Online Tool Selection with Learned Grasp Prediction Models

Khashayar Rohanimanesh[1] and Jake Metzger[2] and William Richards[1] and Aviv Tamar[3*]

***Abstract*— Deep learning-based grasp prediction models have become an industry standard for robotic bin-picking systems. To maximize pick success, production environments are often equipped with several end-effector tools that can be swapped on-the-fly, based on the target object. Tool-change, however, takes time. Choosing the *order* of grasps to perform, and corresponding tool-change actions, can improve system throughput; this is the topic of our work. The main challenge in planning tool change is *uncertainty* – we typically cannot see objects in the bin that are currently occluded. Inspired by queuing and admission control problems, we model the problem as a Markov Decision Process (MDP), where the goal is to maximize expected throughput, and we pursue an approximate solution based on model predictive control, where at each time step we plan based only on the currently visible objects. Special to our method is the idea of *void zones*, which are geometrical boundaries in which an unknown object will be present, and therefore cannot be accounted for during planning. Our planning problem can be solved using integer linear programming (ILP). However, we find that an approximate solution based on sparse tree search yields near optimal performance at a fraction of the time. Another question that we explore is how to measure the performance of tool-change planning: we find that throughput alone can fail to capture delicate and smooth behavior, and propose a principled alternative. Finally, we demonstrate our algorithms on both synthetic and real world bin picking tasks.**

I. Introduction

Automated bin picking has gained considerable attention from manufacturing, e-commerce order fulfillment, and warehouse automation. The problem generally involves grasping of a diverse set of novel objects, which are often packed randomly inside a bin (Figure 1(b)). A common model-free approach is based on learning *grasp prediction models* – deep neural networks that map an image of the bin to success probabilities for different grasps [1], [2] (see Figures 1(c) and 1(d) for examples of learned grasp prediction models for two different vacuum suctions of diameters 30mm and 50mm).

In order to handle a diverse range of objects, robotic cells are often equipped with a tool changer mechanism (Figure 1(a)), allowing the robot to select a new end-effector from a set of available end-effectors (e.g., vacuum end-effectors varying in size, antipodal end-effectors) and swap it with the current one automatically in real time (as it can be observed in Figure 1, the grasp prediction model for 50mm vacuum suction (Figure 1(d)) generally places more probability mass over the larger objects compared to the grasp prediction model for 30mm vacuum suction (Figure 1(c)), given the input bin image Figure 1(b)). Choosing the right tool for each object can increase the pick success, potentially leading to improved throughput. Tool changing, however, comes at a cost of cycle time: navigating the end-effector to the tool changing station, and performing the swap.[1]

In common picking tasks, the agent is free to choose the *order* of objects to pick, and respectively, the order of tool changes. Thus, by carefully planning the picking order, we can potentially improve picking efficiency, by, e.g., using the same tool repeatedly for several objects. This is our main objective in this work. Optimizing tool selection, however, is challenging due to several reasons. Typically, some objects in the bin are occluded, and even objects that are currently visible may move unexpectedly due to grasp attempts of nearby objects, affecting their optimal tool selection in the future. Furthermore, even if the objects positions were known in advance, the complexity of computing the optimal picking order scales exponentially with the planning horizon, and is effectively intractable for real-time operation. Our goal in this work is to develop a scalable, well-performing, and fast method for approximately optimal tool selection.

We present a general formulation of this stochastic decision making problem, which we term *Grasp Tool Selection Problem* (GTSP), as a Markov Decision Process (MDP) (Section II). In practice, solving this MDP is difficult due to its large state space and difficult-to-estimate transition dynamics. To address this, we introduce an approximation of the problem where we: (1) replace the discounted horizon problem with a receding horizon Model Predictive Control (MPC); (2) In the inner planning loop of the MPC component, we replace the stochastic optimization with an approximate deterministic problem that does not require the complete knowledge of the true transition dynamics. We show that this deterministic problem is an instance of integer linear programming (ILP), which can be solved using off-the-shelf software packages. However, we further show that an approximate solution method based on a sparse tree search improves the planning speed by orders of magnitude, with a negligible effect on the solution quality, and is fast enough to run in real time.

Our approach decouples grasp prediction learning from

*A. Tamar is funded by the European Union (ERC, Bayes-RL, Project No. 101041250). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council Executive Agency. Neither the European Union nor the granting authority can be held responsible for them.

