Optimal Knock-Pick Planning for Tightly Packed Tabletop Blocks With Parallel Grippers

Hao Lu and Rahul Shome

School of Computing
Australian National University

Abstract. Rearranging densely packed tabletop objects is challenging when parallel-gripper picks are infeasible without sufficient clearance around an object. This work studies the problem characteristics for practically motivated settings with uniformly sized blocks placed at planar tabletop grid locations. Since purely prehensile removal can become infeasible, a directional knock primitive is therefore introduced and the optimal knock-pick variant of the problem is formulated. The work proposes a series of abstractions wherein minimal constraining gadgets are covered to identify the necessary knocks. Utilizing maximum-weight perfect matching on a graphical abstraction yields efficient polynomial time computation of the optimal plan that minimizes the number of actions. Experiments are reported for increasing grid sizes in synthetic settings as well as in IsaacSim. The theoretical observations provide a promising stepping stone towards rigorously building efficient manipulation strategies that interleave prehensile and non-prehensile actions.

1 Introduction

Tightly packed objects are common in logistics and automation settings [19, 27]. While object rearrangement [9] is a well-studied problem with wide applications in automation, tightly packed settings like the sorting task in Fig 1 expose a critical gap — using parallel gripper picks is often infeasible when objects have no clearance. A combination of prehensile picks and some non-prehensile action is imperative. This work introduces a non-prehensile *knock* action that displaces an object in a desired direction. It remains to be seen whether *optimal knock–pick plans can be computed efficiently in tightly packed settings?*

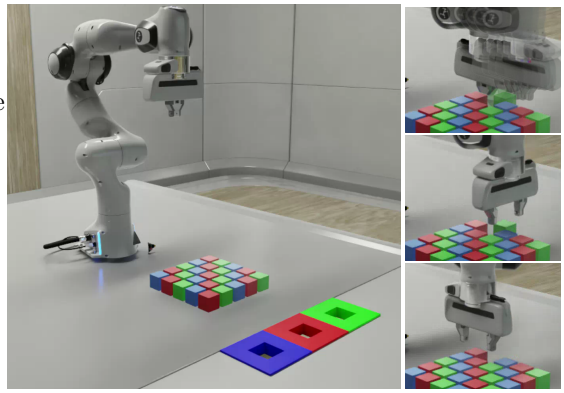

Fig. 1. A color sorting task with parallel grippers. *(Right)* A *knock* makes the subsequent *picks* valid.

Towards the broader aim, the current work focuses on a more specific setting. This work inspects the category of problems arising from using a parallel gripper to pick a set of tabletop cubical blocks that are tightly packed in a grid (Fig 1). Having a suction-based end-effector permits immediate feasibility with top-down picks and allows the problem to be solved and analyzed using existing techniques [9]. In contrast, finger-based end-effectors, like parallel grippers, typically need clearance around objects to allow them to be picked. Densely packed arrangements often exhibit local geometric locking: surrounding objects prevent access to grasp points, and no object is immediately graspable with finger-based end-effectors. Such dense scenes can arise in warehouse bin clearing, tabletop packing/unpacking, and similar "blocks world" tasks, where objects lack the necessary clearance to admit pick feasibility with parallel grippers. Note that the occurrence of even a single square-like pattern of four blocks introduces infeasibility with parallel gripper picks. It is of interest to solve and characterize the theoretical properties of sequential manipulation problems in such tightly packed scenarios.

In tightly packed tabletop settings, *non-prehensile* actions—pushing, toppling, or knocking—that create space by moving an object without a stable grasp. While necessary in the current setting, they come with tangible costs: they are less reliable than grasping, more likely to cause collateral motion, and may require additional recovery steps for execution feasibility. In many systems, a non-prehensile action is best viewed as an *expensive clearance-creation operation*, after which standard pick–place can proceed. Designing robust non-prehensile primitives is, by itself, a challenging problem and an active area of research [28, 2]. In the current work, the hand-crafted knock primitive is tested on IsaacSim and shows promising performance, with room for motivating improvements to robust execution by bridging the Sim-to-Real gap [11]. Despite this, the empirical results motivate the utility of optimal knock–pick plans for tightly packed blocks and thereby preserve the primary focus of the current work in characterizing optimal knock–pick planning complexity and solving strategies.

The focus of the current work on grid arrangements provides an opportunity to study the regime under simplified abstractions defined over the tabletop grid. Interestingly, the core result uncovers that *computing the high-level optimal knock-pick plan can be done in* ***polynomial time***. This follows from transforming the minimal knock action computation to a maximum weight perfect matching problem that yields a way to break every local pick constraint. Minimal constraining gadgets are identified and ensuring all objects are covered by such gadgets achieves our aim. Under, the reachable tabletop setting, the top-down primitives will be motion feasible. The current theoretical result for the optimal plan is essentially the best performance achievable given the available actions. Practical execution strategies of non-prehensile actions may introduce the need to retry or repeat actions or more advanced online adaptation and recovery strategies, which is beyond the current scope of the paper. The reported physics simulator results include rudimentary retry and resume strategies and provide promising motivating performance.

This paper makes three contributions: (i) a formal graph-based abstraction of tightly packed block manipulation with a parallel gripper using top-down picks and non-prehensile knock actions; (ii) framing the problem of minimum-knock plan computation to a special case of optimal exact cover problem that can be solved in polynomial time using maximum-weight perfect matching; and (iii) an execution routine that converts the combinatorial optimum into a knock–pick (KP)-feasible sequence that is evaluated in synthetic graphs and physics simulation in IsaacSim.

2 Related Work

Object Rearrangement. A large body of work addresses object rearrangement [17], with strong results for optimality and complexity [9] in scenarios involving different number and constraints exhibited for prehensile object picks and placements. Notably, tightly packed block settings can be infeasible solely using prehensile parallel gripper grasps so novel solvers have to be devised beyond the scope of classical object rearrangement, including rigorous exploration of underlying combinatorial structures that can lead to effective solvers.

Cluttered Manipulation Cluttered settings [18] have also been optimized [25] in task and motion planning frameworks, particularly with parallel grippers. However, the focus here is problems where low but existing clearance and objects having different heights permitted feasibility with parallel grasp picks. Other work [10] has focused on geometric considerations of the problem. Task and motion planning [3] paradigms provide generalizability and are complementary to effective combinatorial problem abstractions.

Non-prehensile Manipulation Non-prehensile object interaction [21, 12, 20] has attracted sustained interest since the seminal advances in push-grasping [1, 4]. Among early works, [13] analyzes the mechanics of planar sliders under *tapping*, while [15] formalizes *toppling* as a manipulation primitive and derives the associated mechanical conditions and planning considerations. Categories of manipulation problems require a combination of prehensile and non-prehensile actions. The current problem lies in such a category. Certain topological results have led to enhanced insights into the nature of push-grasp problems [26]. The current work foregoes generalization for aiming to understand theoretical optimality bounds for such problem categories. Towards this end, the current study focuses on a less general, yet practical setting of tight block arrangements. Here, non-prehensile actions are necessary and tight theoretical bounds of polynomial solvability and optimality of task planning can be devised.

