Non-prehensile Rearrangement Planning with Learned Manipulation States and Actions

Joshua A. Haustein[1,†], Isac Arnekvist[1,†], Johannes Stork[1], Kaiyu Hang[2] and Danica Kragic[1]

***Abstract*— In this work we combine sampling-based motion planning with reinforcement learning and generative modeling to solve non-prehensile rearrangement problems. Our algorithm explores the composite configuration space of objects and robot as a search over robot actions, forward simulated in a physics model. This search is guided by a generative model that provides robot states from which an object can be transported towards a desired state, and a learned policy that provides corresponding robot actions. As an efficient generative model, we apply Generative Adversarial Networks.**

I. Introduction

In cluttered environments, a robot needs to be able to rearrange obstacles in a purposeful manner. This requires a manipulation planning algorithm that computes a sequence of actions that achieve the desired rearrangement. Wilfong et al. [1] showed that this rearrangement planning problem is PSPACE-hard. The problem is particularly challenging due to the high dimensionality of the search space and the constraint that objects only move as a result of robot actions. Alami et al. [2] therefore introduced the distinction between *transfer* and *transit* actions. In transfer actions the robot manipulates objects, whereas in transit actions it only changes its own configuration.

Many existing works on rearrangement planning address the problem for pick-and-place transfer actions, where the outcome is simple to model and only a single object is moved at a time [2]–[8]. When transfer actions include non-prehensile manipulation their outcomes are more difficult to model and are therefore often abstracted by manually designed manipulation primitives [9]–[14].

Modeling primitives manually, however, is labor-intensive, and difficult to adjust to different robot embodiments and object types. Furthermore, it may restrict the robot's manipulation abilities unnecessarily, e.g. forcing a robot to push only a single object at a time. One approach to overcome these limitations is to perform a forward search over robot actions propagated through a physics model [15]–[18]. This allows the robot to manipulate multiple objects simultaneously and also move sliding or rolling objects. Early approaches [16], [17], however, suffer from poor exploration rates of the search space, since robot-centric actions are sampled uniformly at random, leading to little interaction with objects. Recently, King et al. [18] addressed this by combining a physics-based forward search with object-centric manipulation primitives. This biases the planner to apply the primitives, but does not limit it to it. Providing such primitives, however, is yet a limiting factor.

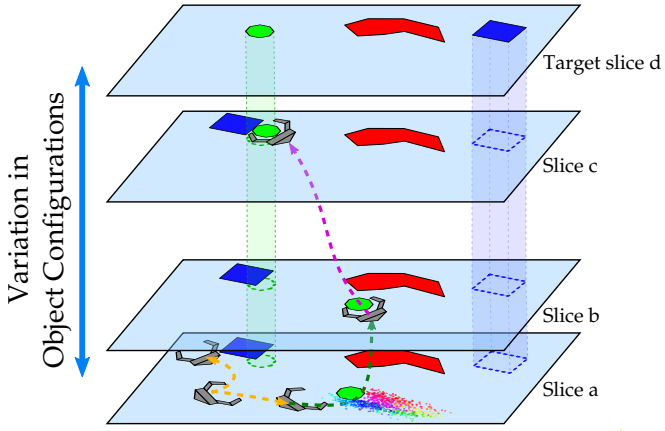


Fig. 1: Our planning algorithm rearranges movable objects on a plane (drawn in green and blue) in the presence of obstacles (red) using non-prehensile manipulation. The algorithm explores the composite configuration space of robot and objects in slices. Each slice corresponds to an object arrangement and is the subset of composite configurations with this arrangement.

In the present work, we adopt these ideas and combine a physics-based forward search with reinforcement learning and generative modeling. Our algorithm computes sequences of non-prehensile actions to rearrange multiple target objects among movable and non-movable obstacles on planar surfaces. In contrast to previous works, our approach requires no manually designed manipulation primitives. Furthermore, in our experiments, our approach achieves a better exploration rate and finds solutions faster than previous physics-based approaches.

