2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

ASAP: Automated Sequence Planning for Complex Robotic Assembly with Physical Feasibility

Yunsheng Tian[1,†], Karl D.D. Willis[2], Bassel Al Omari[3,†], Jieliang Luo[2], Pingchuan Ma[1], Yichen Li[1], Farhad Javid[2], Edward Gu[1], Joshua Jacob[1], Shinjiro Sueda[4], Hui Li[2], Sachin Chitta[2] and Wojciech Matusik[1]

***Abstract*— The automated assembly of complex products requires a system that can automatically plan a physically feasible sequence of actions for assembling many parts together. In this paper, we present ASAP, a physics-based planning approach for automatically generating such a sequence for general-shaped assemblies. ASAP accounts for gravity to design a sequence where each sub-assembly is physically stable with a limited number of parts being held and a support surface. We apply efficient tree search algorithms to reduce the combinatorial complexity of determining such an assembly sequence. The search can be guided by either geometric heuristics or graph neural networks trained on data with simulation labels. Finally, we show the superior performance of ASAP at generating physically realistic assembly sequence plans on a large dataset of hundreds of complex product assemblies. We further demonstrate the applicability of ASAP on both simulation and real-world robotic setups. Project website: asap.csail.mit.edu**

I. Introduction

Real-world products often involve complex assemblies, necessitating specially designed assembly lines for their assembly, maintenance, and repair. In manufacturing, these *fixed* assembly lines are highly efficient for individual assemblies. However, creating them demands significant effort for design, programming, and setup, and they struggle to adapt to different assemblies. Particularly for high-mix, low-volume products, there is a need for automated assembly lines that are *flexible* and easily repurposed.

Assembly automation begins with understanding the sequence for assembling parts, typically done manually by experienced manufacturing teams. However, generating physically feasible assembly sequences remains a challenging research problem for several reasons: (1) Sequence Planning: The number of potential assembly sequences grows exponentially with part count, with not all sequences being equally good or feasible. (2) Physical Feasibility: Planned sequences may not be physically executable due to unsatisfied physical constraints. For example, a collision-free path might not exist for a given assembly order, or parts could fall during assembly due to instability. (3) Geometric Complexity: The geometry of objects can be highly complex, such as gears with numerous faces. Planning collision-free paths or assessing gravitational stability accurately in these scenarios is non-trivial.

[1]MIT CSAIL {yunsheng, pcma, yichenl, jmjacob, wojciech}@csail.mit.edu, egu@mit.edu
[2]Autodesk Research {karl.willis, rodger.luo, farhad.javid, hui.xylo.li, sachin.chitta}@autodesk.com
[3]University of Waterloo b2alomar@uwaterloo.ca
[4]Texas A&M University sueda@tamu.edu
†Work partially done while interning at Autodesk Research.

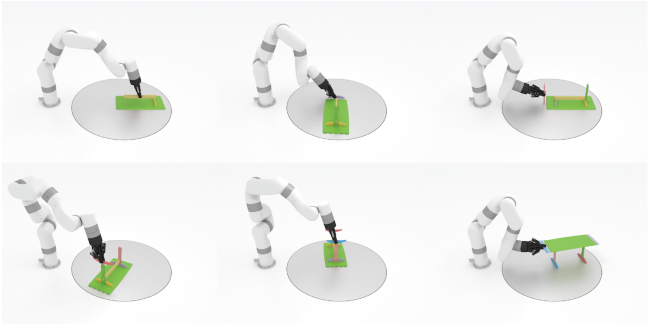

Fig. 1: Assembly plans generated autonomously from ASAP for a desk positioned on a rotary table including physically feasible assembly sequences, collision-free paths, gravitationally stable poses, gripper grasps, and robot arm motion.

We tackle these challenges with **ASAP** (**A**utomated **S**equence Planning for Complex Robotic **A**ssembly with **P**hysical Feasibility). To guarantee physical feasibility of assembly sequences, we use physics-based simulation to check for collision-free paths. We also introduce an efficient method to verify gravitational stability and identify parts to be held for stability. Additionally, we find the most stable pose when placing the assembly on a support surface. To speed up planning and reduce the combinatorial search space, we employ the assembly-by-disassembly principle and efficient tree search algorithms. We guide the sequence search using geometric heuristics or graph neural networks (GNNs) trained on simulation-based data.

In summary, our work makes the following contributions:

1) An efficient assembly sequence planning algorithm for generating physically feasible sequences for complex-shaped, contact-rich assemblies, ensuring both collision-free paths and gravitational stability.
2) A learning-based approach for selecting the next part to disassemble, using GNNs trained on a large dataset of product assemblies.
3) A greedy approximate gravitational stability checking algorithm that more efficiently determining unstable parts for stabilization compared to the combinatorially expensive naive approach.
4) Evaluation on a large dataset of hundreds of complex product assemblies, demonstrating state-of-the-art performance compared to established baselines.
5) Integration of grasp planning and robot arm inverse kinematics, guaranteeing feasible robotic execution in both simulation and real-world experiments.