Packing Assemblies Beyond clutter, tightly packed objects [27] are recognized to be a particularly challenging category of object manipulation problems where the clearance is low to non-existent. While pipelines [19] designed for packing have been designed to compose such tightly packed arrangements, taking apart such

an arrangement forms the focus of the current work. Interestingly, packing tasks have been often addressed with suction-based end-effectors as they typically do not depend on clearance around the object like parallel grippers. The current work attempts to look into this somewhat understudied application of parallel grippers to packing tasks. There is a rich history of related planning problems with computational geometric motion space modeling of assembly planning [8] towards designing execution pipelines [22, 24]. The current work is a specific investigation of theoretical optimality for tightly packed uniform geometry blocks that need both prehensile and non-prehensile actions with generalization to other geometries and integrations of robust primitive design and execution pipelines being a motivating future direction.

Learning Primitives Non-prehensile grasps can exacerbate the sim-to-real gap [11] as complex object interactions introduces considerations of contact, friction, physics, etc. Learning from demonstration [28] can play a key role in robust non-prehensile primitives. Recent advances in imitation learning methods based on using generative models [2] have shown promise. Effective primitive design, though not the focus of the current paper, is essential for practical robust deployment. The parameterized knock action has been hand-crafted and advanced learned controllers can be foreseeably plugged into robust online frameworks.

3 Problem Formulation

We consider a robot rearrangement manipulator motivated by tightly packed tabletop scenes. A set of identical cubical objects is placed on a planar tabletop and arranged on a discrete axis-aligned grid. A parallel-jaw gripper can grasp an object only when sufficient lateral clearance exists along at least one grasping axis. Otherwise, non-prehensile actions such as *knocking* can create clearance.

The problem is set up with a manipulator with parallel gripper end effectors and a set of k objects o at an initial arrangement $\mathcal{P}_{\text{init}} = (p_i, p_i \in SE(3))$ such that p_i is an initial pose of o_i. The manipulator can interact the objects using a set of manipulation actions $\mathcal{A} = \{a_j\}$ that correspond to a motion of the manipulator and alter the pose of the objects. A goal arrangement is defined in terms of goal sets per object, $\mathcal{P}_{\text{final}} = (P_i, P_i \subset SE(3))$, a feasible solution to the manipulation problem is a plan $\mathit{\Pi} = (a_1, a_2, \cdots a_\ell)$ that moves the objects from the initial arrangement to a desired goal arrangement.

Assumption 1 (Initial tight grid arrangement of blocks) *The objects are assumed to be all cubical blocks of identical block geometry. Each pose in $\mathcal{P}_{\text{init}}$ lies on a tabletop grid location where the object (in isolation) can be top-down grasped using parallel grippers. The dimension of the grid is equal to the dimension of a block, i.e., there is no clearance if adjacent locations are occupied.*

The no-clearance scenario is a key motivation for devising strategies with parallel grippers, which need clearance for grasp feasibility.

Assumption 2 (Reachable unconstrained target arrangement) *The feasibility of placing an object at a target location does not depend on the initial block arrangement, i.e., there will exist feasible reachable candidates from* $\mathcal{P}_{\text{final}}$.

The target arrangement assumption is easily motivated by categories of problems where the target locations are reachable but non-overlapping with $\mathcal{P}_{\text{init}}$.

Assumption 3 (Local non-prehensile primitive) *A non-prehensile primitive (knock) is assumed to be available as an action in* $\mathcal{A}$ *that can act locally on a block, alter the pose of the block to make it feasible to pick, and only affect the pick feasibility of the current block and a local neighborhood.*

A non-prehensile action is needed in $\mathcal{A}$ but its locality restriction allows a systematic inspection of the underlying combinatorial structure of the problem. Intuitively, the assumption distinguishes knock actions that move a block deliberately to an adjacent region from alternative strategies that might want to move larger groups of objects simultaneously, with the current work focusing on the former. The remaining formulation of the problem can proceed in the context of these assumptions. Set the tabletop grid arrangement by fixing integers $m, n \geq 1$ and define the (axis-aligned) *workspace grid* on the tabletop

$$\mathcal{H} \triangleq \{(i,j) \in \mathbb{Z}^2 : \ 0 \leq i \leq m-1, \ 0 \leq j \leq n-1\}.$$

Definition 1 (Object Location Graph).

An grid location graph *is defined in terms of a subset* $\mathcal{B} \subseteq \mathcal{H}$*, where each* $v \in \mathcal{B}$ *represents one cubical object placed at grid location* v*. Given* $\mathcal{B} \subseteq \mathcal{H}$*, define an undirected graph*

$$G(\mathcal{B}) \triangleq (\mathcal{V}, E), \qquad \mathcal{V} = \mathcal{B}, \qquad \{u,v\} \in E \iff \|u - v\|_1 = 1.$$

3.1 Manipulation primitives

It is necessary to model two manipulation primitives: *pick* (prehensile grasp with a parallel gripper as shown in Fig 2) and *knock* (non-prehensile removal along a grid-aligned direction as shown in Fig 3).

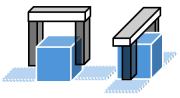

Fig. 2. Parallel gripper Pick.

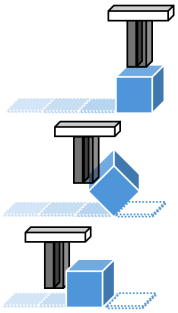

Fig. 3. Knock action (to the left).

Pick feasibility. An object at a grid location $v \in \mathcal{V}$ is *pickable* if two adjacent locations are unoccupied in the same row or the same column. This leads to two conditions for feasibility in terms of vertex degree: (P1) $\deg_G(v) \leq 1$, or (P2) $\deg_G(v) = 2$ and its two neighbors are collinear. We denote this predicate by $\mathsf{Pick}(G, v)$.

Knock feasibility. Let $\mathcal{D} \triangleq \{(\pm 1, 0), (0, \pm 1)\}$. For $v \in \mathcal{V}$ and $d \in \mathcal{D}$, define the *workspace ray* $\mathsf{Ray}(v,d) \triangleq \{v + td \in \mathcal{H} : \ t \in \mathbb{Z}, \ t \geq 1\}$. A knock at v in direction d is valid if

$$\mathsf{Ray}(v,d) \cap \mathcal{V} = \emptyset. \tag{1}$$

We say v is *knockable* if $\exists d \in \mathcal{D}$ satisfying (1), and denote this by $\mathsf{Knock}(G, v)$.

A *knock–pick* KP-plan for $\mathcal{B}$ is a finite sequence of actions which can be $a_t \in \{\mathsf{Pick}, \mathsf{Knock}\}$ for an object at vertex v_t for each discrete step t.

The action at time t is feasible if the corresponding predicate $\mathsf{Pick}(G_t, v_t)$ or $\mathsf{Knock}(G_t, v_t)$ holds for all the actions sequentially across time steps. The execution is *complete* if it removes all vertices from $G(\mathcal{B})$.

We measure cost by the number of actions in the plan. This is equivalent to minimizing the number of knocks, since every object must be removed exactly once and picks are unavoidable whenever feasible.