II. Method

The concepts of our approach are illustrated in Fig. 1 and pseudo-code is provided in Algorithm 1. Our algorithm constructs a search tree $\mathcal{T}$ on the composite configuration space $\mathcal{C}_{r:m}$ of the robot r and $1 \dots m$ movable objects in a similar fashion as the kinodynamic RRT algorithm [19]. Similar to [17], it constructs the tree by forward propagating robot actions through a dynamic physics model. An action

[1] Robotics, Perception and Learning Lab (RPL), CAS, EECS, KTH Royal Institute of Technology, Stockholm, Sweden, E-mail: haustein, isacar, jastork, dani@kth.se

[2]GRAB Lab, Yale University, New Haven, USA, E-mail: kaiyu.hang@yale.edu

† J.A. Haustein and I. Arnekvist contributed equally to this work.

About this document: This document is part of the Workshop on Machine Learning in Robot Motion Planning at the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems. It is an extended abstract of a longer manuscript that is currently in submission to a journal. This longer manuscript, an illustrative video and additional supplementary material are available on `https://joshuahaustein.github.io/learning_rearrangement_planning/`

is parameterized by the velocity at which it moves the robot from one resting state to another. The physics model models object-to-object contacts as well as object dynamics, and predicts where the objects come to rest after an action.

A major difference to the kinodynamic RRT algorithm and [17] is that our algorithm explores the configuration space in *slices*. By slice $s = \{x \in \mathcal{C}_{r:m} | x_{1:m} = x^f_{1:m}\}$ we refer to a subset of the configuration space that shares the same object configurations $x^f_{1:m}$, and only differs in robot states. As illustrated in Slice a in Fig. 1, the planner's search tree may contain multiple configurations within the same slice. The motions between two such configurations are collision-free robot motions (transit actions) illustrated by yellow arrows. Moving between two configurations within a slice is a classical motion planning problem. In particular, steering the robot between two configurations within a slice ignoring collisions is simple for many types of robots. It is the purposeful transition to a different slice, a transfer action, that requires insight about manipulation.

To achieve a transfer action towards a target slice s^d of the configuration space, i.e. another object arrangement, our algorithm first randomly selects which object t it aims to transport (illustrated in Fig. 1 as green object). It then queries a learned generative model for a robot state within the current slice to steer to. This is illustrated in Fig. 1 by a set of colored points. Each point represents a *manipulation state* generated by the model and is a state from which the robot can transport the object towards its target within a single action. The colors encode different orientations of the robot.

After acquiring such a state from the generative model, the algorithm selects the closest robot state in its search tree that lies in the current slice. Thereafter, it computes a sequence of robot actions to move from this state to the generated state and forward propagates these actions through the physics model. In the example, this approach leads to contact and the resulting composite configuration lies in a new slice, Slice b. As there are no collisions with static obstacles, the new configuration is added to the search tree. Next, a learned policy is queried to provide a robot action that transports the target object towards its state in the target slice. Again, this action is forward propagated through the physics model and the search tree is updated accordingly.

The algorithm proceeds with this procedure iteratively. In each iteration it first randomly samples an object arrangement that identifies a target slice s^d. It then selects the slice in its search tree s^a that is most similar in terms of object configurations and proceeds as described. In some iterations instead of selecting to move an object, it focuses on growing the tree within a slice by sampling random robot states as targets, i.e. $t = r$. This continues until either a maximum number of iterations is exceeded or until the search tree contains a composite configuration for which the target objects are located within some goal regions.

Learning the Policy and the Generator

The generative model and the policy are trained in an obstacle-free world containing only the robot and a single object. We leave it to the planning algorithm to find robot states and actions that are feasible in the full problem containing multiple objects and obstacles.

Algorithm 1: The rearrangement planning algorithm.

