* of the trajectory selected with a budget of b total rollouts. Results are averaged across 300 trials. With a small rollout budget ($b < 1000$), small values of n find trajectories with higher success probability (Fig.4). This is expected, as these planners are able to evaluate more trajectories using b rollouts. For example, with a budget of $b = 800$ rollouts, the $n = 400$ planner can only evaluate two trajectories while the $n = 15$ planner can evaluate all 50.

Large values of n find better solutions (Fig.4) as b increases. This is also expected. Increasing n shrinks the confidence interval and allows the planner to accurately differentiate between trajectories with similar success probabilities.

D. Successive Rejects Algorithm

We test **H1** by comparing the successive rejects algorithm described in Sec.VI-B against a baseline algorithm that uses a fixed number n of rollouts. We begin by allocating a budget of $b = 800$ rollouts to each algorithm. We run multiple iterations, increasing b by 400 rollouts each time until we reach $b = 20000$. At this point, the $n = 400$ algorithm can evaluate all 50 candidates. We record the ground-truth success probability p^* of the trajectory selected in each iteration.

Fig.5 shows the relative success probability of each selection algorithm averaged over 300 trials. In each trial, we randomize the order of Π_{can} and the outcome of the noisy rollouts for each candidate trajectory $\pi_i \in \Pi_{\text{can}}$. The ordering of both the trajectories and the outcomes is kept constant within a trial.

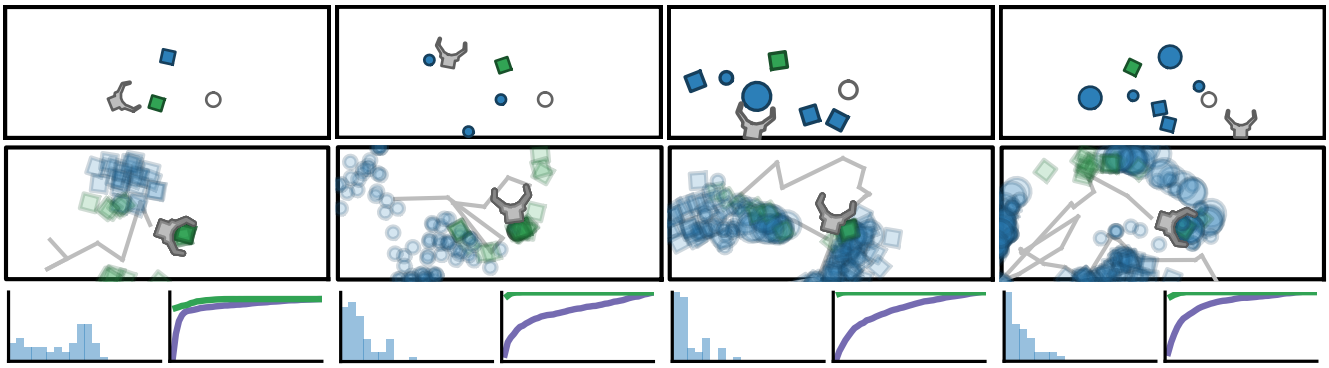
For a fixed budget of 20000, both planners find the same near-optimal trajectory. However, the successive rejects algorithm finds this trajectory with with far fewer rollouts on average (1638 rollouts vs. 8685 rollouts). A *t-test* confirms this difference is significant (two-sample $t(318) = 19.7, p < 0.0001$). This supports our hypothesis: *The successive rejects algorithm requires fewer rollouts to find the best trajectory than a baseline planner that uses a fixed number of rollouts.*

E. Real Robot Experiments

Finally, we test **H2** by executing trajectories selected by our algorithm on a real robot. We use HERB [38], a robot designed and built by the Personal Robotics Laboratory at Carnegie Mellon University to conduct these experiments. We plan for the 7-DOF left arm in joint space. The planner uses a quasistatic physics simulator capable of modeling interactions with the full arm, instead of the two-dimensional Box2D physics engine used to generate the simulation results.