[1]Osaro Inc., San Francisco, CA 94105, USA {khash,will}@osaro.com

[2]Accenture Inc., San Francisco, CA 94105, USA jacob.c.metzger@accenture.com

[3]Technion – Israel Institute of Technology, Haifa, Israel avivt@technion.ac.il

[1]Mounting several tools on the same end effector, while possible [1], can be difficult, as, for example, different vacuum suction cups would require multiple hosing. A tool change station is a more scalable approach.

(a) (b) (c) (d)

Fig. 1: (a) Tool-changer station hosting different vacuum end-effectors of various sizes. The current selection is already mounted on the robot; (b) Example of a bin containing an assortment of various objects; (c) grasp prediction model for 30mm vacuum suction; and (d) grasp prediction model for 50mm vacuum suction (green spectrum denotes higher grasp scores). Note that the larger suction cup model generally predicts higher success over the larger objects compared to the smaller cup model.

tool selection planning – we only require access to a set of pre-trained grasp prediction models for each individual end-effector. Thus, our method can be applied ad hoc in production environments whenever such models are available, and is scalable in the sense that combining different end-effector models does not require re-training the system. In our experiments, on both synthetic and real-world environments, we demonstrate that our planning method significantly improves system throughput when compared to heuristic selection methods. Another novel contribution of this work is the derivation of a set of metrics for benchmarking tool selection algorithms, based on practical considerations relevant to bin picking.

A. Related Work

There is extensive literature on learning grasp prediction models [3], [4], [5]. To the best of our knowledge, few previous studies considered tool change optimization. The closest related work is [5] in the context of controlling an ambidextrous robot for bin picking problem. That work focused on scaling the learning of the grasp prediction models, and not on the tool selection problem. In their approach, the best tool is selected greedily based on the grasp prediction scores generated by the end-effector grasp prediction models. Tackling problems with uncertain transitions by replanning using deterministic models is a common approach in planning [6] and robotics, where it is commonly referred to as model predictive control [7], [8]. To our knowledge, this work is the first application of this idea to tool change optimization with learned grasp models. Several studies focused on planning with deep visual predictors [9], [10], [11], [12], where a deep visual predictive model is learned and combined with MPC. Our work differs from these approaches in that we perform planning directly in grasp proposal space, based on our void zone approximation.

II. Grasp Tool Selection Problem (GTSP) Formulation

We assume a planar workspace $\mathcal{I}$, discretized into a grid of $W \times H$ points. For example, $\mathcal{I}$ could be the bin image, as in Figure 1(b), where every pixel maps to a potential grasp point in the robot frame. A grasp proposal evaluates the probability of succeeding in grasping at a particular point. Formally, a grasp proposal is a tuple $\omega := \{\mathcal{E}, u, \rho\}$, where $\mathcal{E}$ is an end-effector, $u \in \mathcal{I}$ is a position to perform a grasp (e.g., a pixel in the image), and $0 \leq \rho \leq 1$ is the probability of a successful grasp when the end-effector $\mathcal{E}$ is used to perform a grasp on position u. We also use the notations $\omega^{\mathcal{E}}, \omega^{u}$, and ω^{ρ} when referring to individual elements of a grasp proposal ω.

A grasp prediction model $\Gamma_{\mathcal{E}} : \mathcal{I} \rightarrow \{\omega_i\}_{i=1}^{W \times H}$ gives a set of grasp proposals for an input image $\mathcal{I}$ and end-effector $\mathcal{E}$. In practice, only a small subset of grasp proposals yield good grasps. Thus, without loss of generality, we denote $\Gamma_{\mathcal{E}}^{k}(\mathcal{I}) = \{\omega_i\}_{i=1}^{k}$, limiting the model only to the k best grasps (in terms of grasp success probability). Given a set of n end-effector grasp proposal models $\{\Gamma_i\}_{i=1}^{n}$[2], we define the *grasp plan space* $\Omega := \cup_{i=1}^{n} \Gamma_i^k(\mathcal{I})$, which denotes the space of all plannable grasps. We will further denote by Ω_t the grasp plan space that is available at time t of the task.