Definition 2 (Optimal KP Planning). *Given an initial tabletop tightly packed grid arrangement of blocks represented by $\mathcal{B}$, target goal arrangement $\mathcal{P}_{\text{final}}$, available actions $\mathcal{A} = \{\mathsf{Knock}, \mathsf{Pick}\}$, an optimal KP plan $\Pi = (a_i)$ is a sequence of feasible actions that enact the rearrangement for all blocks such that the number of actions in Π is minimized.*

Since each object is eventually removed exactly once, minimizing the total number of actions is equivalent to minimizing the number of knocks, up to the additive constant $|\mathcal{V}|$.

4 Methodology

In this section the graphical abstractions to represent the problem will be proposed and solution strategies built on top of the combinatorial structures these abstractions will express. Our approach proceeds in four conceptual steps: (i) remove all immediately pickable objects by exhaustive picking (*cleanup*), (ii) characterize the remaining non-pickable structure via three minimal gadgets and a face system, (iii) compute an *optimal exact face cover* by reducing to a maximum-weight perfect matching problem, and (iv) generate a *KP-feasible* execution by iterating ray-feasible knocks and cleanup picks.

Definition 3 (Cleaned Graph). *Given a graph $G = (\mathcal{V}, E)$, define $\mathsf{Clean}(G)$ as the graph obtained by repeatedly deleting any pickable vertex until none remains. We call G* cleaned *if $G = \mathsf{Clean}(G)$.*

In what follows, we write $G_0 \triangleq G(\mathcal{B})$ for the initial grid graph and $G^\circ \triangleq \mathsf{Clean}(G_0)$ for its cleaned graph.

4.1 Faces and Gadgets

We define three vertex-induced gadgets that represent minimal non-pickable patterns (illustrated in Fig 4). Each gadget corresponds to either one or two *unit faces*. A *unit face* (or simply *face*) is a 2×2 fully occupied square.

Definition 4 (Faces). *Given $G(\mathcal{B})$, define the set of* unit faces

$$\mathsf{Faces}(G) \triangleq \Big\{(i,j) \in \mathbb{Z}^2 : \{(i,j),(i+1,j),(i,j+1),(i+1,j+1)\} \subseteq \mathcal{V}(G)\Big\}.$$

Each $(i,j) \in \mathsf{Faces}(G)$ identifies the corresponding 2×2 vertex set. We denote this face by $\mathsf{face}(i,j)$.

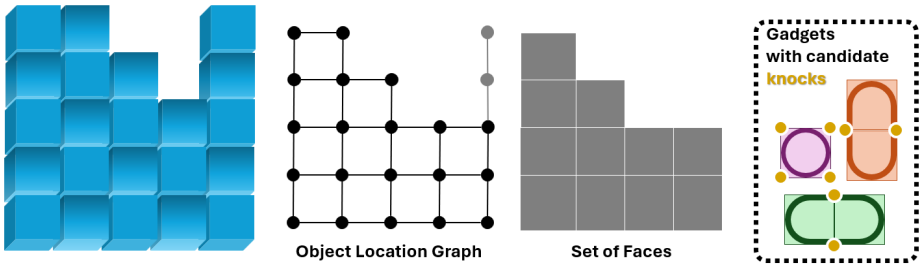

Fig. 4. *(Left)* Tabletop grid arrangement of blocks. *(Second)* Object location graph, which greyed out ones indicating blocks that can be immediately picked (within Clean). *(Third)* A face is a minimal constraining set of four objects arranged in a 2×2 square in the object location graph. The object location graph corresponds to a set of faces. *(Right)* Gadgets are defined in terms of faces alongside knock candidate objects, knocking one of which can help pick the remaining objects in the gadget.

Definition 5 (Gadgets). *Let* $(i, j) \in \mathbb{Z}^2$.
Square gadget $(\mathbf{2 \times 2})$. $Q(i,j) \triangleq \{(i,j), (i+1,j), (i,j+1), (i+1,j+1)\}$. *This corresponds to exactly one face* $\mathsf{face}(i,j)$.
Vertical gadget $(\mathbf{3 \times 2})$. $R_v(i,j) \triangleq \{(i+r, j+c) : r \in \{0,1,2\},\, c \in \{0,1\}\}$. *This contains exactly two faces:* $\mathsf{face}(i,j)$ *and* $\mathsf{face}(i+1,j)$.
Horizontal gadget $(\mathbf{2 \times 3})$. $R_h(i,j) \triangleq \{(i+r, j+c) : r \in \{0,1\},\, c \in \{0,1,2\}\}$. *This contains exactly two faces:* $\mathsf{face}(i,j)$ *and* $\mathsf{face}(i,j+1)$.

To destroy *two* faces with *one* knock in a 2×3 or 3×2 gadget, the knocked vertex must lie on the shared boundary. This yields the following candidate sets.

Definition 6 (Knock-candidate vertices of a gadget). *Let* S *be a gadget occurrence present in the current graph, and let* (i_0, j_0) *be its anchor (the lexicographically minimum coordinate in* S*). Define* $\mathsf{Cand}(S) \subseteq S$ *by gadget type:*

(i) If $S = Q(i_0, j_0)$ *is a* 2×2 *gadget, then* $\mathsf{Cand}(S) = S$.
(ii) If $S = R_h(i_0, j_0)$ *is a* 2×3 *gadget, then*

$$\mathsf{Cand}(S) = \{(i_0, j_0+1), (i_0+1, j_0+1)\},$$

i.e., the two vertices in the middle column *(shared boundary of the faces).*
(iii) If $S = R_v(i_0, j_0)$ *is a* 3×2 *gadget, then*

$$\mathsf{Cand}(S) = \{(i_0+1, j_0), (i_0+1, j_0+1)\},$$

i.e., the two vertices in the middle row *(shared boundary of the faces).*

Lemma 1 (Gadgets are non-pickable). *Let* $X \in \{Q(i,j), R_v(i,j), R_h(i,j)\}$ *be fully occupied and let* G_X *be the induced subgraph on* X*. Then* G_X *has no pickable vertex:* $\mathsf{Pick}(G_X, v)$ *is false for all* $v \in X$.

Proof. In a 2×2 square, each vertex has degree 2 and its two neighbors are non-collinear, so (P2) fails. In a 3×2 (resp. 2×3) rectangle, corners again have degree 2 with non-collinear neighbors, while the two interior boundary vertices have degree 3. Thus no vertex satisfies (P1) or (P2). □

Lemma 2 (One knock suffices for each gadget). *Let X be fully occupied, where $X \in \{Q(i,j), R_v(i,j), R_h(i,j)\}$. There exists a single vertex deletion (a knock) after which exhaustive picking removes all remaining vertices in X.*

Proof. For $Q(i,j)$, knocking any one vertex leaves a 3-vertex tree, which is removable by leaf picking. For $R_v(i,j)$ and $R_h(i,j)$, knocking an appropriate "hinge" vertex breaks both faces at once, leaving an acyclic structure with leaves. Exhaustive picking then removes all remaining vertices. □

4.2 Exact Covers with Gadgets

Let $G^\circ \triangleq \mathsf{Clean}(G_0)$ denote the cleaned graph. Define the universe as the faces in G°: $\mathcal{U} \triangleq \mathsf{Faces}(G^\circ)$. For each gadget in G°, define a set of faces it covers:

- If $Q(i,j) \subseteq V(G^\circ)$ then include $S = \{\mathsf{face}(i,j)\}$.
- If $R_v(i,j) \subseteq V(G^\circ)$ then include $S = \{\mathsf{face}(i,j), \mathsf{face}(i+1,j)\}$.
- If $R_h(i,j) \subseteq V(G^\circ)$ then include $S = \{\mathsf{face}(i,j), \mathsf{face}(i,j+1)\}$.