Input: Start configuration $x^0 \in \mathcal{C}^{\text{free}}_{r:m}$, maximum number of iterations $n_{\max}$, goal region $\mathcal{G} \subset \mathcal{C}^{\text{free}}_{r:m}$
Output: Solution $[(x^0, u^0) \dots, (x^n, \perp)]$ or $\emptyset$

$s^0 \leftarrow$ SLICE(x^0) // Obtain start slice
$\mathcal{S} \leftarrow \{s^0\}$ // Set of all explored slices
$\mathcal{T} \leftarrow$ TREE(x^0) // Tree of explored states
for $j \leftarrow 1, \dots, n_{max}$ **do**
  $s^d \leftarrow$ SAMPLESLICE()
  $s^a \leftarrow \arg\min_{s \in \mathcal{S}} d_s(s, s^d)$ // Closest slice in $\mathcal{S}$
  $t \leftarrow$ RANDOMCHOICE$(\{r\} \cup \{1, \dots, m\})$
  if $t = r$ **then**
    $x^d_r \leftarrow$ SAMPLEUNIFORM$(\mathcal{C}^{free}_r)$
  else
    $x^d_r \leftarrow$ QUERYGENERATOR(s^a, s^d, t)
  // Retrieve closest state w.r.t. r in s^a
  $x^a \leftarrow \arg\min_{x \in s^a} d_{\mathcal{C}_r}(x_r, x^d_r)$
  // s^d_i denotes desired state of object i
  $x^d \leftarrow (x^d_r, s^d_1, \dots, s^d_m)$ // Store targets
  if SAMPLEUNIFORM$([0,1]) \leq p_{rand}$ **then**
    $u \leftarrow$ SAMPLEUNIFORM(U) // Random control
    // Extend $\mathcal{T}, \mathcal{S}$ using physics model
    $\perp, \mathcal{T}, \mathcal{S} \leftarrow$EXTENDSTEP$(x^a, u, \mathcal{T}, \mathcal{S})$
  else
    // Transit
    $u \leftarrow$ STEER(x^a_r, x^d_r)
    // Extend $\mathcal{T}, \mathcal{S}$ and return result x^b
    $x^b, \mathcal{T}, \mathcal{S} \leftarrow$EXTENDSTEP$(x^a, u, \mathcal{T}, \mathcal{S})$
    if $t \neq r$ **then**
      // Transfer
      $u \leftarrow$ QUERYPOLICY(x^b_r, x^b_t, x^d_t)
      $\perp, \mathcal{T}, \mathcal{S} \leftarrow$EXTENDSTEP$(x^b, u, \mathcal{T}, \mathcal{S})$
  if $\mathcal{T} \cap \mathcal{G} \neq \emptyset$ **then**
    return EXTRACTANDSHORTCUT$(\mathcal{T})$
return $\emptyset$

The planning algorithm applies its physics model to predict the outcome of any robot action, and can thus be applied to objects with various physical properties. Accordingly, we wish our policy and generative model to be applicable to objects with different physical properties as well. We achieve this, by parameterizing both by properties like the object's mass, approximate shape and friction. On the other hand, however, we learn both policy and generator only for a single robot at a time. Applying the planner to a different robot requires training a different policy and generator.

Learning the Policy: For simplicity, we choose to model the policy as a deterministic function $u_\theta = \pi_\theta(x_r, x_o, x^d_o, \nu_o)$, where θ denotes the parameters of the function approximator, x_r is the current state of the robot, x_o, x^d_o the current and the desired object state and ν_o the physical parameters of the object. We can train this policy by minimizing the loss function

$$L_\theta(x_r, x_o, x^d_o, \nu_o) := \mathbb{E}_{x'_o}[d(x'_o, x^d_o) \mid u_\theta, x_r, x_o, \nu_o] \quad (1)$$

over some training set of tuples (x_r, x_o, x_o^d, ν_o) by performing deterministic policy gradient descent [20]. This loss expresses the expected distance $d(x_o', x_o^d)$ between the resulting object state x_o' and the desired object state x_o^d, when executing the policy action u_θ from robot state x_r.

To perform this training, we need a differentiable model of the loss L_θ w.r.t θ. For this, we choose the distance function for the $SE(2)$ state of the object to be of the form $d(x_o', x_o^d) = \sum_{j=1}^{3} \sigma_j (x_{oj}' - x_{oj}^d)^2$ with positive weights $\{\sigma_j\}_{j=1}^3$ and where x_{o1}, x_{o2} is the object's position and x_{o3} its orientation. With this the loss becomes

$$L_\theta(x_r, x_o, x_o^d, \nu_o) = \sum_{j=1}^{3} \sigma_j \left(\mathrm{Var}_{x_o'}(x_{oj}') + \left(\mathbb{E}_{x_o'} \left[x_{oj}' \right] - x_{oj}^d \right)^2 \right), \quad (2)$$