a) Grasp Tool Selection Problem (GTSP):: We model the problem as a Markov Decision Process (MDP [13]) $\mathcal{M} \equiv \langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T} \rangle$ defined as follows: $\mathcal{S}$ is a set of states, where each $s = \langle \Omega, \mathcal{E} \rangle \in \mathcal{S}$ consists of the current grasp plan space and the current end-effector on the robot. We denote by s^{Ω} and $s^{\mathcal{E}}$ the individual elements of state s. The action space $\mathcal{A} = \Omega$ is the set of all plannable grasp proposals. The reward balances pick success and tool change cost, and is:

$$\mathcal{R}(s, \omega) = \omega^{\rho} + c\,\mathbf{1}(s^{\mathcal{E}} \neq \omega^{\mathcal{E}}), \tag{1}$$

where ω^{ρ} is the grasp success score, $c < 0$ reflects a negative reward for tool changing, and $\mathbf{1}(\cdot)$ is the indicator function[3]. The state transition function $\mathcal{T}(s_t, \omega_t, s_{t+1}) \rightarrow [0, 1]$ gives the probability of observing the next state s_{t+1} after executing grasp proposal ω_t in state s_t. As a result of performing a grasp, and depending on the grasp outcome, an object is removed from the bin and some other object randomly appears at the position of the executed grasp and new graspable positions will be exposed. The optimal policy $\pi^* : \mathcal{S} \rightarrow \Omega$ is defined as: $\pi^* = \text{argmax}_{\pi}\, \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, \omega_t) | \omega_t \sim \pi(s_t)\right]$.

[2]For simplifying notations we interchangeably use Γ_i to refer to an end-effectors grasp proposal model $\Gamma_{\mathcal{E}_i}$

[3]One can also add a cost for the distance travelled between consecutive grasp proposals, or the proximity of the end-effector to the tool changer station. In this work, however, we found this simpler reward to suffice.

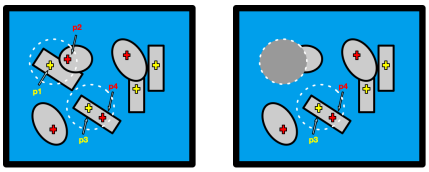

Fig. 2: (Left) examples of cases where performing a grasp proposal would invalidate some other grasp proposals (p_1 invalidates p_2, and p_3 invalidates p_4); (Right) updated grasp plan space for the next horizon step by voiding the impacted grasp proposal (in this case voiding p_2 as a result of committing to the grasp proposal p_1).

III. Approximate GTSP Solution

Solving GTSP is difficult for following reasons: **(1) Prediction:** we do not know the true state transitions, as they capture the future grasps that will be made possible after a grasp attempt, which effectively requires a prediction of objects in the bin that are not directly visible (see, e.g., Figure 1(b)). Although several studies investigated learning visual predictive models (see [9], [10], [11], [12]), learning such models in production environments with a high variability of objects is not yet practical. **(2) Optimization:** even if the state transitions were known, the resulting MDP would have an intractable number of states, precluding the use of standard MDP solvers.

We address (1) by replacing the stochastic optimization with an approximate deterministic problem that does not require the complete knowledge of the true transitions, based on the idea of *void zones*. We address (2) by replacing the infinite horizon discounted horizon problem with an online solver, which at each time step chooses the grasp that is optimal for the next several time steps; we term this part the model predictive control (MPC) component. We propose two computational methods for solving the short horizon problem in the inner MPC loop, either accurately, based on integer linear programming, or approximately, using a simple tree search method.

A. Approximate Prediction using Void Zones

We seek to replace the stochastic (and unknown) transitions in GTSP by deterministic dynamics, such that the solution of the deterministic problem will yield a reasonable approximation of the true optimal controller. Our approach is based on the idea of a *void zone* – not allow a grasp that is in very close proximity to any previous grasp, as movement of objects in the bin resulting from the previous grasp attempt could render a future grasp in its close vicinity impossible. We motivate void zones with the following working hypothesis:

As long as the objects are sufficiently small, when a grasp is attempted, the set of grasp proposals that are sufficiently distant from the attempted grasp position will remain valid in the next state.

This observation is illustrated in Figure 2(Left), for a bin picking problem with two end-effectors. The grasp proposals are color coded for each end-effector. In some cases, grasp proposals lie over different objects with one object partially occluding the other (e.g., p_1 and p_2). In other cases, two or more grasp proposals might lie on the same object. In either case, performing one of the grasp proposals will invalidate some other grasp proposal and hence those proposals should not be available to the planner in the next steps.