II. Related Work

A. Assembly sequence planning

To automatically determine feasible assembly sequences and paths, the non-directional blocking graph [1] is proposed but limited to simple geometries due to computational complexity. Later, motion planning methods were introduced for complex-shaped objects either by geometric collision checking [2], [3], [4] or leveraging physics simulation [5]. For real-world assemblies, gravitational stability is also considered during planning [6], [7]. However, evaluating the physical feasibility of complex-shaped assemblies accurately is difficult and time-consuming. To speed up the feasibility check, [8] propose an iterative procedure on planar profile assembly by checking complex dynamic constraints later only if simpler static constraints are all satisfied.

To tackle the combinatorial complexity of sequence optimization, evolutionary methods have been proposed, such as genetic algorithms [9], [10], [11], ant colony optimization [12]. However, these methods require evaluating massive sequences in order to be successful; hence their evaluation metrics are usually simple and do not consider gravitational stability, preventing the direct deployment of their optimized sequences in a real-world setup. More efficient search techniques have been explored in [13], [14] to specifically showcase stable assembly of frame shapes and bar structures.

In contrast to prior work, we demonstrate our sequence planner on a complex general-shaped 3D assembly dataset derived from the Fusion 360 Gallery Assembly Dataset [15] and a real mechanical assembly dataset [16], featuring more complex shapes than previously considered. While [5] also use the dataset from [15] to demonstrate a sequence planner, they only consider geometric motion constraints in a gravity-free setting and perform assembly in midair. In our work, we consider more physical constraints, such as gravity and friction, as well as robotic constraints, in determining the feasibility of each assembly step. Additionally, we take into account the use of a flat support surface, stable poses, and grippers for holding unstable parts.

B. Physics-based simulation for assembly

Unlike pure geometric analysis, physics-based simulation aids in assessing the physical feasibility of (dis)assembly sequences. Prior work, such as [7] and [17], employs physics engines for stability analysis in disassembly sequences. However, their experiments are limited to simple block-stacking scenarios instead of complex shapes. In [18], physical simulation constructs a physically reasonable graph to speed up searches, tested on box-on-table and peg-in-hole tasks. Reinforcement learning with physical simulation is also used for learning robotic control policies [19], [20], [21], mainly focusing on simple insertion tasks like peg-in-hole or lap-joint due to limitations of convex-hull-based simulators. Recently, [5] and [22] utilize simulators based on Signed Distance Field (SDF) to simulate contact-rich complex assemblies. However, none of these studies explore sequence generation for multi-part assemblies that are both stable and executable in the real world.

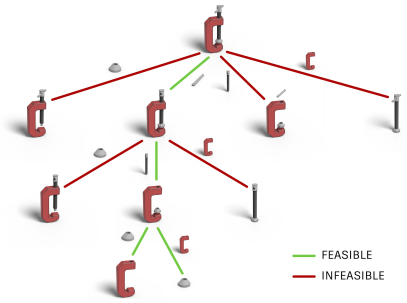

Fig. 2: An example disassembly tree where nodes represent partial assemblies and edges represent feasible (green) and infeasible (red) disassembly actions.

C. Learning for assembly sequence planning

Learning-based methods are promising to tackle the combinatorial problem of assembly sequence generation. Self-supervised learning [23] collects data by trial-and-error kit disassembly but is limited to 2D shape-matching tasks. Reinforcement learning methods [24], [25], [26] can train effective 3D assembly planners but demonstrate on simplified physics or geometries due to high sample complexity. Our approach employs supervised learning inside planning, using a GNN to predict disassembly sequences with physical feasibility for complex 3D assemblies. We gather training data from diverse assemblies generated via physics-based simulation, and the network shows effective guidance for planning disassembly sequences on unseen assemblies.

III. Physically Feasible Sequence Planning

Our objective is to plan an assembly sequence under physical constraints using individual parts and their assembled states as input. The output sequence details each assembly step, including the part to assemble, its motion path, assembly pose, and other parts to be held. Solving them jointly yields a complete and physically feasible sequence.

Searching in a combinatorial sequence space is challenging, especially with realistic physical constraints. To reduce the complexity, we employ the assembly-by-disassembly concept [27], obtaining assembly sequences from reverse disassembly sequences due to a bijection between them for rigid parts. In this section, we propose a tree search algorithm that efficiently prioritizes part disassembly by leveraging geometric heuristics or neural guidance, and identifies stable poses that satisfy physical constraints.

A. Disassembly tree search

We formulate the sequence planning as a tree-search framework, utilizing established techniques for efficient exploration. The core data structure is a disassembly tree, as shown in Fig. 2. Each path of the tree represents a particular way of how the assembly is disassembled from top to bottom.

