IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2023, London, UK. Cite as T-RO paper.

# Long-Horizon Multi-Robot Rearrangement Planning for Construction Assembly

Valentin N. Hartmann[1,2], Andreas Orthey[2], Danny Driess[2], Ozgur S. Oguz[1,3], Marc Toussaint[2]

***Abstract*— Robotic construction assembly planning aims to find feasible assembly sequences as well as the corresponding robot-paths and can be seen as a special case of task and motion planning (TAMP). As construction assembly can well be parallelized, it is desirable to plan for multiple robots acting concurrently. Solving TAMP instances with many robots and over a long time-horizon is challenging due to coordination constraints, and the difficulty of choosing the right task assignment.**

**We present a planning system which enables parallelization of complex task and motion planning problems by iteratively solving smaller subproblems. Combining optimization methods to jointly solve for manipulation constraints with a sampling-based bi-directional space-time path planner enables us to plan cooperative multi-robot manipulation with unknown arrival-times. Thus, our solver allows for completing subproblems and tasks with differing timescales and synchronizes them effectively. We demonstrate the approach on multiple construction case-studies to show the robustness over long planning horizons and scalability to many objects and agents. Finally, we also demonstrate the execution of the computed plans on two robot arms to showcase the feasibility in the real world.**

***Index Terms*—Manipulation Planning, Task Planning, Robotics and Automation in Construction, Multi-Robot Systems.**

## I. Introduction

AS ROBOTS become ubiquitous in manufacturing and production processes, more robust algorithms to coordinate their work are needed. When multiple robots are employed to achieve a desired goal, two main problems have to be solved: (*i*) assigning tasks to individual robots, and (*ii*) coordinating movements of robots to allow effective execution of those tasks. Combined task and motion planning (TAMP) approaches provide a suitable framework to jointly solve such problems. Scaling these methods to robotic teams consisting of multiple agents and to long-horizon problems remains a major challenge.

We focus on multi-robot planning problems in the context of building construction: As one of the largest industries worldwide, building construction can benefit from autonomous robots and planning processes [1], and the increased efficiency they bring. While robotic construction processes have gained more use in this industry [2], [3], particularly in off- and on-site prefabrication, an integrated autonomous decision-making and robot motion planning approach is missing.

This research has been supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC 2120/1–390831618 "IntCDC".

[1]Machine Learning & Robotics Lab, University of Stuttgart, Germany {firstname}.{lastname}@ipvs.uni-stuttgart.de

[2]Learning and Intelligent Systems Group, TU Berlin, Germany

[3]Department of Computer Engineering, Bilkent University, Turkey

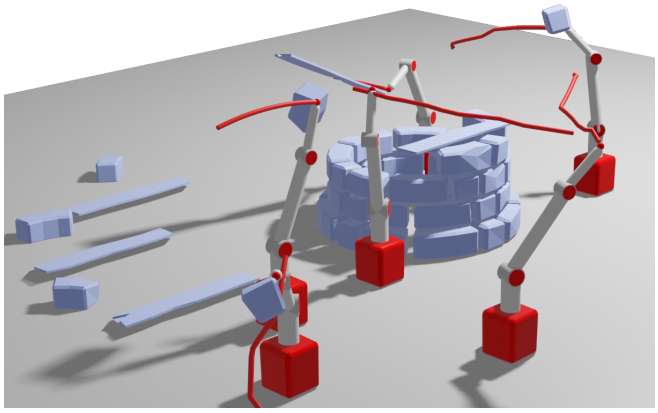


Fig. 1: Six mobile manipulators (red) assembling a well (blue). We visualize the end-effector path of each robot as a red curve.

Previous work on autonomous assembly planning showed promising results in problems such as furniture assembly [4]–[6] or building construction [7]–[9]. Although TAMP formulations are theoretically suitable for such problems, existing approaches do not scale well with an increasing number of robots [7], [8] and/or are only demonstrated on problems spanning short time-horizons [10], [11].

(Construction) Assembly planning can be thought of as rearrangement planning [12], [13] with additional difficulties: *(i)* The ordering of objects is crucial to finish a task, since the feasibility to place a part is highly dependent on previously placed objects [7], [14] (e.g., for stability-reasons, or object availability). *(ii)* Objects and object orderings might impose constraints on the robots (e.g., if an object can not be transported by a single robot or temporary support of the intermediate structure is necessary). These dependencies are amplified when using multiple robots to parallelize the process, due to the non-sequential assembly process.

This paper extends prior work [8], [15] in robotic assembly planning to provide effective coordination of potentially heterogeneous robotic teams in long-horizon settings. We frame the problem as a TAMP problem and use the framework of logic-geometric programming (LGP) [10] to formalize it.