We define void zones based on the Euclidean distance:

Definition 1 (*l-separation*)**.** *Let* $d_{i,j} = \left\|\omega_i^u - \omega_j^u\right\|$ *denote the Euclidean distance on the plane between grasp proposals* ω_i *and* ω_j*. A pair of grasp proposals* $\langle \omega_i, \omega_j \rangle$ *is called* l-separated *if* $d_{i,j} > l$*. We refer to* l *as void radius and use the notation* $\Psi_l(\omega)$ *to refer to a set of grasp proposals which are* l*-separated from* ω*. Note that by definition* $\omega \notin \Psi_l(\omega)$*.*

Based on the above definition, we can formally define deterministic dynamics in GTSP, which we will henceforth refer to as *GTSP-void*. At state $s_t = \langle \Omega_t, \mathcal{E}_t \rangle$, taking action ω_t results in a next state,

$$\begin{aligned} s_{t+1} =& \langle \Omega_{t+1}, \omega_t^{\mathcal{E}} \rangle, \\ \Omega_{t+1} =& \{\omega \mid \omega \in \Omega_t \wedge \omega \in \Psi_l(\omega_t)\} \end{aligned} \tag{2}$$

That is, the end-effector in the next state is as chosen by ω_t, and the grasp plan space is updated to exclude all grasp proposals within the void zone.

As shown in Figure 2, by setting the void zone large enough, we can safely ignore the local changes as a result of executing a grasp. Obviously, using void zones comes at some cost of sub-optimality – as we ignore possible future grasps inside the void zones. To mitigate this cost, we propose a model predictive control (MPC) approach. At every step, the current observation (i.e., bin image $\mathcal{I}$) is fed to the set of pre-trained end-effector models to obtain the plan space Ω_t. Next, we solve the corresponding GTSP-void problem with some fixed horizon H, and the first step of the plan ω is executed. Replanning at every step allows our method to adapt to the real transitions observed in the bin. Next, we propose two methods for solving the inner GTSP-void problem within our MPC.

B. Optimization using Integer Linear Programming

In this section we show that the GTSP-void problem can be formulated as an integer linear program (ILP). To motivate this approach, note that GTSP-void with horizon H seeks to find a trajectory of H l-separated grasp proposals in Ω_t with the highest accumulated return. This motivates us to think of the problem as a walk over a directed graph generating an *elementary path* of length H of l-separated grasp proposals with the highest return, where the nodes of the graph are the grasp proposals in the current state, s_t^{Ω}, and the directed edges represent the order at which the grasp proposals are executed.

Our formulation is mainly inspired by the ILP formulation of the well known *Travelling Salesman Problem* (TSP) [14] with the following changes: (1) the main objective is modified to finding an elementary path of length h with maximal return, anywhere on the graph (as opposed to the conventional *tour* definition in TSP); (2) addition of the l-separation constraints to enforce voiding; (3) a modification

of *Miller-Tucker-Zemlin* sub-tour elimination technique [15] for ensuring the path does not contain any sub-tour. Due to lack of space, we refer the reader to our technical report [16], which contains a detailed explanation and pseudo-code for the ILP algorithm.

The ILP formulation allows us to use off-the-shelf optimization packages, such as Gurobi [17], to solve GTSP-void accurately. However, as we report in Section IV, we found that the solution time can be too slow for ILP be a practical component in the bin picking pipeline. In the next section, we propose a simple approximate solution that is much faster.

C. Approximate Optimization using Sparse Tree Search

We now present a simple alternative to ILP for approximately solving GTSP-void based on a *sparse tree search* (STS). Our approach, outlined in Algorithm 1, performs a tree search of depth H where tree node expansion takes place over a *sparse* subset of grasp proposals respecting the *l-separation* constraint (see Definition 1 in Section III-A). At every search step a node is expanded using a sparse subset of available grasp proposals (line 6). In our approach we use the union of top k grasp proposals per end-effector according to the grasp proposal scores ω^ρ. The parameter k – hereafter the *sparsity factor* – determines the sparsity of the subset of grasp proposals for the tree search node expansion (while expanding over the set of all available grasp proposals at every node is possible and optimally solves the problem in theory, in practice it makes the planning significantly slow and hence is not suitable for real world production environments). The algorithm then recursively calculates a sub-plan rooted at that node for a receding horizon of H (line 9), accumulates the results of each recursion and computes the value of the sub plan (line 10), and it returns the best sub-plan and its value among all the sub-plans calculated at that horizon.

Algorithm 1 Sparse Tree Search (STS)