Algorithm 1 outlines our disassembly tree search framework. Given the fully assembled assembly G_0 with n parts, our goal is to find a physically feasible disassembly sequence within the total simulation evaluation budget N. In our algorithm, we build and iteratively update a disassembly tree until a feasible sequence is found. The success of the

Algorithm 1: Physics-Based Disassembly Tree Search

Input: Assembly $G_0 = \{p_1, ..., p_n\}$, evaluation budget N.
Output: A physically-feasible disassembly sequence.

$T \leftarrow$ EmptyTree(); $i = 0$;
T.AddNode(G_0, valid=true);
while $i < N$ and search not complete **do**
  $G \leftarrow$ SelectNode(T); // §III-B
  for part p in SelectPart(G) **do** // §III-C
    $G' \leftarrow G \setminus \{p\}$;
    if T.HasEdge(G, G') **then continue**;
    for pose s in SelectPose(G) **do** // §III-D
      $P_a \leftarrow$ CheckAssemblable(G', p, s); // §IV-B
      $P_g \leftarrow$ CheckStable(G', s); // §IV-C
      $i = i + 1$;
      if $P_a \neq$ null and $P_g \neq$ null **then**
        T.AddNode(G', valid=true);
        $e \leftarrow T$.AddEdge(G, G', valid=true);
        e.StoreInfo(s, P_a, P_g);
        break; // feasible step
    if not T.HasNode(G') **then**
      T.AddNode(G', valid=false);
    if not T.HasEdge(G, G') **then**
      T.AddEdge(G, G', valid=false);
    else if $|G'| = 1$ **then** // complete sequence
      return T.FindPath(G_0, G');
    break;
return null;

algorithm relies on the following implementations to expand the tree in each iteration, as highlighted in Algorithm 1:

- SelectNode(T): which node to expand tree T from (i.e., which partial assembly to disassemble).
- SelectPart(G): which part to disassemble from node G.
- SelectPose(G): under which pose to disassemble G.

In each iteration, we first select a node G from the tree T to expand, guided by the specific tree search method in SelectNode(T). After that, we decide which part p to disassemble from G using SelectPart(G). The remaining assembly without p is referred to as G'. Finally, we select a pose s to orient G and attempt to disassemble part p.

The feasibility of disassembly is determined by: (1) CheckAssemblable(), which gives a disassembly motion plan P_a (§IV-B) and (2) CheckStable(), which gives a part holding plan P_g to ensure stability (§IV-C). If both procedures return feasible solutions, the tree expansion is successful. Otherwise, we continue searching for new poses and different paths to expand the tree until either a feasible sequence is found, or the budget is reached, or the tree is fully expanded.

B. Node selection

Any standard tree traversal method can be applied to select a node to expand in each iteration, i.e., SelectNode() in Algorithm 1. For simplicity, we apply a standard implementation of depth-first search (DFS) that explores the tree as far as possible along each branch before backtracking.

C. Part selection

1) Geometric heuristic: To wisely select a part to disassemble, we leverage the insight that exterior parts are generally easier to disassemble due to fewer precedence constraints. We use the distance between a part's center of mass and the assembly center (bounding box center) to prioritize outside-in part selection.

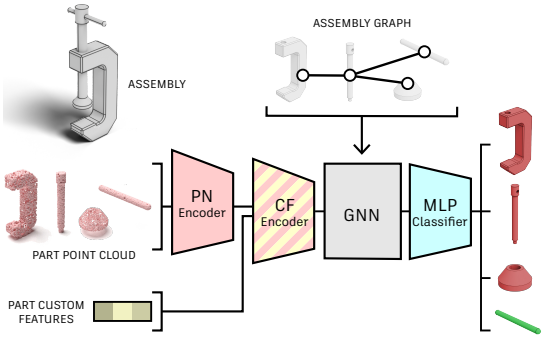

Fig. 3: Network architecture for learning part disassembly priority. Given an assembly as input, a graph neural network is used to predict the next part to be removed.

2) Learning-based: We also implement a learning-based approach to accelerate the tree search described in Algorithm 1. Our network is trained by supervision on the task of node classification to predict the next part to disassemble. 6,368 disassembly sequence labels are generated from 1,906 training assemblies out of 2,146 total assemblies from the dataset described in §V-A using simulation following a dynamic beam search to balance data diversity and computational efficiency. Inspired by previous works on learning for furniture assembly [28], [29] and CAD modeling [15], we implement a GNN model that takes as input a graph G with n parts $\{p_1, p_2, ..., p_n\}$ forming graph nodes connected by adjacency. The network outputs a probability distribution over all nodes, indicating the likelihood of each part being next in the disassembly sequence.

Fig. 3 illustrates the network architecture consisting of four modules: a PointNet encoder (PN encoder) [30], a custom feature encoder (CF encoder), GNN message passing layers, and an MLP classifier. The two encoders extract two kinds of per-part features. The PN encoder extracts part geometric features from uniformly sampled point clouds (1K points per part), and the CF encoder provides additional information, such as part volume, part distance to the assembly center, and the number of adjacent parts. The two kinds of features together form a graph node embedding. Two rounds of message passing are used to update the graph node features between connected neighboring parts. Finally, the MLP classifier predicts the next part to disassemble from the final graph node embedding. We use a standard binary cross-entropy loss as the supervision signal between the predicted probability to disassemble a given part next $\hat{p}_i$ and the ground truth label p_i^{gt} indicating the next part to disassemble.