Our algorithm decomposes the overall planning problem by sequentially considering smaller, limited horizon problems with only a subset of robots to enable scalability. These subproblems aim to solve the task sequencing and motion planning while keeping previously planned robot trajectories fixed. A heuristic prioritizes the order in which subproblems are solved. The path planning for one subproblem needs to

account for the previously computed, concurrent motion of other robots. We present a novel bi-directional space-time embedded RRT method that allows planning to an unknown arrival time and enables accounting for previously planned robot-trajectories. The goals for path planning are generated using an optimization-based approach that jointly solves for all manipulation constraints within the subproblem.

Summarizing, our core contribution is a *method to solve time-embedded TAMP problems*, which is enabled by two main novelties:

- a *time-embedded keyframes optimization* that samples time embeddings of manipulation constraints and employs optimization methods to jointly solve for them in the limited horizon subproblems,
- a novel *bi-directional space-time motion planner* that finds paths between keyframes in combined space-time with unknown arrival times, thereby integrating sampling-based path planning with optimization-based methods.

Using these methods we can *decompose* an assembly problem into a series of limited horizon subproblems which only contain a subset of agents while accounting for the constraints implied by previously solved subproblems. This enables us to compute solutions to TAMP problems with multiple robots and long planning horizons.

We demonstrate our algorithm on various construction assembly problems with up to 12 heterogeneous agents, including two long-horizon case-studies using real architectural models where up to 113 parts have to be placed. Our evaluations show that this decomposition-based approach scales well with the number of robots and can efficiently coordinate robotic teams. We also show the feasibility of the plans computed by our algorithm on real robots.

## II. Related Work

Construction robotics [16] is a growing industry which aims to automate building construction. Specialized robots such as bricklaying robots [17] or automated hydraulic excavators [18] recently emerged. Such robots have been used in various constructions settings [19]–[22]. However, the algorithmic aspects of construction planning when dealing with multiple robots have only begun to be studied.

### A. Multi-Robot Motion Planning

On the lowest level, construction planning needs to solve a multi-robot motion planning problem for heterogeneous robot teams [23]. This problem is often divided into two categories: First, one can plan roadmaps in parallel for individual robots, combine those roadmaps into an (implicit) joint state space roadmap and eventually search this roadmap using algorithms such as M* [24], or discrete RRT [25], [26]. Those algorithms can often be significantly improved using heuristics learned from prior experience [27].

Second, prioritization frameworks [14], [28], [29] can be used, where robot-paths are planned sequentially, imposing the previously planned movements as constraints for the next robot. Planning sequentially can require backtracking, which can be time consuming. Several approaches exist to avoid backtracking, e.g., analyzing start or goal conflicts [30].

In our approach, we use a prioritization approach by employing a heuristic to prioritize exploration of LGP subproblems. Contrary to prioritization in multi-robot motion planning we embed planning in space-time [31], [32] to enable planning in a dynamic environment and with unknown arrival times. Space-time planning has previously been investigated in the context of rendezvous planning [33]. In assembly tasks, it is often necessary to sample several keyframes at different time instances, due to blockages of the goal region from other agents. We combine space-time planning explicitly with discrete constraint switches. Our novel bidirectional space-time RRT allows us to efficiently find time-dependent plans while avoiding collisions with previously planned robots.

### B. Assembly Planning

The algorithmic aspects of construction planning are studied in the field of assembly or rearrangement planning [34], [35]. Assembly planning traditionally focuses on finding valid sequences to assemble objects [36], [37]. More recently, this was also applied to masonry constructions [38], [39]. Assembly sequence planning approaches have been scaled to large number of parts: In [40] an assembly planning algorithm for a large number of bricks that are assembled in a plane is introduced, but the approach, in contrast to our work, does not take the path-planning problem into account. In [41] a method that computes assembly plans for a large numbers of objects is presented, but the approach is not general purpose, i.e., it is only applicable to grid-like structures, and the specific robot-building material pairing of TERMES [42]. A review on collective robotic construction can be found in [43].

A review of solution approaches to TAMP problems can be found in [15]. In this review, we focus on integrated task and motion planning applied to robotic assembly planning. Constraints and their intersections are explicitly enumerated in the state space [44]. Such a constraint graph can be exploited by biased sampling at constraint intersections [45], [46], and these samples can be connected along the constraint manifolds using projection methods [47]–[49]. Complex applications of such an approach are demonstrated in [25], such as concurrent handovers between multiple robots with multiple objects and capacity constraints.

On the other hand, symbolic assembly approaches explicitly introduce symbolic states to find task-level decisions and to factorize the problem [10], [50], [51]. Once symbolic decision sequences (skeletons) are found, lower level planners are used to execute a skeleton, using sampling-based [52], [53] or optimization-based [10] methods. This approach can be tailored towards many different applications, for example by including force constraints [51], dealing with re-planning [54], [55] or handling partial observability [56].

Previous work dealt with long-horizon construction planning for a single agent [57], or two agents [8], but solving long-horizon TAMP problems using multiple robots is an open challenge. Initial steps towards solving multi-robot, multi-object rearrangement tasks in a simple configuration space for

homogeneous robot teams with few objects were made in [11], [58]. Such previous work assumes that the actions of robots are synchronized [8], [10], [11], [25], plan in the combined space of all robots, and do therefore not scale [8], are only demonstrated for simple robots and few objects and state that they do not expect to scale [58], or are not demonstrated on long time-horizons [11], [25].