where both variance and expected value are conditioned on $x_r, x_o, \nu_o, u_\theta$. This means we can express the loss L_θ as a function of forward models

$$f_\mu(x_r, x_o, \nu_o, u) = \mathbb{E}_{x_o'} \left[x_o' | x_r, x_o, \nu_o, u \right] \quad (3)$$

and

$$f_{\sigma^2}(x_r, x_o, \nu_o, u) = \mathrm{Var}_{x_o'}(x_o' | x_r, x_o, \nu_o, u). \quad (4)$$

In particular, if these forward models are differentiable w.r.t. to the action argument u, we can apply the chain rule in Eq. (2) to obtain the gradient of L_θ w.r.t. θ. Such models f_μ and f_{σ^2} can be learned in a supervised manner from the physics model of the planner by generating a training set $D = \{(x_r^j, x_o^j, \nu_o^j, u^j, (x_o')^j) \mid j \in \{1, \ldots, n\}\}$ of n robot-object interactions.

Learning the generative model: Given the trained policy, the purpose of the generative model is to provide robot state samples from which the policy can be applied. Given an object state x_o, a target state x_o^d and the object's properties ν_o, we can find such states by minimizing the loss $L_\theta(x_r, x_o, x_o^d, \nu_o)$ w.r.t. to x_r. However, since the policy does not take any obstacles into account, limiting the model to providing the minima of L_θ is too restrictive. Instead, we opt to define a distribution over robot states with wider support

$$p(x_r | x_o^d, x_o, \nu_o) = \frac{1}{Z} e^{-\lambda L_\theta(x_r, x_o, x_o^d, \nu_o)}, \quad (5)$$

where Z is a normalization constant and $\lambda \in \mathbb{R}^+$ a manually selected parameter. The distribution becomes narrower for large choices of λ and wider for small ones.

We can generate state samples from this distribution using the Markov Chain Monte Carlo (MCMC) method. This, however, is computationally too expensive to be applied within the planner. Instead, we use this method to generate samples $x_r \sim p_{\text{MCMC}}$ offline and train a (conditional) generative adversarial network (GAN) [21] to mimic this distribution online. We train the GAN by optimizing

$$\min_{\varphi_G} \max_{\varphi_D} V(\varphi_D, \varphi_G) \quad (6)$$

where V is defined as

$$V(\varphi_D, \varphi_G) = \mathbb{E}_{x_r \sim p_{\text{MCMC}}} \left[\log(D_{\varphi_D}(x_r, x_o, x_o^d, \nu_o)) \right] + \mathbb{E}_z \left[\log(1 - D_{\varphi_D}(G_{\varphi_G}(z, x_o, x_o^d, \nu_o), x_o, x_o^d, \nu_o)) \right] \quad (7)$$

Here, φ_G parameterizes the generator network G, and φ_D the discriminator D. Once trained, we only need to perform a single forward pass through the generator network to provide a sample from the above distribution.

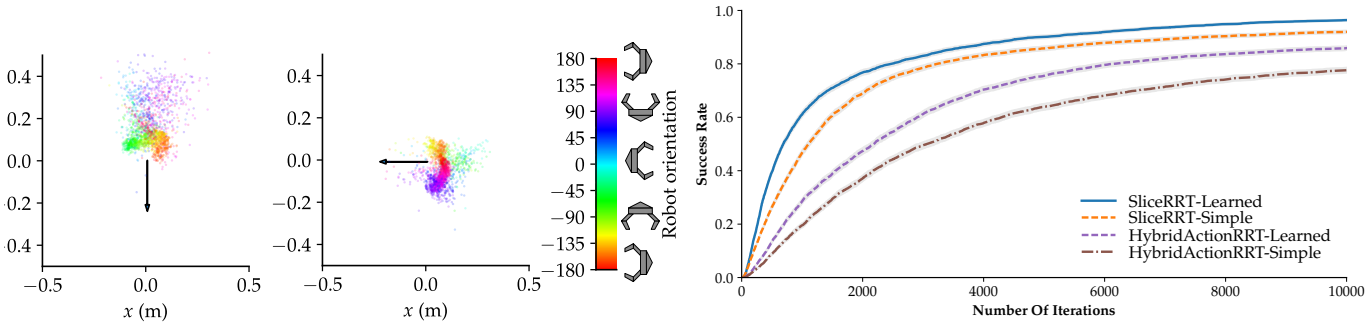
III. Experiments

We implemented and evaluated our approach for a holonomic planar robot endowed with a configuration space in $SE(2)$. As shape of the robot, we chose the shape of the gripper shown in Fig. 1 and in Fig. 2a on the right.