D. Pose selection

Since it is impractical to evaluate infinite poses in a 3D space for a given assembly, we propose to use quasistatic stable poses [31] of assembly G as the pose candidates to select from (top-5 is empirically good enough), which provides a much reduced and reasonable search space. Furthermore, we observe that reusing poses during the assembly generates more intuitive sequences and also improves success rate.

IV. Feasibility check for assembly sequences

This section describes how we leverage physics-based simulation to check whether an assembly sequence is physically feasible. First, §IV-A describes our physics-based assembly simulation; Then, §IV-B focuses on finding collision-free paths for parts to be (dis)assembled using a physics-based path planning approach. Finally, §IV-C describes our method to check whether a given (partial) assembly can be gravitationally and frictionally stable, taking into account the support surface and a subset of parts to be held.

A. Physics-based simulation for assembly

Our physics-based model is designed to work robustly with complex geometry for assembly and is able to handle minor problems in the input geometry, including small overlaps, potentially conflicting collision normals, and slight gaps. We use the penalty-based contact model for the contact normal force and the frictional forces [32], [33]. We use implicit Euler to integrate the system forward, which allows us to handle these penalty forces robustly and stably. The penetration distance is computed accurately and efficiently with pre-computed signed distance fields (SDF).

B. Disassembly path planning

To check whether a part can be assembled, we leverage again the idea of assembly-by-disassembly where disassembly path planning can be applied to connect the part's assembled state to a disassembled state. Motivated by the recent success of physics-based path planning, we adopt [5] to generate a collision-free assembly path for each part. Specifically, the planner starts from the assembled state and searches for a sequence of forces to apply until a disassembled state or a predefined budget is reached.

C. Gravitational stability check

Existing works using physics simulation to check stability are limited to simple block-stacking scenarios with a fixed pose on the support surface [7], [17]. However, complex real-world assemblies usually require at least two hands for assembly and re-orientation since the stable pose might change as more parts are assembled. To address this challenge, we develop an efficient physics-based stability check algorithm for a given assembly G with n parts under a specific pose s, represented as a homogeneous transformation matrix. The algorithm outputs the set of maximum M parts to be held to make G stable under pose s or returns null if it needs to hold more than M parts to stabilize G.

A naive way to discover feasible part-holding plans is by trying all $O(n^M)$ combinations of parts to hold, which is impractical for large assemblies. Therefore, we introduce a greedy approximation with $O(M)$ part-holding plans to search in Algorithm 2. The key idea is to start with no parts held and only hold parts when we observe a part falling until M parts are held. We use P_g to denote the set of parts we hold to ensure stability, which is an empty set initially and gets iteratively updated. In each iteration, we obtain the initial position of all the parts, denoted as $\mathbf{q}_{i_0}$ for part p_i.

Algorithm 2: Physics-Based Stability Check

Input: Assembly $G = \{p_1, p_2, ..., p_n\}$ with pose s, max M parts to be held, max simulation steps N, stable moving distance threshold d_{th}.
Output: Parts to be held to make G stable with pose s.

$P_g \leftarrow \{\}$;
while $|P_g| \leq M$ **do**
  ResetSimulation(G, s, P_g);
  for i in $1, ..., n$ **do** $\mathbf{q}_{i_0} \leftarrow$ GetPosition(p_i);
  $stable \leftarrow$ true;
  for j in $1, ..., N$ **do** `// N-step stability check`
    Simulate(Δt); `// run physics simulation`
    for i in $1, ..., n$ **do**
      $\mathbf{q}_{i_j} \leftarrow$ GetPosition(p_i);
      if $\|\mathbf{q}_{i_j} - \mathbf{q}_{i_0}\| > d_{th}$ or IsDisconnected(p_i, G) **then**
        $stable \leftarrow$ false; `// p_i falls`
        $P_g \leftarrow P_g \cup \{p_i\}$; `// hold p_i`
        break;
    if not $stable$ **then break**;
  if $stable$ **then return** P_g; `// stable for N steps`
return null; `// unstable`

Next, we run the physics simulation for maximum N time steps to check stability. In each step j, we obtain the updated position $\mathbf{q}_{i_j}$, and calculate the Euclidean distance between $\mathbf{q}_{i_j}$ and initial position $\mathbf{q}_{i_0}$. If the distance is larger than a threshold d_{th}, this means part p_i is falling and thus not stable. Apart from that, we also check if part p_i loses contact with its neighboring parts; if so, it is also unstable. If any part is unstable in step j, we add part p_i to P_g and restart the whole process. The algorithm stops until either a feasible P_g is found, or we cannot hold extra parts ($|P_g| > M$). Note that we use position change as the stability check criterion because it is smoother in time for contact-rich simulation, as opposed to numerically noisy velocity or acceleration in previous works [7], [17].