We generate four random scenes using the method described above, construct each by measuring the nominal location of each object relative to HERB, and perturb the pose of each object by an offset drawn from a Gaussian distribution. We use the successive rejects selection algorithm to generate five trajectories for each scene and record the success rate estimated by the planner for each candidate trajectory. Finally, we select seven trajectories with varying estimated success rates and execute each 10 times on HERB. We record success or failure of each execution.

Fig.6 shows the estimated and actual success rate of each trajectory. The estimated success rate does not perfectly predict the results that we see on the real robot. However, there is a clear correlation between the estimated success rate and the probability that executing the trajectory succeeds on the real robot. This supports our hypothesis: *Selecting a good trajectory*



(a) $m = 1$ Object (b) $m = 3$ Objects (c) $m = 5$ Objects (d) $m = 6$ Objects

Fig. 7: Results from four different scenes. (Top) Initial configuration. (Middle) The best candidate trajectory returned by the successive rejects algorithm. (Bottom-Left) Histogram of the success probability of plans returned by the state space planner . See Fig.3 for more information. (Bottom-Right) Success probability of the selected trajectory as a function of rollout budget for $n = 400$ (blue) and the successive rejects algorithm (green). See Fig.5 for more information.

increases the likelihood of successful execution on a real robot.

The qualitative aspects of the real-robot experiments are perhaps more interesting. Fig.8 shows *Trajectory 7*. This trajectory exhibits the highest success rate. As can be seen, a sweeping motion is used to move the target object. Prior work [12] shows that sweeping primitives are particularly effective at reconfiguring objects under uncertainty. It in encouraging that our selection algorithm can produce similar behavior.

Next, we examine *Trajectory 3*. Our planner estimated the success rate of this trajectory to be $\hat{p} = 0.41$. However, we were unable to achieve any successful executions on the real robot. Examining the rollouts used to evaluate this trajectory reveals errors in the physics model that are not modeled in our simulator. This highlights a fundamental limitation of our formulation: our estimate of $\hat{p}_\pi$ can only be as good as our model of noise in the system dynamics.

VIII. Discussion and Future Work

Our results show that selecting the best trajectory from a set of candidates is a surprisingly effective method of improving the likelihood of the plan executing successfully. Additionally, we show that using the successive rejects algorithm [2] dramatically reduces the number of rollouts need for a desired level of performance. This algorithm is simple to implement and performs strictly better than using a fixed number of rollouts.

The performance of this algorithm depends entirely on the quality of the trajectories included in Π_{can}. First, the successive rejects algorithm is most beneficial when few candidates achieve the goal with high probability (Fig.7b). The improvement is minimal when many robust candidates exist (Fig.7a). Second, the success probability of the output trajectory cannot exceed that of π^*. We see this in Fig.7d: the set of candidate trajectories all achieve the goal with low probability, so the best trajectory is brittle.

We can avoid this issue by seamlessly trading off between evaluating existing candidates and expanding Π_{can}. This is a *smooth bandit problem* [23] defined over the continuous set of control sequences Π. The key challenge is to define a meaningful metric over Π and insure that Λ is sufficiently smooth. This formulation may result in an *anytime* planner that continually outputs better trajectories over time.

Ideally, we would incorporate uncertainty directly into the planner. Prior work [10, 12] has shown that some pushing actions reduce uncertainty, while others increase it. Currently, we rely on such actions being randomly generated by a state space planner. Our planner should actively seek to include such actions, since they increase the probability of success.

Acknowledgements

This material is based on work supported by NSF award IIS-1218182, the NASA Space Technology Research Fellowship program (awards NNX13AL61H and NNX13AL62H), and the Toyota Motor Corporation. We would like to thank Akshay Krishnamurthy, Aaditya Ramdas, Ryan Tibshirani, and the members of the Personal Robotics Lab for their input.