Fig. 2a shows the robot states that the trained generative model provides for this robot when queried to push an object. The states are predominantly oriented such that the robot traps the object between its fingers. We achieve this behavior by training the forward models f_μ and f_{σ^2} under observation noise of the physical properties ν_o and the object state x_o. As a result the model f_{σ^2} learns that there is variance that depends on the robot state x_r and action u. As can be seen in Eq. (2), the loss L_θ is minimized, if the variance of the successor state is minimized. Thus, the generator produces its states accordingly. In the states shown in Fig. 2a the object is trapped between the fingers and the object thus moves with the robot, resulting in low variance successor states.

We compare our algorithm to King et al.'s [18] HybridActionRRT algorithm and our learned models to a simple manually defined generative model and policy. The HybridActionRRT algorithm differs from ours in that it does not structure its search tree in slices, and requires object-centric manipulation primitives. For this we define primitives by concatenating a straight line approach to a state generated by a generator with the respective policy action for that state. Although this is a similar behavior as applied in our algorithm's extension step, HybridActionRRT treats the primitive as black box and queries the policy prior to modeling the approach action. As simple generator we sample a position located next to the object, and as simple policy we steer the robot in a straight line towards the object's target position.

We run the algorithms on various rearrangement problems that include different numbers of obstacles and objects with varying physical properties. Fig. 2b shows the planning success rates as a function of iterations of the algorithms. Our algorithm achieves the steepest increase and highest final success rate after 10000 iterations. The learned models outperform the simple hand-defined models. We observe in particular that the learned models perform significantly better than the simple ones on problems where objects can slide. This highlights that taking an object's physical properties into account is highly beneficial, which is difficult to model manually. For more experimental evaluation we refer to our full manuscript.



(a) Learned manipulation states for a gripper-like robot. Each colored point represents a robot pose relative to the object generated by the generator. The arrow represents the desired push of an object located in the origin. For most robot poses the palm is facing the object, and the robot is placed opposite to the pushing direction. The mapping from colors to orientations is shown on the right.

(b) Planning success rate as function of number of iterations. For any given number of iterations n, the corresponding success rate can be interpreted as an empirically determined probability that an algorithm successfully finds a solution within n iterations. The shaded areas show the 95% Wilson confidence interval of this probability.

Fig. 2: Samples of robot states from our generative adversarial network and success rate curves of the evaluated algorithms. Consult our online supplementary material for more results including a different robot shape, and a video showing how the samples are influenced by different arguments to the generator.

IV. Discussion And Conclusion

In the present work, we designed a new algorithm that combines sampling-based motion planning with reinforcement learning and generative modeling for non-prehensile rearrangement planning. Learning a policy and generative model provides valuable guidance to the sampling-based search without the need of manually defining manipulation primitives. This in combination with the physics modeling allows the approach to produce solutions where the robot applies a versatile set of manipulation actions.

Our planner assumes a deterministic physics model and its solutions are therefore likely to fail when executed on a real robot. While this has been addressed by related previous works [22]–[24], we believe that combining such approaches with learning an uncertainty reducing policy is a promising direction for future work.

Lastly, to solve challenging instances of rearrangement problems efficiently, a higher level logic is required that guides the physics-based search. The question whether an object obstructs another strongly depends on a robot's embodiment. Thus, we believe that learning a high-level policy conditioned on this is a promising direction for future work.

References

[1] G. Wilfong, "Motion planning in the presence of movable obstacles," *Annals of Mathematics and Artificial Intelligence*, vol. 3, no. 1, pp. 131–150, March 1991.