V. Evaluation

A. Dataset for assembly sequence planning

We build a large complex 3D assembly dataset derived from the Fusion 360 Gallery Assembly Dataset [15] and a real mechanical assembly dataset [16], consisting of 2,146 assemblies from 3 to 50+ parts with more complex shapes than those used previously for this problem. To ensure feasibility, we perform data filtering processes to make sure part meshes are watertight and able to be (dis)assembled without physical deformation, and remove parts that intersect with others in the assembly greater than a given distance. Among these assemblies, we select 240 assemblies to be the test set for benchmarking using the following procedures: First, we filter out all potentially unstable assemblies in the dataset by loading them into simulation and checking for stability using the top stable poses generated by §III-D. Next, we group the remaining dataset by the number of parts within the assembly and randomly sample up to 10 assemblies from each group to ensure a suitable representation of complex assemblies in the distribution. Finally, we combine all sampled assemblies to form the test set and the remaining 1,906 assemblies become our training set.

TABLE I: Success rate (%) comparison of ASAP on the assembly test dataset against several baseline methods.

Method		Success Rate (%) (Low Budget)			Success Rate (%) (High Budget)		
		2 Parts Held	3 Parts Held	4 Parts Held	2 Parts Held	3 Parts Held	4 Parts Held
ASAP (Ours)	Heuristics	51.25	61.25	68.75	66.67	74.17	80.83
	Learning	**54.58**	**62.92**	**69.58**	**67.08**	**76.25**	**82.08**
Baseline	Random Permutation	14.58	25.42	41.25	27.92	43.33	55.42
	Genetic Algorithm [9]	14.17	25.83	40.00	30.83	41.25	51.25
	Assemble Them All [5]	19.17	27.08	35.42	30.42	46.25	56.67

B. Baseline methods

We compare ASAP with a naïve baseline and two established methods that generate and optimize (dis)assembly sequences: (1) *Random Permutation* generates completely random (dis)assembly sequences and evaluates them until the budget is met, to serve as a lower bound on performance. (2) *Genetic Algorithm* [9], [11] starts from randomly permuted sequences and iteratively optimizes the sequences based on a *fitness function*, defined as the number of steps of the sequence that are physically feasible. (3) *Assemble Them All* [5] similarly leverages physics simulation to find assembly sequences by randomly selecting parts to disassemble and checking for collision-free disassembly paths. However it does not consider stability as a criterion, but assumes a gravity-free setting with no supporting surface and parts to be held. For comparison, we run *Assemble Them All* repeatedly to generate sequences until the budget is met. To take into account physical feasibility, we use our stability check algorithm in §IV-C under the top poses searched by the quasistatic pose estimator in §III-D to evaluate the feasibility of the generated sequences for all baseline methods.

C. Experimental setup

We benchmark two variants of our ASAP algorithm with different part selection methods (see §III-C). We run our experiments in parallel on Intel Xeon Platinum 8260 CPUs with 4GB RAM per core, where each experiment for a single assembly runs on a single CPU core. We set a 2-hour timeout budget for each experiment. For network implementation, all modules use 512 hidden neurons, ReLU activation, and batch normalization. The GNN uses two GATv2 [34] message-passing layers with eight heads. We withhold the test set described in § V-A and split the remaining data 90:10 for training and validation. Training takes 5 hours using an NVIDIA V100 GPU for 150 epochs, with a batch size of 160 and a learning rate of 0.001. We use the Adam optimizer with a scheduler that reduces the learning rate on the plateau.

D. Quantitative success rate comparison

We now compare the performance of our algorithms against the baseline methods using the *success rate* metric: the percentage of assemblies from the test dataset that can be successfully (dis)assembled, given a fixed simulation evaluation budget and a set number of parts to be held. To represent a realistic scenario, we show results for both a low budget (50 evaluations) and a high budget (400 evaluations) in Table I, holding 2, 3, and 4 parts respectively.

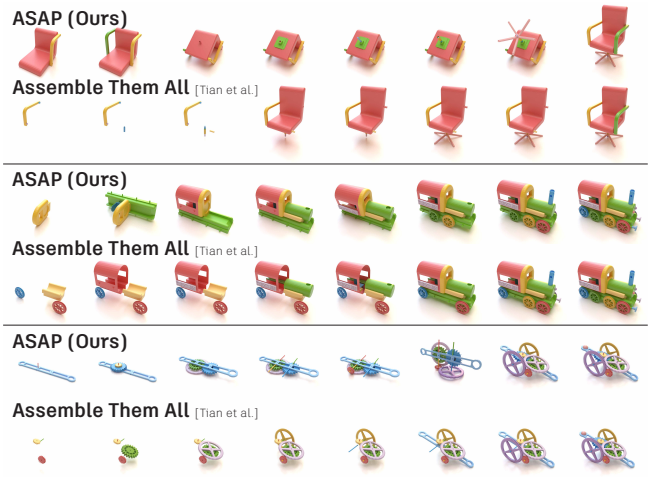

Fig. 4: Assembly sequence comparison between ASAP and *Assemble Them All* [5] on camera, train, and gear set.