References

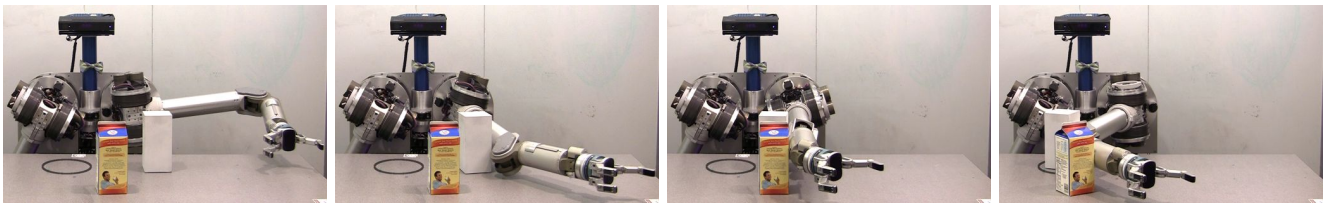
[1] Box2D. `http://box2d.org`, 2010 (accessed August 2014).

[2] J-Y Audibert and S. Bubeck. Best arm identification in multi-armed bandits. In *COLT*, 2010.

[3] J.A. Bagnell, S.M. Kakade, J.G. Schneider, and A.Y. Ng. Policy search by dynamic programming. In *NIPS*, 2003.

[4] R-P Berretty, K. Goldberg, M.H. Overmars, and A. Frank van der Stappen. Orienting parts by inside-out pulling. In *IEEE ICRA*, 2001.

[5] M. Brokowski, M. Peshkin, and K. Goldberg. Curved fences for part alignment. In *IEEE ICRA*, 1993.



(a) Initial Scene (b) (c) (d) Final Scene

Fig. 8: Trajectory 7, from the results presented in Fig.6, executed on HERB [38]. The trajectory selection algorithm chose to use a sweeping motion to robustly push the white box into the circular goal region.

[6] A. Cimatti and M. Roveri. Conformant planning via symbolic model checking. In *JAIR*, 2000.

[7] M.P. Deisenroth, G. Neumann, and J. Peters. A survey on policy search for robotics. *FTROB*, 2(1-2):1–142, 2013.

[8] M.R. Dogar, K. Hsiao, M. Ciocarlie, and S.S. Srinivasa. Physics-based grasp planning through clutter. In *RSS*, 2012.

[9] M.R. Dogar, M.C. Koval, A. Tallavajhula, and S.S. Srinivasa. Object search by manipulation. In *IEEE ICRA*, 2013.

[10] M.R. Dogar and S.S. Srinivasa. Push-grasping with dexterous hands: Mechanics and a method. In *IEEE/RSJ IROS*, 2010.

[11] M.R. Dogar and S.S. Srinivasa. A framework for push-grasping in clutter. In *RSS*, 2011.

[12] M.R. Dogar and S.S. Srinivasa. A planning framework for non-prehensile manipulation under clutter and uncertainty. *AuRo*, 33(3):217–236, 2012.

[13] B. Donald, P. Xavier, J. Canny, and J. Reif. Kinodynamic motion planning. *JACM*, 40(5):1048–1066, 1993.

[14] M.A. Erdmann and M.T. Mason. An exploration of sensorless manipulation. *IJRA*, 4(4):369–379, 1988.

[15] J.A. Haustein, J.E. King, S.S. Srinivasa, and T. Asfour. Kinodynamic randomized rearrangement planning via dynamic transitions between statically stable configurations. In *IEEE ICRA*, 2015.

[16] H. Hitakawa. Advanced parts orientation system has wide application. In *Assembly Automation*, 1988.

[17] M. Horowitz and J. Burdick. Interactive non-prehensile manipulation for grasping via POMDPs. In *IEEE ICRA*, 2013.

[18] K. Hsiao. *Relatively robust grasping*. PhD thesis, 2009.

[19] N. Hyafil and F. Bacchus. Conformant probabilistic planning via CSPs. In *ICAPS*, 2003.

[20] L.P. Kaelbling, M.L. Littman, and A.R. Cassandra. Planning and acting in partially observable stochastic domains. *JAIR*, 101(1–2):99–134, 1998.

[21] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal. STOMP: Stochastic trajectory optimization for motion planning. In *IEEE ICRA*, pages 4569–4574, 2011.

[22] J.E. King, J.A. Haustein, S.S. Srinivasa, and T. Asfour. Nonprehensile whole arm rearrangement planning on physics manifolds. In *IEEE ICRA*, 2015.