1: **function** STS(s_t, H, c, l, k) ▷ current state $s_t = \langle \Omega_t, \Gamma_t \rangle$; horizon H, tool changing costs $c < 0$, void radius l, k is the sparsity factor where it specifies the top k grasp proposals per end-effector to be included for the search tree node expansion
2: **if** $H = 0$ **then**
3: return $(\_, 0)$
4: **end if**
5: $\Xi \leftarrow \emptyset$
6: Let $\Lambda_t^k \subset \Omega_t$ be the union of the top k grasp proposals per end-effector (in terms of grasp score ω^ρ) available in Ω_t
7: **for** $\omega \in \Lambda_t^k$ **do** ▷ $\omega = \langle \mathcal{E}, u, \rho \rangle$
8: $s_{t+1} = \mathcal{T}_l(s_t, \omega)$ ▷ forward dynamics (Eq. 2)
9: $(\omega^+, v_{\omega^+}) \leftarrow$ STS($s_{t+1}, H-1, c, l, k$)
10: $v_\omega = \mathcal{R}(s_t, \omega) + v_{\omega^+}$ ▷ reward function (Eq. 1)
11: $\Xi \leftarrow \Xi \cup (\omega, v_\omega)$
12: **end for**
13: **return** $\text{argmax}_{v_\omega}(\Xi)$
14: **end function**

IV. Experiments

We divide our presentation to: (1) an investigation of the optimization performance of STS vs. ILP, and (2) a demonstration that our approach can significantly improve bin picking performance in realistic settings. For answering (1), we recall that STS and ILP are solvers for the same GTSP-void problem, and the comparison can be decoupled from the physical setting that underlies GTSP-void. We therefore designed a synthetic GTSP-void problem generator to perform a rigorous comparison. For demonstrably proving (2), we performed real-world experiments on an industrial bin picking station that is currently in production. We opted not to perform physical simulations for (2) since, at present, a faithful physical simulation of industrial bin picking is very challenging to produce. For example, suction cups, which are preferred due to their speed and form, are difficult to simulate even in state of the art simulators such as Mujoco [18].

a) Synthetic Experiments: In the first set of experiments we compared the two GTSP-void solvers outlined in Sections III-B and III-C. Our goal is to answer the following question: How do the ILP and STS solvers compare in terms of the optimization quality and speed?

We crafted a synthetic tool selection problem generator as follows. A problem instance $\mathcal{T}$ is generated by first selecting the number of end-effectors and then, for each end-effector, we generate a random set of grasp proposals over a fixed grid resolution H×W (we used H=70, W=110 in our experiments) [4] To generate realistic grasp proposals, we first choose $n = 25$ random object positions, uniformly sampled on the grid. Next, for each-end effector we generate random Gaussian kernels with randomized scale and standard deviation, centered on each object position. The resulting grasp proposal grid for each pixel gives a higher probability of success to pixels that are closer to an object center (for more details and examples of the synthetic grasp maps refer to our technical report [16]).

In our experiments, we report the *advantage* metric, defined as $\text{Adv}(\mathcal{T}) = \text{ILP}(\mathcal{T}) - \text{STS}(\mathcal{T})$, where $\text{ILP}(\mathcal{T})$ and $\text{STS}(\mathcal{T})$ denote the return of the best plan in each algorithm calculated for horizon H using the reward function defined in Equation 1, respectively. These values represent the long-horizon performance of each algorithm. We report $\text{Adv}(\mathcal{T})$ which measures the advantage in optimization quality of ILP over STS, and the *planning time* for each algorithm, both evaluated on our Python implementation of STS, and the commercial Gurobi [17] ILP solver, using MacBook Pro 2.8 GHz Quad-Core Intel Core i7 hardware. We used a fixed void radius $l = 20$ and swap cost $c = -0.2$, and report results over $n = 100$ random problem instances as defined above.

Figure 3 shows our results for number of end-effectors 2 (Left 3 columns), and 3 (Right 3 columns). In each group, the top row shows the *advantage* results over STS sparsity factor $k \in \{1, 2, 3\}$ and various horizons. The bottom row shows the planning time for each case. In terms of quality, STS is observed to perform as well as ILP or marginally worse. In terms of planning speed, STS is orders of magnitude

[4] We generate grasp proposal sets directly, without requiring an image (cf. Section II).

faster in both cases. It can be also observed that reducing k significantly improves speed with a negligible effect on quality. These results motivate us to use STS in our real world application.

b) Real World Experiments: We conducted experiments to evaluate the performance of various grasp tool selection algorithms, and to validate the adequacy of the proposed tool changing score in capturing efficiency. First, we compare the MPC-STS with a set of heuristic baselines. Next, we compare the performance of MPC-STS and heuristics baselines against experiments where only a single end-effector was used (no tool changing allowed). We also conduct a series of ablations on MPC-STS in terms of its void radius and max horizon (i.e., H). Before we present our results, we first discuss how real-world performance should be best evaluated.

c) Metric Definitions: Our primary goal is to minimize the cost associated with changing tools, yet still maximize pick success. One way to measure performance is by *grasp throughput* – the average number of successful picks in a unit time. However, grasp throughput does not correctly penalize strategies that execute many failed grasps quickly, which can be inappropriate for scenarios where items may become damaged as a result of repeated, aggressive picking.