Let $\mathcal{S}$ be the family of all such sets. Each $S \in \mathcal{S}$ has size 1 or 2.

The current problem then needs to describe a set of such gadgets that covers all faces. Minimizing the cardinality of this cover correspondingly minimizes the number of actions in the original problem.

Definition 7 (Optimal exact face cover). *An* exact cover *is a subfamily $\mathcal{C} \subseteq \mathcal{S}$ such that (i) $\bigcup_{S \in \mathcal{C}} S = \mathcal{U}$ and (ii) the sets in $\mathcal{C}$ are pairwise disjoint: $S \cap S' = \emptyset$ for all distinct $S, S' \in \mathcal{C}$. We seek an optimal exact cover*

$$\mathcal{C}^\star \in \arg\min_{\mathcal{C} \subseteq \mathcal{S}} |\mathcal{C}| \quad \textit{s.t. } \mathcal{C} \textit{ is an exact cover of } \mathcal{U}.$$

Existence. An exact cover always exists for system $(\mathcal{U}, \mathcal{S})$: for every face $f \in \mathcal{U}$, the square gadget supplies a singleton set $\{f\} \in \mathcal{S}$, so selecting all singletons yields an exact cover.

Lemma 3 (Any face requires a knock). *Let G be cleaned (i.e., $G = \mathsf{Clean}(G)$) and let $\mathsf{face}(i,j) \in \mathsf{Faces}(G)$. Then in any KP execution, the first removal of a vertex from $Q(i,j)$ must be a knock.*

Proof. By Lemma 1, no vertex of $Q(i,j)$ is pickable while the face is intact. Since G is cleaned, no vertex is pickable at this stage, so the first removal from the face cannot be a pick and must be a knock. □

Lemma 4 (A knock destroys at most two faces). *In a grid-induced graph, deleting one vertex destroys at most two faces of $\mathsf{Faces}(\cdot)$. Moreover, it destroys two faces iff the deleted vertex belongs to two adjacent faces, i.e., it is a shared corner of a fully occupied 2×3 or 3×2 gadget.*

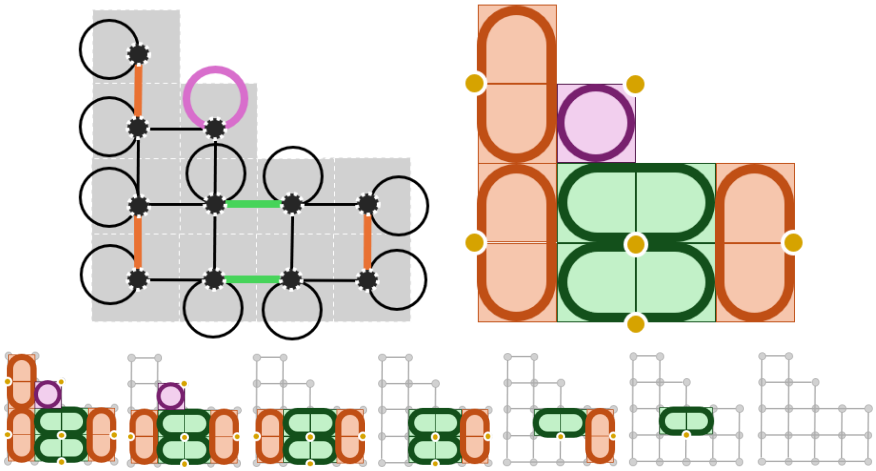

Fig. 5. (Top) Face-graph abstraction (left) and the resulting optimal gadget cover (right). *(Top-left)* Each vertex represents a unit face in the cleaned instance. A self-loop at a vertex represents a square (2×2) gadget covering that single face; an edge between two vertices represents a horizontal (2×3) or vertical (3×2) gadget covering the corresponding adjacent face pair. Highlighted edges (including loops) show the maximum-weight perfect matching solution, which induces an optimal exact face cover. *(Top-right)* The corresponding optimal gadget cover on the object graph (each vertex represents an object); highlighted object vertices indicate knock-candidate locations with at least one feasible knock direction in a KP-feasible execution. **(Bottom)** An illustration of a KP feasible plan corresponding to a sequence of gadgets knocked and cleaned till all objects are handled.

Proof. A face is a 2×2 set of vertices, and a vertex participates in a face only as a corner. If a vertex is knockable under (1), it has at least one missing incident neighbor, hence degree at most 3. In the grid, a degree-$\leq$ 3 vertex can be a corner of at most two fully occupied 2×2 faces (those using the two perpendicular present edges adjacent to the missing direction). Thus deleting one vertex destroys at most two faces. Two faces are destroyed exactly when the vertex is shared by two faces, which occurs precisely in the fully occupied 2×3 or 3×2 pattern. □

4.3 Optimal Exact Cover via Maximum Weight Perfect Matching

Set cover itself is an NP-hard problem. However, the set system $(\mathcal{U}, \mathcal{S})$ is a special case of set cover in which every set has size 1 or 2 and every element admits a singleton set. This special structure lets us solve the exact set cover *optimally* in polynomial time via maximum-weight perfect matching (see Fig 5).

Definition 8 (Face-adjacency graph with loops). *Define the* face-adjacency graph $\mathcal{G}_F$ *as follows. Its vertex set is* $\mathcal{U} = \mathsf{Faces}(G^\circ)$. *For two distinct faces* $f, g \in \mathcal{U}$, *add an undirected edge* $\{f, g\}$ *iff* f *and* g *are adjacent faces (i.e., they share a grid edge) and the corresponding* 2×3 *or* 3×2 *gadget is present in* G°.

Additionally, add a self-loop (f, f) at every $f \in \mathcal{U}$ corresponds to 2×2 gadgets. Assign weights

$$w(f, f) = 0 \quad \textit{and} \quad w(f, g) = 1 \ \textit{for all distinct adjacent } f, g.$$

Perfect matchings with loops. A *perfect matching* of $\mathcal{G}_F$ is a set of edges $\mathcal{M} \subseteq E(\mathcal{G}_F)$ such that every vertex $f \in \mathcal{U}$ is incident to *exactly one* edge in $\mathcal{M}$, where a loop (f, f) counts as incident to f.