[23] R.D. Kleinberg. Nearly tight bounds for the continuum-armed bandit problem. In *NIPS*, 2004.

[24] J. Kober and J.R. Peters. Policy search for motor primitives in robotics. In *NIPS*, pages 849–856, 2009.

[25] M.C. Koval, M.R. Dogar, N.S. Pollard, and S.S. Srinivasa. Pose estimation for contact manipulation with manifold particle filters. In *IEEE/RSJ IROS*, 2013.

[26] M.C. Koval, N.S. Pollard, and S.S. Srinivasa. Pre- and post-contact policy decomposition for planar contact manipulation under uncertainty. In *RSS*, 2014.

[27] M. Laskey, Z. McCarthy, J. Mahler, F.T. Pokorny, S. Patil, J. van den Berg, D. Kragic, P. Abbeel, and K. Goldberg. Budgeted multi-armed bandit models for sample-based grasp planning in the presence of uncertainty. In *IEEE ICRA*, 2015.

[28] S.M. LaValle. Rapidly-exploring Random Trees: A new tool for path planning. 1998.

[29] J. Lee, R. Marinescu, and R. Dechter. Applying marginal MAP search to probabilistic conformant planning: Initial results. In *Workshops at AAAI*, 2014.

[30] M. Levihn, J. Scholz, and M. Stilman. Planning with movable obstacles in continuous environments with uncertain dynamics. In *IEEE ICRA*, 2013.

[31] O. Maron and A.W. Moore. Hoeffding races: Accelerating model selection search for classification and function approximation. In *NIPS*, 1993.

[32] D. Nieuwenhuisen, A. Stappen., and M. Overmars. An effective framework for path planning amidst movable obstacles. In *WAFR*, 2008.

[33] M.A. Peshkin and A.C. Sanderson. Planning robotic manipulation strategies for workpieces that slide. *IJRA*, 4(5):524–531, 1988.

[34] R. Platt, L.P. Kaelbling, T. Lozano-Pérez, and R. Tedrake. Efficient planning in non-Gaussian belief spaces and its application to robot grasping. In *ISRR*, 2011.

[35] H. Robbins. Some aspects of the sequential design of experiments. *Bulletin of the AMS*, 58(5):527–535, 1952.

[36] J. Schulman, J. Ho, A. Lee, I. Awwal, H. Bradlow, and P. Abbeel. Finding locally optimal, collision-free trajectories with sequential convex optimization. In *RSS*, volume 9, pages 1–10, 2013.

[37] D.E. Smith and D.S. Weld. Conformant graphplan. In *AAAI*, 1998.

[38] S.S. Srinivasa, D. Ferguson, C.J. Helfrich, D. Berenson, A. Collet, R. Diankov, G. Gallagher, G. Hollinger, J. Kuffner, and M.V. Weghe. HERB: A Home Exploring Robotic Butler. *AuRo*, 28(1):5–20, 2010.

[39] M. Stilman. and J. Kuffner. Navigation among movable obstacles: Real-time reasoning in complex environments. In *IEEE-RAS Humanoids*, 2004.

[40] I. Sucan, M. Moll, and L. Kavraki. The Open Motion Planning Library. *IEEE RAM*, 19(4):72–82, 2012.

[41] N.E. Du Toit and J.W. Burdick. Robotic motion planning in dynamic, cluttered, uncertain environments. In *IEEE ICRA*, pages 966–973, 2010.

[42] J. van den Berg, P. Abbeel, and K. Goldberg. LQG-MP: Optimized path planning for robots with motion uncertainty and imperfect state information. In *RSS*, 2010.

[43] J. van den Berg, M. Stilman, J. Kuffner, M. Lin, and D. Manocha. Path planning among movable obstacles: a probabilistically complete approach. In *WAFR*, 2008.

[44] M. Zucker, N. Ratliff, A.D. Dragan, M. Pivtoraiko, M. Klingensmith, C.M. Dellin, J.A. Bagnell, and S.S. Srinivasa. CHOMP: Covariant Hamiltonian optimization for motion planning. *IJRR*, 32(9-10):1164–1193, 2013.