To address this, we propose a combined score based on pick success rate (PSR), and tool consistency rate (TCR), defined as: $\text{PSR} = \frac{\text{PS}}{\text{PA}}, \quad \text{TCR} = 1 - \frac{\text{TC}}{\text{PA}}$, where PS is the pick success count, PA is the pick attempt count, and TC is the tool change count (here, we do assume that there is no more than one tool change per pick attempt). Ideally, we would like both scores to be high. Also, the PSR and TCR should be balanced according to the time cost of tool change compared to the time cost of a failed grasp. We posit that the following β-TC-score captures these desiderata,

$$\beta\text{-TC-score} = \frac{(1+\beta^2) * \text{PSR} * \text{TCR}}{\beta^2 \text{PSR} + \text{TCR}}, \tag{3}$$

where β is analogous to an F-$beta$ score [19]. We recommend that β be set to the *opportunity cost* of a single tool change – the approximate number successful picks that could have been completed in the time it takes to execute a tool change. For our setup, we estimated β to be 0.33.

We further motivate the β-TC-score with a numerical example. Consider two tool selection algorithms A, B, being evaluated over a similar scenario (independently) with two items in the bin (i.e., each needs two successful picks to clear the bin), and producing the following event sequences:

$$A : T\,F\,F\,F\,S\,T\,S, \quad B : T\,F\,F\,F\,S\,F\,F\,F\,S$$

where T is a tool change event, F is pick fail, and S is pick success. Assume each pick attempt takes 1 second, and each tool change takes 3 seconds. In the above trajectories, both A and B have the same throughput (2 successes per 11 seconds), but we have a preference for A due to less failed pick attempts (A has 3 vs. 6 in B). In each case we have:

$$PSR(A) = 2/5, \quad TCR(A) = 3/5,$$
$$PSR(B) = 2/8, \quad TCR(B) = 7/8.$$

For small values of β (e.g., $\beta < 1.0$), TC score places more importance on PSR. At the extreme $\beta = 0$, we have TCR = PSR, ignoring the cost of tool change. In this case, scores are:

$$\beta\text{-TC}(A) = PSR(A) = 0.4 > \beta\text{-TC}(B) = PSR(B) = 0.25.$$

For larger values of β (e.g., $\beta > 1.0$), TCR gains more importance and in the limit of $\beta = \infty$, we have TC = TCR. In the above example, for $\beta = 2$, we have:

$$\beta\text{-TC}(A) = 0.545 < \beta\text{-TC}(B) = 0.583$$

As we suggested above, a good balance is obtained when selecting β to be the opportunity cost. Here, the overall pick success rate is $(2+2)/(5+8) \approx 0.3$, and therefore the opportunity cost is slightly less than 1. For $\beta = 1$ we obtain:

$$\beta\text{-TC}(A) = 0.48 > \beta\text{-TC}(B) = 0.39,$$

favoring A, but taking into account the cost of tool swap.

d) Experimental Setup: [5] We used *Fanuc LRMate 200iD/7L* arm, with a tool selection hardware using two vacuum end-effectors: *Piab BL30-3P.4L.04AJ* (30mm) and *Piab BL50-2.10.05AD* (50mm). We used an assortment of mixed items (various sizes, weights, shapes, colors, etc., see Figure 1(b) for an example). Each end-effector is associated with a grasp proposal model trained using previously collected production data appropriate for that end-effector. Since it is not in the scope of this paper, we only provide a brief overview of our grasp proposal model architecture. Our grasp proposal models are inspired by the architecture proposed in [20] which consists of encoder-decoder convolutional neural nets consisting of a feature pyramid network [21] on a *ResNet-101* backbone and a pixelwise sigmoidal output of volume $W \times H$, where $W \times H$ are the dimensions of the grasp success probabilities $\Gamma_{\mathcal{E}}$. The network is then trained end-to-end using previously collected grasp success/failure data consisting of $5k$ grasp data per end-effector using stochastic gradient descent with momentum ($LR = 0.0003; p = 0.8$). We only used the STS solver in our evaluation, as our ILP implementation cannot run in real time.