Lemma 5 (Exact covers $\leftrightarrow$ perfect matching). *There is a bijection between exact covers $\mathcal{C} \subseteq \mathcal{S}$ of $\mathcal{U}$ and perfect matching $\mathcal{M}$ of $\mathcal{G}_F$. Under this bijection, each chosen singleton set $\{f\}$ corresponds to the loop (f, f), and each chosen 2-face set $\{f, g\}$ corresponds to the edge $\{f, g\}$.*

Proof. Given an exact cover $\mathcal{C}$, every $f \in \mathcal{U}$ belongs to exactly one chosen set. Map that set to the corresponding loop (if singleton) or adjacency edge (if a pair). The disjointness of $\mathcal{C}$ implies no two mapped edges share a vertex, and coverage implies each vertex is incident to exactly one mapped edge. Hence we obtain a perfect matching. Conversely, given a perfect matching, map each loop/edge to its singleton/pair set. The matching property yields the exactness of the cover and vertex coverage yields $\bigcup \mathcal{C} = \mathcal{U}$. □

Lemma 6 (Optimal exact cover via maximum weighted perfect matching). *The minimum exact cover $\mathcal{C}^* \subseteq \mathcal{S}$ of $\mathcal{U}$ corresponds to a maximum weighted perfect matching $\mathcal{M}^*$ of $\mathcal{G}_F$.*

Proof. For any perfect matching $\mathcal{M}$, let $p(\mathcal{M})$ be the number of non-loop edges in $\mathcal{M}$. Because $\mathcal{M}$ covers all vertices exactly once, the number of loops is $|\mathcal{U}| - 2p(\mathcal{M})$, hence its weight is

$$w(\mathcal{M}) = 1 \cdot p(\mathcal{M}) + 0 \cdot (|\mathcal{U}| - 2p(\mathcal{M})) = p(\mathcal{M}).$$

Therefore maximizing $w(\mathcal{M})$ is equivalent to maximizing $p(\mathcal{M})$, i.e., using as many 2-face gadgets as possible. Since any exact cover $\mathcal{C}$ satisfies $|\mathcal{C}| = p(\mathcal{M}) + (|\mathcal{U}| - 2p(\mathcal{M})) = |\mathcal{U}| - p(\mathcal{M})$ under the bijection, a maximum-weight perfect matching $\mathcal{M}^*$ yields a minimum-cardinality exact cover $\mathcal{C}^\star$. □

Polynomial-time solvability: Maximum-weight matching in general graphs is solvable in polynomial time by Edmonds' matching framework [5]. Here, $\mathcal{G}_F$ always admits a perfect matching because every vertex has a loop available.

4.4 Optimal knock minimization and KP-feasibility

We now connect the combinatorial optimum to the KP objective (minimum number of knocks) and show that the resulting optimal gadget set is always executable under the ray constraint.

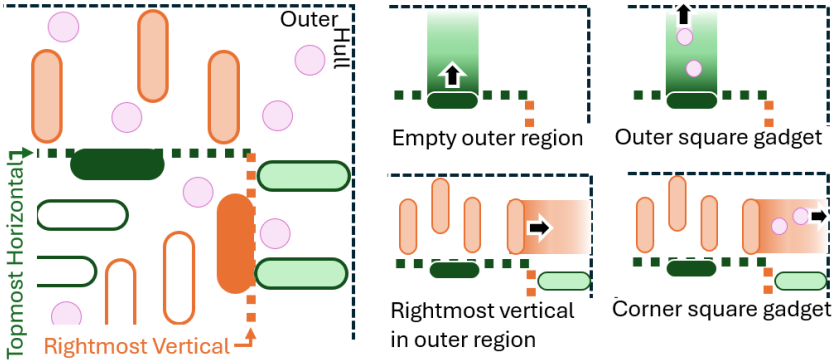

Fig. 6. Illustration of Lemma 7 for the case $\sigma = (-1, 1)$, corresponding to the top-right corner. *Left:* the horizontal and vertical σ-outer boundaries are determined by the extremal horizontal gadget S_h^σ (dark green) and extremal vertical gadget S_v^σ (dark orange). Horizontal gadgets are shown in green, vertical gadgets in orange, and square gadgets in pink. *Right:* representative configurations: an unobstructed boundary gadget (top-left); a boundary gadget whose first blocker and terminal gadget are both square (top-right); a boundary gadget whose first blocker and terminal gadget are both vertical (bottom-left); and a boundary gadget whose first blocker is vertical while its terminal gadget is square (bottom-right). In each subfigure, the arrow indicates a KP-feasible knock, either directly on the boundary gadget or on the terminal gadget reached by following first blockers in the outward boundary directions.

Definition 9 (σ-outer boundaries, outward directions, and boundary distance). *Let G be a cleaned graph with nonempty face set $U = \mathsf{Faces}(G)$, and let $\mathcal{C}$ be an exact cover of U by gadgets in $\mathcal{S}$. Fix $\sigma = (\sigma_1, \sigma_2) \in \{\pm 1, \pm 1\}$, and let c_σ denote the corresponding hull corner. If horizontal gadgets exist in $\mathcal{C}$, choose $S_h^\sigma = R_h(i_h, j_h) \in \arg\max\{(\sigma_1 i, \sigma_2 j) : R_h(i,j) \in \mathcal{C}\}$, with lexicographic maximization. If vertical gadgets exist in $\mathcal{C}$, choose $S_v^\sigma = R_v(i_v, j_v) \in \arg\max\{(\sigma_2 j, \sigma_1 i) : R_v(i,j) \in \mathcal{C}\}$, again with lexicographic maximization. Define the outward boundary directions $d_h^\sigma \triangleq (\sigma_1, 0), d_v^\sigma \triangleq (0, \sigma_2)$. These are the directions toward the hull corner c_σ for horizontal and vertical gadgets, respectively. If $S_h^\sigma = R_h(i_h, j_h)$ exists, define its* horizontal σ-outer boundary coordinate *by $b_h^\sigma \triangleq i_h + \frac{1+\sigma_1}{2}$, and if no horizontal gadget exists, let b_h^σ be the coordinate of the hull edge incident to c_σ in direction d_h^σ. If $S_v^\sigma = R_v(i_v, j_v)$ exists, define its* vertical σ-outer boundary coordinate *by $b_v^\sigma \triangleq j_v + \frac{1+\sigma_2}{2}$, and if no vertical gadget exists, let b_v^σ be the coordinate of the hull edge incident to c_σ in direction d_v^σ. The* horizontal σ-outer boundary *and* vertical σ-outer boundary *are $B_h^\sigma \triangleq \{(i,j) \in \mathcal{H} : i = b_h^\sigma\}, B_v^\sigma \triangleq \{(i,j) \in \mathcal{H} : j = b_v^\sigma\}$. For a gadget S, let $x_\sigma(S)$ denote its candidate vertex facing c_σ, namely $x_\sigma(Q(i,j)) = \left(i + \frac{1+\sigma_1}{2}, j + \frac{1+\sigma_2}{2}\right)$, $x_\sigma(R_h(i,j)) = \left(i + \frac{1+\sigma_1}{2}, j + 1\right)$, $x_\sigma(R_v(i,j)) = \left(i + 1, j + \frac{1+\sigma_2}{2}\right)$. Define $\delta_\sigma(S)$ as the σ-distance of a gadget S, which is the sum of the distances from the σ-facing candidate of S to the two hull edges incident to c_σ. A* first blocker *of a gadget in an outward direction $d \in \{d_h^\sigma, d_v^\sigma\}$ means any gadget whose vertices intersect the corresponding outward ray and whose first point of intersection occurs at minimum positive distance from the candidate vertex along that ray.*