The results show that both variants of our algorithm outperform the three baseline methods by a significant margin, thanks to the tree search formulation and efficient guidance on part selection that greatly reduces the search space. We observe improvements for all methods if allowing more parts to be held or given more evaluation budget, though a setup with two part holders and a limited budget is more practical. The outstanding performance of our *Learning* variant shows that the neural guidance trained on a diverse set of assemblies with simulation-generated labels generalizes well to unseen and more challenging assemblies in the test set.

We report the median runtime of ASAP with a single CPU thread on different sizes of assemblies: $\leq$5 parts: 43s; (5,10] parts: 97s; (10, 20] parts: 431s; (21, 30] parts: 1705s; >30 parts: 4190s. Significant speed up can be achieved through parallelization in several components of Algorithm 1.

We conduct quantitative comparisons without constraints from robotic manipulators, which isolates the impact of planning strategies on assembly success while abstracting away the nuances of robotic hardware and specialized tools. For methods and results on assembly planning with ASAP using a parallel gripper and a robot arm, please see §V-G.

E. Qualitative assembly sequence comparison

Fig. 4 shows a comparison between assembly sequences generated by our method and *Assemble Them All*. As described in §V-B, *Assemble Them All* does not take gravitational stability into consideration in assembly sequence generation, thus producing sequences with many floating parts and apparently unstable poses. In contrast, ASAP generates

TABLE II: Accuracy and speed up of the proposed stability check algorithm compared to the combinatorial ground truth (*TP* = true positive, *TN* = true negative, *FN* = false negative).

# Parts to Hold	TP (%)	TN (%)	FN (%)	Timeout (%)	Acc. (%)	Speed Up
2	58.5	19.2	9.6	12.7	89.0	13.90x
3	66.3	8.4	7.8	17.5	90.5	17.03x
4	69.8	4.0	4.1	22.1	94.7	23.04x

physically feasible sequences with a stability guarantee, which is more applicable to a real-world assembly setup.

F. Ablation studies

We compare our greedy stability check algorithm (Algorithm 2) to the full combinatorial check as the ground truth. We perform the comparison using 1,000 pairs of partial assemblies and poses randomly sampled from the data collected during simulation label generation. Table II shows the accuracy of our greedy approximation. Due to the complexity of the combinatorial algorithm, we limit each comparison with a 1-hour timeout and report the percentage of timeouts. The greedy algorithm achieves around 90% accuracy and an order of magnitude speed-up, which becomes more significant as the number of parts that can be held increases due to a more expensive combinatorial check.

We also implemented a custom beam search for node selection with dynamically growing beam width, which performs similarly as DFS. Regarding geometric heuristics for part selection, we found the outside-in selection scheme introduced in §III-C offers a slight advantage over prioritizing parts with small volume or few adjacent parts.

G. Robotic demonstration

We integrate ASAP with a robotic setup for real-world deployment. The motion of the robotic arm and gripper is governed by grasp planning, inverse kinematics (IK) and collision detection. To determine feasible grasping points on an object using a parallel gripper, we sample 100 pairs of antipodal points on its surface through ray casting and surface normal comparisons to ensure force closure. Each pair of antipodal points guides the placement of the gripper's fingertips, along with the selection of 5 orientations that positively align with the assembly direction. Next, we execute the assembly path with the chosen grasping pose. At each step of the path, we calculate the robot arm's joint angles via IK, using the gripper pose as input. A rotary table can be used to improve arm reachability, with the rotation angle optimized based on the desired gripper location and orientation. This process also includes thorough collision checking to ensure the robot arm, gripper, assembly parts, and the support surface remain collision-free. We prioritize grasp planning with poses that enhance stability and iterate until a collision-free and IK-feasible assembly plan is achieved.

Following the above approach, ASAP generates feasible robotic assembly plans for diverse assemblies (e.g., Fig. 1). While comprehensive simulated results can be found on our project website, here we demonstrate the sim-to-real transfer on a real-world hardware setup with a 3D printed beam assembly with 5 parts. The central component of the hardware setup is a UFACTORY xArm 7 robotic arm, which utilizes a Robotiq 2F-140 gripper. To aid in the spatial localization of assembly components, we use a laser-cut placemat that allows the robot to determine the precise positioning of parts, thereby reducing potential assembly errors.