Algorithm (w/30mm + 50mm)	TC	PA	PS	TC-Score (β=0.33)	PS/hr
Randomized	800	2191	744	0.3558	186
Naive Greedy	733	2093	1268	0.6099	317
Greedy	261	2702	1288	0.4999	295.41
MPC-STS (H=2, k=2)	229	2563	1719	**0.6885**	**429.75**

TABLE I: Performance comparison over different tool selection algorithms. PS/hr is the throughput.

e) Comparison with Baselines: Table I compares our method (MPC-STS) with 3 baselines. The first is a *randomized selector*, which randomly changes tools with probability $p = 0.75$ at each step, and forcing a change if not swapped after 10 steps. The second baseline is *naive*

[5] We encourage the readers to view our accompanying video for a demonstration of our experimental setup and results.

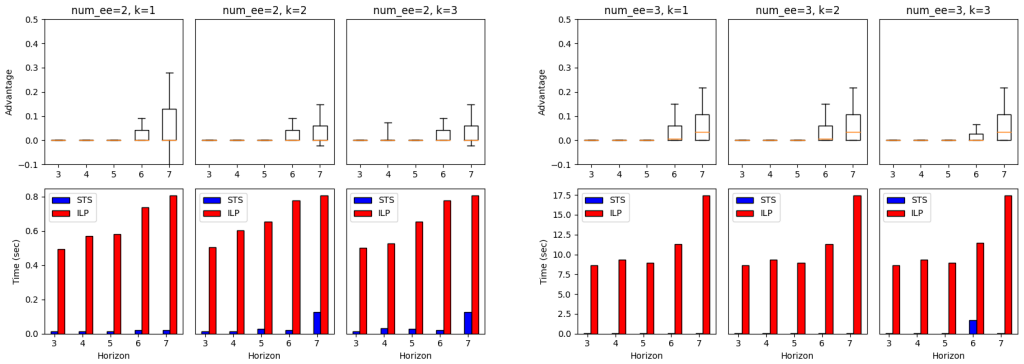

Figure 3: Results for synthetic experiments: (Left 3 columns) 2 end-effectors; (Right 3 columns) 3 end-effectors. While ILP is marginally better than STS in terms of advantage, STS yields superior speedup.

greedy selector, which chooses the next grasp proposal based on one-step reward function (see Equation 1). The third baseline is *greedy selector*, which accumulates the top $n = 5$ likelihood scores for each tool, and selects the tool with the highest sum. Our MPC-STS selector was configured with a void radius of $l = 100$mm (roughly 60 pixels), a maximum of 10 initial grasp proposal samples per end-effector, sparsity factor $k = 2$, and a max horizon of $H = 2$ (since it yielded the best results for MPC-STS in this domain based on the ablation results in Table IV). Observe that MPC-STS significantly outperforms the other baselines in terms of both TC-score and pick success rate per hour (improving over the best baseline by 50%).

f) Single end-effector Comparison: This set of comparisons is based on a separate set of shorter experimental runs with similar items; results are reported in Table II. Here, note the divergence between the TC-score and the throughput (PS/hr) in the ordering of the performance of the single 50mm end-effector and the naive greedy baseline. While the

Configuration	TC	PA	PS	TC-Score ($\beta = 0.33$)	PS/hr
Single (30mm)	0	745	359	0.508	287.2
Single (50mm)	0	864	572	0.685	490.3
Naive Greedy (30mm + 50mm)	217	636	465	0.751	348.8
MPC-STS (H=2, k=2) (30mm + 50mm)	71	691	524	**0.770**	**507.1**

TABLE II: Comparison of single end-effector performance vs multiple end-effectors and tool selection.

throughput for the single 50mm strategy is higher, the TC-score correctly reflects that this strategy is less pick efficient. Indeed, the successful pick percentage for the 50mm strategy is 66% while the successful pick percentage for the naive greedy strategy is 73%. The throughput in this case is inflated by executing failing picks quickly. As expected, MPC-STS outperforms all the baselines.

g) Parameter Study: In these experiments, reported in Tables III and IV, we investigate the dependence on the void radius and max horizon. On our item set, increasing the size of the void radius leads to a decrease in tool-changing efficiency and overall throughput at an MPC-STS with $H = 3$. As the tree search progresses, the bin becomes increasingly voided. For large void radii, a large fraction of the bin will be voided, leading to unreliable reward estimates.

MPC-STS (H=3, k=2)	TC	PA	PS	TC-Score ($\beta = 0.33$)	PS/hr
l = 50mm	72	720	586	**0.822**	**540.9**
l = 100mm	58	649	431	0.682	417.1
l = 150mm	98	619	409	0.675	348.8

TABLE III: Investigation of void radius l (in mm).

Thus, as long as the void radius is large enough to cover areas disturbed by previous picks, the smaller radius the better. We also see that increasing the max horizon H from 1 to 2 leads to an increase in performance, but thereafter there is a decrease in performance metrics even though the overall tool change count remains similar. We conjecture that this is due the crude approximation of the deterministic dynamics, which are not reliable for a long planning horizon.