Lemma 7 (Existence of a KP-feasible gadget). *Let G be a cleaned graph with nonempty face set $U = \mathsf{Faces}(G)$, and let $\mathcal{C}$ be an exact cover of U by gadgets in $\mathcal{S}$. Fix $\sigma \in \{\pm 1, \pm 1\}$, and let S_h^σ, S_v^σ, B_h^σ, B_v^σ, d_h^σ, d_v^σ, x_σ, δ_σ be as in Definition 9. Then there exists a gadget $S^\star \in \mathcal{C}$ and a candidate-direction pair $(v^\star, d^\star) \in \mathsf{Cand}(S^\star) \times \mathcal{D}$ such that $\mathsf{Ray}(v^\star, d^\star) \cap V(G) = \emptyset$.*

Proof. Fix $\sigma \in \{\pm 1, \pm 1\}$. Let H be a horizontal gadget on B_h^σ maximizing $\sigma_2 j$ and let V be a vertical gadget on B_v^σ maximizing $\sigma_1 i$, whenever such gadgets exist. If either H or V has an unobstructed outward boundary direction toward c_σ, then it is KP-feasible. Assume therefore that both H and V, whenever they exist, are blocked in the outward directions d_h^σ and d_v^σ. Starting from H and V, successively take first blockers in the corresponding outward directions. At each step, the blocker does not increase the σ-distance δ_σ and moves strictly toward the hull corner c_σ along at least one of the two directions d_h^σ or d_v^σ. Intuitively, this holds since choosing any blocker towards d_h^σ or d_v^σ (which are orthogonal directions) is guaranteed to be a gadget whose σ-distance is not larger than the current gadget. Observe that these gadgets with non-increasing σ-distance are broadly contained within the diagonal region between the current gadget and the corner c_σ. With each iteration, this process will reduce the region or set of possible blockers to consider, since by structure backtracking is not possible. Since the hull is bounded, this blocker process cannot continue indefinitely. Hence it terminates at a gadget $S^\star$ that has no blocker in one of its relevant outward directions, i.e., offering an unobstructed knock direction $d^\star$ for candidate $v^\star = x_\sigma(S^\star)$. Therefore $S^\star$ is KP-feasible. Note that the guaranteed existence of such a gadget is of interest, not the precise process of finding this gadget or its type. Hence, showing that such a $S^\star$ should exist suffices for our arguments. □

Theorem 1 (Optimal and feasible knock set). *Let $G^\circ = \mathsf{Clean}(G(\mathcal{B}))$ and $\mathcal{U} = \mathsf{Faces}(G^\circ)$. Let $\mathcal{C}^\star$ be a minimum-cardinality exact cover of $(\mathcal{U}, \mathcal{S})$ obtained from a maximum-weight perfect matching of $\mathcal{G}_F$. Then: (i) The minimum number of knocks in any complete KP execution equals $|\mathcal{C}^\star|$; (ii) There exists a KP-feasible execution achieving exactly $|\mathcal{C}^\star|$ knocks.*

Proof. (i) Optimality: minimum knocks $= |\mathcal{C}^\star|$. Consider any complete KP execution starting from G°. For each face $f \in \mathcal{U}$, let its first destroyed vertex be assigned to the knock that destroys it. By Lemma 3, this first destruction must be a knock. By Lemma 4, each knock destroys at most two faces, and two faces can be destroyed together only when they form a horizontal or vertical two-face gadget. Hence the knocks induce a pairwise-disjoint family of sets from $\mathcal{S}$ covering $\mathcal{U}$, that is, an exact cover $\mathcal{C}$ with $|\mathcal{C}| \leq$ #knocks. Therefore, #knocks $\geq \min\{|\mathcal{C}| : \mathcal{C} \text{ exact cover}\} = |\mathcal{C}^\star|$. Conversely, each set in an exact cover can be destroyed by one knock (Lemma 2), so $|\mathcal{C}^\star|$ knocks suffice in principle. This proves (i).

(ii) Every exact cover has a KP-feasible execution. Let $\mathcal{C}$ be an exact cover of $\mathcal{U} = \mathsf{Faces}(G^\circ)$. We argue by induction on $|\mathcal{C}|$. If $\mathcal{U} = \emptyset$, exhaustive picking removes all remaining vertices and the execution terminates. Assume $\mathcal{U} \neq \emptyset$, and

Algorithm 1 OPTKPSolver

Input: Set of blocks $\mathcal{O}$ with their grid-arranged locations
Output: A complete KP execution π

$G \leftarrow$ getGridGraph($\mathcal{O}$)
$\pi_{\text{PICKS}} \leftarrow$ pick(G) ▷ initial feasible pick sequence
$\pi \leftarrow \pi \oplus \pi_{\text{PICKS}}$; $G \leftarrow G \setminus \pi_{\text{PICKS}}$
$\mathcal{U} \leftarrow$ getFaces(G)
$\mathcal{G}_F \leftarrow$ buildFaceGraph($\mathcal{U}, G$)
$\mathcal{M}^\star \leftarrow$ maxWeightPerfectMatching($\mathcal{G}_F$)
$\mathcal{C}^\star \leftarrow$ edgeCoverToGadgets($\mathcal{M}^\star$)
while $G.V \neq \emptyset$ **do**
  $(v, d) \leftarrow$ chooseKnock($\mathcal{C}^\star, G$) ▷ next feasible knock
  $\pi \leftarrow \pi \oplus (v, d)$; $G \leftarrow G \setminus \{v\}$
  $\pi_{\text{PICKS}} \leftarrow$ pick(G) ▷ next feasible pick sequence
  $\pi \leftarrow \pi \oplus \pi_{\text{PICKS}}$; $G \leftarrow G \setminus \pi_{\text{PICKS}}$
end while
return π

fix any $\sigma \in \{\pm 1\}^2$. By Lemma 7, the σ-outer boundary construction determines a KP-feasible gadget $S^\star \in \mathcal{C}$. Execute knocking on $S^\star$ and then apply exhaustive picking. By Lemma 2, the face(s) covered by $S^\star$ are destroyed, so if $\mathcal{C}' \triangleq \mathcal{C} \setminus \{S^\star\}$, then $\mathcal{C}'$ is an exact cover of the remaining face set $\mathsf{Faces}(G')$, where G' is the resulting cleaned graph. By induction, G' admits a complete KP execution using exactly $|\mathcal{C}'|$ knocks. Prepending the feasible knock of $(v^\star, d^\star)$ yields a complete KP execution for G using exactly $1 + |\mathcal{C}'| = |\mathcal{C}|$ knocks. Applying this to $\mathcal{C} = \mathcal{C}^\star$ yields a complete KP-feasible execution using exactly $|\mathcal{C}^\star|$ knocks.

□

4.5 Optimal KP Solver

Theorem 1 characterizes the optimal knock count via an optimal exact cover (equivalently, a maximum-weight perfect matching in $\mathcal{G}_F$). To produce an *executable* KP sequence under the ray constraint (1), we use a local selection routine that checks ray feasibility and respects the gadget-dependent candidate sets.