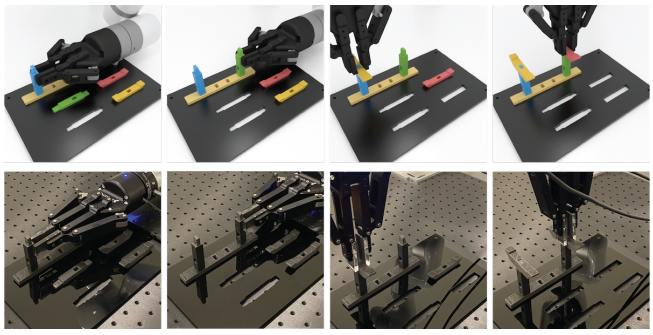

Fig. 5: Robotic assembly plans of 3D printed beams generated by ASAP executed in simulation (top row) and real-world (bottom row).

The direct sim-to-real transfer is non-trivial because due to tight millimeter-level clearances in assembly joints for stability and inherent errors in part fabrication and arm localization. After extensive calibrations, in Fig. 5, we show a successful step-by-step correspondence between simulated ASAP plans and real hardware execution. The sim-to-real transfer can be made more robust by incorporating vision or force feedback and adaptive manipulation skills [35], [36].

VI. Limitations and Future Work

We introduced ASAP, an efficient sequence planning algorithm for automated assembly that generates physically-feasible assembly plans for complex-shaped assemblies. We observe three notable failure cases of ASAP: (1) The disassembly tree is completely explored, but no feasible sequence is found, e.g., loosely assembled parts where stability is hard to guarantee or need extra grippers for unstable parts. (2) Physics evaluation can be slow when the contact dynamics in assemblies is complex. (3) Assemblies having strict precedence constraints may only have a few feasible sequences. In this case, ASAP fails to find a solution under a limited budget. The ratios of these failure cases are 50%/45%/5%.

In future work, we plan to explore generating and executing more diverse assembly sequences in the real world using a more flexible robotic setup. To this end, we are interested to integrate different manipulation tools such as suction grippers and screwdrivers, and explore imitation learning or reinforcement learning to train physical assembly skills that are adaptable to real-life assembly tasks with uncertainties.

Acknowledgement

We thank the MIT SuperCloud and Lincoln Laboratory for HPC resources, Jie Xu for discussing the initial idea, Yifei Li for help with experiments, Michael Foshey, Chao Liu and Branden Romero for help on the robotic setup. This work is funded by Autodesk, and in part by NSF CAREER-1846368.

REFERENCES

[1] D. Halperin, J.-C. Latombe, and R. H. Wilson, "A general framework for assembly planning: The motion space approach," *Algorithmica*, vol. 26, no. 3, pp. 577–601, 2000.

[2] S. Sundaram, I. Remmler, and N. M. Amato, "Disassembly sequencing using a motion planning approach," in *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*, vol. 2. IEEE, 2001, pp. 1475–1480.

[3] D. T. Le, J. Cortés, and T. Siméon, "A path planning approach to (dis) assembly sequencing," in *2009 IEEE International Conference on Automation Science and Engineering*. IEEE, 2009, pp. 286–291.

[4] X. Zhang, R. Belfer, P. G. Kry, and E. Vouga, "C-space tunnel discovery for puzzle path planning," *ACM Transactions on Graphics (TOG)*, vol. 39, no. 4, pp. 104–1, 2020.

[5] Y. Tian, J. Xu, Y. Li, J. Luo, S. Sueda, H. Li, K. D. Willis, and W. Matusik, "Assemble them all: Physics-based planning for generalizable assembly by disassembly," *ACM Trans. Graph.*, vol. 41, no. 6, 2022.

[6] S. Abe, T. Murayama, F. Oba, and N. Narutaki, "Stability check and reorientation of subassemblies in assembly planning," in *IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No. 99CH37028)*, vol. 2. IEEE, 1999, pp. 486–491.

[7] J. Aleotti and S. Caselli, "Efficient planning of disassembly sequences in physics-based animation," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2009, pp. 87–92.

[8] I. Rodrıguez, K. Nottensteiner, D. Leidner, M. Kaßecker, F. Stulp, and A. Albu-Schäffer, "Iteratively refined feasibility checks in robotic assembly sequence planning," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1416–1423, 2019.

[9] E. Kongar and S. M. Gupta, "Disassembly sequencing using genetic algorithm," *The International Journal of Advanced Manufacturing Technology*, vol. 30, pp. 497–506, 2006.

[10] R. M. Marian, L. H. Luong, and K. Abhary, "A genetic algorithm for the optimisation of assembly sequences," *Computers & Industrial Engineering*, vol. 50, no. 4, pp. 503–527, 2006.

[11] H.-E. Tseng, C.-C. Chang, S.-C. Lee, and Y.-M. Huang, "A block-based genetic algorithm for disassembly sequence planning," *Expert Systems with Applications*, vol. 96, pp. 492–505, 2018.

[12] H. Wang, Y. Rong, and D. Xiang, "Mechanical assembly planning using ant colony optimization," *Computer-Aided Design*, vol. 47, pp. 59–71, 2014.