MPC-STS ($k = 2$, l=100mm)	TC	PA	PS	TC-Score ($\beta = 0.33$)	PS/hr
H = 1	64	712	522	0.747	481.8
H = 2	60	653	511	**0.793**	**502.6**
H = 3	58	649	431	0.682	417.1
H = 5	65	646	365	0.586	353.2

TABLE IV: Investigation of planning horizon.

V. Conclusions and Future Directions

In this work we introduced the Grasp Tool Selection Problem (GTSP), and presented several approximate solutions that can be deployed in real time on realistic robotic setups. Our experiments demonstrated that significant gains can be reaped by carefully planning the tool selection. For industrial bin picking, where every performance gain directly translates to revenue, we believe that our method could be valuable.

Deep learning based prediction models are becoming increasingly popular in robotics. Our work explored an optimization-based approach for maximizing the utilization of the learned models. In general, we believe that optimally choosing between several learned models could be relevant for other robotic tasks, for example, choosing between different gaits in robotic locomotion. The ideas in this work may inspire algorithms for more general problems.

REFERENCES

[1] A. Zeng, S. Song, K.-T. Yu, E. Donlon, F. R. Hogan, M. Bauza, D. Ma, O. Taylor, M. Liu, E. Romo, *et al.*, "Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 3750–3757.

[2] I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps," *The International Journal of Robotics Research*, vol. 34, no. 4-5, pp. 705–724, 2015.

[3] S. Levine, P. Pastor, A. Krizhevsky, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," 2016.

[4] J. Redmon and A. Angelova, "Real-time grasp detection using convolutional neural networks," 2015.

[5] J. Mahler, M. Matl, V. Satish, M. Danielczuk, B. DeRose, S. McKinley, and K. Goldberg, "Learning ambidextrous robot grasping policies," *Science Robotics*, vol. 4, no. 26, 2019. [Online]. Available: https://robotics.sciencemag.org/content/4/26/eaau4984

[6] S. J. Russell and P. Norvig, "Artificial intelligence: A modern approach," 1995.

[7] E. F. Camacho and C. B. Alba, "Model predictive control," in *Springer Science & Business Media*, 2013.

[8] A. Tamar, G. Thomas, T. Zhang, S. Levine, and P. Abbeel, "Learning from the hindsight plan - episodic MPC improvement," *CoRR*, vol. abs/1609.09001, 2016. [Online]. Available: http://arxiv.org/abs/1609.09001

[9] C. Finn and S. Levine, "Deep visual foresight for planning robot motion," *CoRR*, vol. abs/1610.00696, 2016. [Online]. Available: http://arxiv.org/abs/1610.00696

[10] F. Ebert, C. Finn, A. X. Lee, and S. Levine, "Self-supervised visual planning with temporal skip connections," 2017.

[11] F. Ebert, C. Finn, S. Dasari, A. Xie, A. Lee, and S. Levine, "Visual foresight: Model-based deep reinforcement learning for vision-based robotic control," 2018.

[12] A. Xie, F. Ebert, S. Levine, and C. Finn, "Improvisation through physical understanding: Using novel objects as tools with visual foresight," 2019.

[13] D. Bertsekas, *Dynamic programming and optimal control: Volume I*. Athena scientific, 2012, vol. 1.

[14] G. B. Dantzig, D. R. Fulkerson, and S. M. Johnson, *Solution of a Large-Scale Traveling-Salesman Problem*. Santa Monica, CA: RAND Corporation, 1954.

[15] C. Miller, A. Tucker, and R. Zemlin, "Integer programming formulations and traveling salesman problems," *Journal of Association for Computing Machinery*, vol. 7, p. 326–329, 1960.

[16] K. Rohanimanesh, J. Metzger, W. Richards, and A. Tamar, "Online tool selection with learned grasp prediction models," 2023. [Online]. Available: https://arxiv.org/abs/2302.07940

[17] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2022. [Online]. Available: https://www.gurobi.com

[18] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 5026–5033.

[19] R. Baeza-Yates, B. Ribeiro-Neto, *et al.*, *Modern information retrieval*. ACM press New York, 1999, vol. 463.

[20] B. Goodrich, A. Kuefler, and W. D. Richards, "Depth by poking: Learning to estimate depth from self-supervised grasping," *CoRR*, vol. abs/2006.08903, 2020. [Online]. Available: https://arxiv.org/abs/2006.08903

[21] T. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 936–944.