Algorithm 1 alternates exhaustive picking with single feasible knocks. After initial cleanup, it extracts the face set, builds the face-adjacency graph, and computes an optimal perfect matching $\mathcal{M}^\star$, which yields a minimum-cardinality exact cover $\mathcal{C}^\star$ by Lemma 5. By Theorem 1, $|\mathcal{C}^\star|$ is the optimal knock count and $\mathcal{C}^\star$ admits a KP-feasible execution. The routine chooseKnock selects a ray-feasible candidate from the remaining gadgets of $\mathcal{C}^\star$, and each such knock is followed by exhaustive picking until the graph is empty. Hence the returned execution is complete and optimal in the number of knocks.

5 Results

We evaluate OPTKPSolver in two settings: (i) a fast *synthetic* benchmark that isolates the combinatorial structure, and (ii) an *IsaacSim* benchmark that

executes the resulting KP sequence with a Franka Panda arm under physics. Across all experiments, the *knock set* is computed by our *optimal* reduction to maximum-weight perfect matching on the face graph (Section 4). Hence all reported knock counts are globally optimal for the cleaned instance.

5.1 Synthetic Benchmark

Setup. We consider two families of instances: (i) *full grids* of sizes 3×3, 5×5, 10×5, 10×10, 20×10, and 20×20; and (ii) *subgraphs* formed by uniformly deleting vertices from the corresponding full grid to obtain $|V| \in \{5, 16, 25, 50, 100, 200\}$. For each subgraph size we generate $N = 20$ random instances and report averages. All synthetic experiments are run on an AMD Ryzen 9 9950X 16-Core CPU. The maximum-weight perfect matching is computed using the NetworkX [7] implementation of Edmonds' blossom algorithm [5].

Metrics. We report (i) the optimal number of knocks, and (ii) total solver time t_{total} (ms), which includes computing the matching / exact cover and producing an executable knock–pick sequence. The number of actions is always $|V|+\#\text{knocks}$ (each object is removed exactly once, and each knock adds one action), so we omit it from the tables.

Table 1. Synthetic results (optimal knock count via maximum-weight perfect matching) on full grids and random subgraphs.

grid dims	$\lvert V\rvert$	#knocks	t_{total} (ms)
3x3	9	2.00	0.55
5x5	25	8.00	1.47
10x5	50	18.00	3.51
10x10	100	41.00	13.05
20x10	200	86.00	61.62
20x20	400	181.00	189.79
3x3	5	1.00	0.13
5x5	16	3.12	0.47
10x5	25	7.00	1.43
10x10	50	14.04	3.03
20x10	100	27.36	9.40
20x20	200	60.16	36.02

Observations. Table 1 shows the statistics from the synthetic benchmark. On full grids, the optimal knock count closely tracks half the number of unit faces. Indeed, a full $m \times n$ grid contains $|\mathcal{U}| = (m-1)(n-1)$ faces, and the matching-based optimum prefers weight-2 edges whenever possible, yielding $\lceil |\mathcal{U}|/2 \rceil$ knocks (e.g., 20×20 has 361 faces and the optimum is 181 knocks). For subgraphs, random removal of vertices allows more blocks to be pickable and the face graph is sparser, reducing the knock-to-face ratio as expected.

In terms of computation, t_{total} increases smoothly with instance size and remains well below a second up to $|V| = 400$. This matches the polynomial-time nature of maximum-weight matching and the subsequent linear-time sequence construction in our pipeline.

5.2 IsaacSim Benchmark

Setup. We execute the KP sequences with a Franka Panda (7-DoF) and parallel gripper in Isaac Sim 5.0.0 using the PhysX engine. Motion planning is performed in ROS2 [16] and MoveIt2 task constructor [6], using RRTConnect [14]

Table 2. IsaacSim results. Knock counts are optimal for the cleaned instance. `knRetry` reports the mean number of physics-triggered knock retries per run.

grid dims	$\|V\|$	run succ.	obj pick succ.	#knocks	t_{comp} (s)	t_{exec} (s)	`knRetry`
3x3	9	1.0	1.00	2.0	4.50	92.07	0.70
5x5	25	0.8	0.97	8.0	13.56	272.48	5.00
3x3	5	1.0	1.00	1.0	2.44	51.03	0.56
5x5	16	1.0	1.00	3.6	8.49	165.41	1.44

in OMPL [23] for each pick/knock motion. Experiments are run on an RTX 5090 GPU workstation. We test full grids of size 3×3 and 5×5 (10 runs each), and random subgraphs with $|V|$=5 (from 3×3) and $|V|$=16 (from 5×5). For each subgraph size we use 5 random problems and repeat each 10 times (50 runs total).

Metrics. We report success rate, optimal knock count, computation time t_{comp} (MoveIt/OMPL), execution time t_{exec} (IsaacSim), all in seconds. We additionally report `knRetry`, the mean number of knock retries per run: if a knock does not create sufficient clearance due to contact/friction/torque effects, the same knock motion is repeated until clearance is detected or the run is declared failed.

Observations. Table 2 shows the statistics of the IsaacSim benchmark. Execution dominates total time: physical simulation execution time t_{exec} is one to two orders of magnitude larger than motion planning time t_{mp} across all cases, since each action requires approach, contact, and retraction under physics. While the *number of actions* scales as $|V| + \#\text{knocks}$, the wall-clock execution time is also sensitive to physical interactions during knocks.

The `knRetry` column makes this explicit: larger and denser instances (notably full 5×5) require more retries on average, reflecting the fact that a nominally valid knock (ray-feasible in the discrete model) may under-deliver displacement when realized with a finite-stiffness gripper and frictional contacts. This same effect explains why the full 5×5 case exhibits runs where primitive failures were recorded: some runs enter unrecoverable contact configurations where repeated knock attempts cannot produce sufficient clearance. Despite primitives failing in two runs (run succ. of 0.8), proceeding with the remaining actions succeeded to exhaustively pick almost all of the objects (pick success of 0.97).

6 Discussion

This work studies grid-aligned, tightly packed tabletop rearrangement where purely prehensile removal with a parallel gripper can be blocked. Allowing a ray-constrained directional knock enables progress, and the minimum-knock objective admits an exact combinatorial characterization: after cleanup, optimal knock sets correspond to minimum exact face covers, computed optimally via

matching on the face graph. Executability is recovered by interleaving one ray-feasible knock (chosen from gadget-specific candidate vertices) with exhaustive picking, which monotonically reduces the instance until all blocks are removed.

Empirical observations. The synthetic benchmark indicates that the optimal solver scales smoothly with instance size and that runtime grows polynomially with the number of objects/faces, consistent with the matching-based reduction. In Isaac Sim, the same optimal knock counts are realized, while total wall-clock time is dominated by physics-based execution rather than the combinatorial solver. Knock retries were occasionally required: even when ray-clearance holds, contact dynamics (e.g., friction, gripper compliance, impulse transfer) can prevent a block from fully leaving its pocket. This highlights the gap between discrete clearance abstractions and continuous rigid-body interactions.