[13] Y. Huang, J. Zhang, X. Hu, G. Song, Z. Liu, L. Yu, and L. Liu, "Framefab: Robotic fabrication of frame shapes," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 6, pp. 1–11, 2016.

[14] Y. Huang, C. R. Garrett, I. Ting, S. Parascho, and C. T. Mueller, "Robotic additive construction of bar structures: Unified sequence and motion planning," *Construction Robotics*, vol. 5, pp. 115–130, 2021.

[15] K. D. Willis, P. K. Jayaraman, H. Chu, Y. Tian, Y. Li, D. Grandi, A. Sanghi, L. Tran, J. G. Lambourne, A. Solar-Lezama, and W. Matusik, "Joinable: Learning bottom-up assembly of parametric cad joints," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022.

[16] K. Lupinetti, F. Giannini, M. Monti, and J.-P. Pernot, "Content-based multi-criteria similarity assessment of cad assembly models," *Computers in Industry*, vol. 112, p. 103111, 2019.

[17] S. Rakshit and S. Akella, "The influence of motion paths and assembly sequences on the stability of assemblies," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 2, pp. 615–627, 2014.

[18] S.-K. Kim and M. Likhachev, "Parts assembly planning under uncertainty with simulation-aided physical reasoning," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 4074–4081.

[19] M. Yu, L. Shao, Z. Chen, T. Wu, Q. Fan, K. Mo, and H. Dong, "Roboassembly: Learning generalizable furniture assembly policy in a novel multi-robot contact-rich simulation environment," *arXiv preprint arXiv:2112.10143*, 2021.

[20] J. De Winter, I. El Makrini, G. Van de Perre, A. Nowé, T. Verstraten, and B. Vanderborght, "Autonomous assembly planning of demonstrated skills with reinforcement learning in simulation," *Autonomous Robots*, vol. 45, no. 8, pp. 1097–1110, 2021.

[21] J. Luo and H. Li, "A learning approach to robot-agnostic force-guided high precision assembly," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 2151–2157.

[22] Y. Narang, K. Storey, I. Akinola, M. Macklin, P. Reist, L. Wawrzyniak, Y. Guo, A. Moravanszky, G. State, M. Lu *et al.*, "Factory: Fast contact for robotic assembly," *arXiv preprint arXiv:2205.03532*, 2022.

[23] K. Zakka, A. Zeng, J. Lee, and S. Song, "Form2fit: Learning shape priors for generalizable assembly from disassembly," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 9404–9410.

[24] K. Watanabe and S. Inada, "Search algorithm of the assembly sequence of products by using past learning results," *International Journal of Production Economics*, vol. 226, p. 107615, 2020.

[25] K. Kitz and U. Thomas, "Neural dynamic assembly sequence planning," in *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2021, pp. 2063–2068.

[26] N. Funk, S. Menzenbach, G. Chalvatzaki, and J. Peters, "Graph-based reinforcement learning meets mixed integer programs: An application to 3d robot assembly discovery," 2022.

[27] L. Homem de Mello and A. Sanderson, "A correct and complete algorithm for the generation of mechanical assembly sequences," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 2, pp. 228–240, 1991.

[28] O. Aslan, B. Bolat, B. Bal, T. Tumer, E. Sahin, and S. Kalkan, "Assemblerl: Learning to assemble furniture from their point clouds," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 2748–2753.

[29] Y. Li, K. Mo, L. Shao, M. Sung, and L. Guibas, "Learning 3d part assembly from a single image," *European conference on computer vision (ECCV 2020)*, 2020.

[30] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.

[31] K. Goldberg, B. V. Mirtich, Y. Zhuang, J. Craig, B. R. Carlisle, and J. Canny, "Part pose statistics: Estimators and experiments," *IEEE Transactions on Robotics and Automation*, vol. 15, no. 5, pp. 849–857, 1999.

[32] M. Geilinger, D. Hahn, J. Zehnder, M. Bächer, B. Thomaszewski, and S. Coros, "Add: Analytically differentiable dynamics for multi-body systems with frictional contact," *ACM Transactions on Graphics (TOG)*, vol. 39, no. 6, pp. 1–15, 2020.

[33] J. Xu, T. Chen, L. Zlokapa, M. Foshey, W. Matusik, S. Sueda, and P. Agrawal, "An end-to-end differentiable framework for contact-aware robot design," *Robotics Science Systems*, 2021.

[34] S. Brody, U. Alon, and E. Yahav, "How attentive are graph attention networks?" *arXiv preprint arXiv:2105.14491*, 2021.

[35] S. R. Chhatpar and M. S. Branicky, "Search strategies for peg-in-hole assemblies with position uncertainty," in *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No. 01CH37180)*, vol. 3. IEEE, 2001, pp. 1465–1470.

[36] J. C. Triyonoputro, W. Wan, and K. Harada, "Quickly inserting pegs into uncertain holes using multi-view images and deep network trained on synthetic data," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 5792–5799.