[2] T. S. Rachid Alami, Jean-Paul Laumond, "Two manipulation planning algorithms," in *Proceedings - Workshop on Algorithmic Foundations of Robotics*, 1994.

[3] T. Siméon, J. P. Laumond, J. Cortés, and A. Sahbani, "Manipulation planning with probabilistic roadmaps," *International Journal of Robotics Research*, vol. 23, no. 7-8, pp. 729–746, August 2004.

[4] J. Ota, "Rearrangement of multiple movable objects - integration of global and local planning methodology," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 1962–1967, 2004.

[5] M. Stilman, J. U. Schamburek, J. Kuffner, and T. Asfour, "Manipulation planning among movable obstacles," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 3327–3332, April 2007.

[6] A. Krontiris, R. Shome, A. Dobson, A. Kimmel, and K. Bekris, "Rearranging similar objects with a manipulator using pebble graphs," *Proceedings - IEEE-RAS International Conference on Humanoid Robots*, pp. 1081–1087, November 2015.

[7] A. Krontiris and K. Bekris, "Dealing with Difficult Instances of Object Rearrangement," *Proceedings - Robotics: Science and Systems*, July 2015.

[8] A. Krontiris and K. E. Bekris, "Efficiently solving general rearrangement tasks: A fast extension primitive for an incremental sampling-based planner," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 3924–3931, May 2016.

[9] M. R. Dogar and S. S. Srinivasa, "A planning framework for non-prehensile manipulation under clutter and uncertainty," *Autonomous Robots*, vol. 33, no. 3, pp. 217–236, October 2012.

[10] J. Barry, L. P. Kaelbling, and T. Lozano-Perez, "A hierarchical approach to manipulation with diverse actions," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 1799–1806, November 2013.

[11] J. Barry, K. Hsiao, L. P. Kaelbling, and T. Lozano-Pérez, "Manipulation with Multiple Action Types," *Proceedings - International Symposium on Experimental Robotics*, pp. 531–545, June 2012.

[12] G. Havur, G. Ozbilgin, E. Erdem, and V. Patoglu, "Geometric rearrangement of multiple movable objects on cluttered surfaces: A hybrid reasoning approach," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 445–452, May 2014.

[13] S. Jentzsch, A. Gaschler, O. Khatib, and A. Knoll, "MOPL: A multimodal path planner for generic manipulation tasks," *Proceedings - IEEE International Conference on Intelligent Robots and Systems*, pp. 6208–6214, September 2015.

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

[15] C. Zito, R. Stolkin, M. Kopicki, and J. L. Wyatt, "Two-level RRT planning for robotic push manipulation," *Proceedings - IEEE International Conference on Intelligent Robots and Systems*, pp. 678–685, October 2012.

[16] J. E. King, J. A. Haustein, S. S. Srinivasa, and T. Asfour, "Non-prehensile whole arm rearrangement planning on physics manifolds," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 2508–2515, June 2015.

[17] J. A. Haustein, J. King, S. S. Srinivasa, and T. Asfour, "Kinodynamic randomized rearrangement planning via dynamic transitions between statically stable states," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 3075–3082, June 2015.

[18] J. E. King, M. Cognetti, and S. S. Srinivasa, "Rearrangement planning using object-centric and robot-centric action spaces," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 3940–3947, May 2016.

[19] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, May 2001.

[20] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," *Proceedings - International Conference on Machine Learning*, pp. I–387–I–395, June 2014.

[21] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley,

S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Proceedings - Advances in Neural Information Processing Systems*, pp. 2672–2680, December 2014.

[22] A. M. Johnson, J. King, and S. Srinivasa, "Convergent planning," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 1044–1051, July 2016.

[23] M. C. Koval, J. E. King, N. S. Pollard, and S. S. Srinivasa, "Robust trajectory selection for rearrangement planning as a multi-armed bandit problem," *Proceedings - IEEE International Conference on Intelligent Robots and Systems*, pp. 2678–2685, September 2015.

[24] J. E. King, V. Ranganeni, and S. S. Srinivasa, "Unobservable Monte Carlo planning for nonprehensile rearrangement tasks," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 4681–4688, May 2017.