Limitations. The setting being studied is heavily simplified — top-down reachable, uniformly sized, blocks arranged in a grid, with hand-crafted non-prehensile primitives. The setting allows us to glean theoretical insights but extending to practical robust performance still poses open, challenging questions. The Sim-to-Real gap will be a persistent bottleneck in applying combinatorial graphical abstractions to real-world executions with physics. The primitive itself is simplistic and represents the minimal operation needed to locally relax adjacency constraints. There may be more sophisticated contact-rich non-prehensile primitives, including aggregating push strategies that may benefit some tasks. Bespoke fingered end-effectors may pose their own constrained abstraction and complexity. The placement side of the problem is also assumed to be unconstrained, which though valid in non-overlapping situations, need to be explored in general overlapping scenaros. Moreoever, theoretical insights into generalized object shapes start converging on typical considerations within general assembly planning and it is to be seen whether the current theoretical efficiencies can be used to speed up more general settings.

Despite the current result being a preliminary step, the theoretical insight into the studied category of problems permitting computationally efficient solutions to compute optimal task plans, promises opportunities to build principled theoretically-driven practically-robust strategies that can seamlessly switch between diverse prehensile and non-prehensile strategies that balance practical performance with theoretical guarantees.

References

1. Ben-Shahar, O., Rivlin, E.: Practical Pushing Planning for Rearrangement Tasks. IEEE Transactions on Robotics and Automation **14**(4) (August 1998)
2. Chi, C., Xu, Z., Feng, S., Cousineau, E., Du, Y., Burchfiel, B., Tedrake, R., Song, S.: Diffusion policy: Visuomotor policy learning via action diffusion. The International Journal of Robotics Research p. 02783649241273668 (2023)
3. Dantam, N.T., Kingston, Z.K., Chaudhuri, S., Kavraki, L.E.: An incremental constraint-based framework for task and motion planning. The International Journal of Robotics Research **37**(10), 1134–1151 (2018)

4. Dogar, M., Srinivasa, S.: A framework for push-grasping in clutter. Robotics: Science and Systems VII **1**, 65–72 (2011)
5. Galil, Z.: Efficient algorithms for finding maximum matching in graphs. ACM Computing Surveys (CSUR) **18**(1), 23–38 (1986)
6. Görner, M., Haschke, R., Ritter, H., Zhang, J.: Moveit! task constructor for task-level motion planning. In: 2019 International Conference on Robotics and Automation (ICRA). pp. 190–196. IEEE (2019)
7. Hagberg, A., Swart, P.J., Schult, D.A.: Exploring network structure, dynamics, and function using networkx. Tech. rep., Los Alamos National Laboratory (LANL) (2007)
8. Halperin, D., Latombe, J.C., Wilson, R.H.: A general framework for assembly planning: The motion space approach. In: Proceedings of the Fourteenth Annual Symposium on Computational Geometry. pp. 9–18 (1998)
9. Han, S.D., Stiffler, N., Krontiris, A., Bekris, K., Yu, J.: Complexity results and fast methods for optimal tabletop rearrangement with overhand grasps. International Journal of Robotics Research (2018)
10. Havur, G., Ozbilgin, G., Erdem, E., Patoglu, V.: Geometric rearrangement of multiple movable objects on cluttered surfaces: A hybrid reasoning approach. In: International Conference on Robotics and Automation (2014)
11. Höfer, S., Bekris, K., Handa, A., Gamboa, J.C., Mozifian, M., Golemo, F., Atkeson, C., Fox, D., Goldberg, K., Leonard, J., et al.: Sim2real in robotics and automation: Applications and challenges. IEEE Transactions on Automation Science and Engineering **18**(2), 398–400 (2021)
12. Huang, E., Jia, Z., Mason, M.T.: Large-scale multi-object rearrangement. In: 2019 International Conference on Robotics and Automation (ICRA). pp. 211–218. IEEE (2019)
13. Huang, W.H., Mason, M.T.: Mechanics, planning, and control for tapping. The International Journal of Robotics Research **19**(10), 883–894 (2000)
14. Kuffner, J.J., LaValle, S.M.: RRT-connect: an efficient approach to single-query path planning. In: Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065). vol. 2, pp. 995–1001. IEEE (2000)
15. Lynch, K.: Toppling manipulation. In: Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C). vol. 4, pp. 2551–2557 vol.4 (1999). https://doi.org/10.1109/ROBOT.1999.773981
16. Macenski, S., Foote, T., Gerkey, B., Lalancette, C., Woodall, W.: Robot operating system 2: Design, architecture, and uses in the wild. Science Robotics **7**(66), eabm6074 (2022). https://doi.org/10.1126/scirobotics.abm6074
17. Ota, J.: Rearrangement Planning of Multiple Movable Objects. In: Prof. of the IEEE Intern. Conference on Robotics and Automation (ICRA) (2004)
18. Quintero-Pena, C., Kingston, Z., Pan, T., Shome, R., Kyrillidis, A., Kavraki, L.E.: Optimal grasps and placements for task and motion planning in clutter. In: International Conference on Robotics and Automation (ICRA). IEEE (2023)
19. Shome, R., Tang, W.N., Song, C., Mitash, C., Kourtev, H., Yu, J., Boularias, A., Bekris, K.E.: Towards robust product packing with a minimalistic end-effector. In: IEEE International Conference on Robotics and Automation (ICRA) (2019)
20. Song, C., Boularias, A.: Object rearrangement with nested nonprehensile manipulation actions. In: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 6578–6585. IEEE (2019)

21. Song, H., Haustein, J.A., Yuan, W., Hang, K., Wang, M.Y., Kragic, D., Stork, J.A.: Multi-object rearrangement with monte carlo tree search: A case study on planar nonprehensile sorting. In: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 9433–9440. IEEE (2020)
22. Suárez-Ruiz, F., Pham, Q.C.: A framework for fine robotic assembly. In: 2016 IEEE International Conference on Robotics and Automation (ICRA). pp. 421–426. IEEE (2016)
23. Sucan, I.A., Moll, M., Kavraki, L.E.: The open motion planning library. IEEE Robotics & Automation Magazine **19**(4), 72–82 (2012)
24. Tian, Y., Willis, K.D., Al Omari, B., Luo, J., Ma, P., Li, Y., Javid, F., Gu, E., Jacob, J., Sueda, S., et al.: Asap: Automated sequence planning for complex robotic assembly with physical feasibility. In: 2024 IEEE International Conference on Robotics and Automation (ICRA). pp. 4380–4386. IEEE (2024)
25. Toussaint, M.: Logic-geometric programming: An optimization-based approach to combined task and motion planning. In: International Joint Conference on Artificial Intelligence (2015)
26. Vieira, E.R., Nakhimovich, D., Gao, K., Wang, R., Yu, J., Bekris, K.E.: Persistent homology for effective non-prehensile manipulation. In: 2022 International Conference on Robotics and Automation (ICRA). pp. 1918–1924. IEEE (2022)
27. Wang, F., Hauser, K.: Dense robotic packing of irregular and novel 3d objects. IEEE Transactions on Robotics **38**(2), 1160–1173 (2021)
28. Wen, B., Lian, W., Bekris, K., Schaal, S.: You only demonstrate once: Category-level manipulation from single visual demonstration. In: 18th Robotics: Science and Systems, RSS 2022. MIT Press Journals (2022)