With our approach, we are able to plan for heterogeneous robot teams with complex interactions, and to scale to more objects and robots. We achieve this by combining sampling based methods for path-planning, and optimization-based methods for finding the mode-switches. Contrary to previous work, our approach does also not assume synchronicity of actions, thereby allowing the parallelization of assembly tasks efficiently.

## III. Multi Robot Rearrangement Planning Notation and Problem Formulation:

Given $n$ unique objects, indexed by $o \in \mathcal{O}$, $|\mathcal{O}| = n$, with initial poses $p_o^0 \in SE(3)$ at time $t = 0$, and $m$ robots, indexed by $r \in \mathcal{R}$, $|\mathcal{R}| = m$, the aim is to *rearrange* all objects to their (given) goal locations $p_o^G \in SE(3)$. Each robot may have its own configuration space $\mathcal{Q}_r \subset \mathbb{R}^{d_r}$.

We formulate the problem as a non-linear mathematical program over the path $x : [0, T] \to \mathcal{X}$. The configuration space $\mathcal{X} = \mathcal{Q} \times SE(3)^n$ consists of all robot configuration spaces $\mathcal{Q} = \mathcal{Q}_{r_1} \times \cdots \times \mathcal{Q}_{r_m}$ and object configuration spaces.

Over time, different constraints on the path are active, e.g., at the end of a *pick*-task, the end-effector of an agent needs to fulfill gripping-constraints. Which constraints are active is determined by the task assignment $s \in \mathcal{S} = \mathcal{S}_{r_1} \times \cdots \times \mathcal{S}_{r_m}$, where $\mathcal{S}_r$ indicates the feasible tasks for robot $r$. Thus, the state $s$ determines the current task assignment of each robot[1]. We use $s_{r,1:K_r} \in \mathbb{S}(\mathcal{R}, \mathcal{O})$ to denote the discrete sequence of tasks of robot $r$, with $s_{r,j} \in \mathcal{S}_r$. $K_r$ is the number of discrete states for robot $r$ in the sequence $s_{r,1:K_r}$. The set $\mathbb{S}(\mathcal{R}, \mathcal{O})$ denotes all valid state sequences induced by a first-order logic-language for the robots $\mathcal{R}$ and objects $\mathcal{O}$. For example, a *handover*-task necessitates a *pick*-task as precondition.

In most approaches to solve TAMP problems the transitions between task assignments occur at fixed intervals [8], [10], [11], [25]. Our problem formulation allows for task assignments to switch (finitely often) at any time. This is achieved by the *scheduling function*, $k : [0, T] \to \mathcal{N} = (1, \dots, K_1) \times \cdots \times (1, \dots, K_m)$, that maps continuous time $t$ into a vector of indices that select the currently active task assignments for all robots, such that $s(t) = s_{k(t)}$. We use $k_r(t)$ to denote the scheduling function for robot $r$. The scheduling function is constrained to respect the order of the indices, i.e., $k_r(t_1) \leq k_r(t_2)\ \forall t_1 \leq t_2$.

Therefore, we try to find the path $x$, the terminal time $T > 0$, the scheduling function $k$, and the sequences of discrete states $\{s_{r,1:K_r}\}_{r=1}^m$ to optimize

$$\min_{\substack{x,T,k,\\ \{s_{r,1:K_r}\}_{r=1}^m}} \int_0^T c(x(t), \dot{x}(t), \ddot{x}(t))\, \mathrm{d}t \tag{1a}$$

$$\text{s.t.} \quad x(0) = x_0 \tag{1b}$$

$$\forall t \in [0, T]: \quad g(x(t), \dot{x}(t), s_{k(t)}) \leq 0 \tag{1c}$$

$$\{s_{r,1:K_r}\}_{r=1}^m \in \mathbb{S}(\mathcal{R}, \mathcal{O}) \tag{1d}$$

$$g_{\text{goal}}\left(x(T), p_{\mathcal{O}}^G, \mathcal{O}\right) \leq 0. \tag{1e}$$

The task assignment state $s_{r,k_r(t)} \in \mathcal{S}_r$ determines currently active constraints for each robot $r$ on the path $x$ at time $t$ via the constraint function $g$ in (1c). For example, $s_{r,k_r(t_1)} = s_{r,j}$ could specify the necessary constraints for robot $r$ such that it grasps an object at time $t_1$ as the $j$-th discrete state $s_{r,j}$ of the sequence $s_{r,1:K_r}$. Additionally, (1c) could describe collision constrains, or joint-limits. In the following, we refer to the discontinuities in $s$ as *mode-switches*, or *keyframes*. Figure 2 illustrates the components of $s$, and how the sequence is mapped to continuous time.

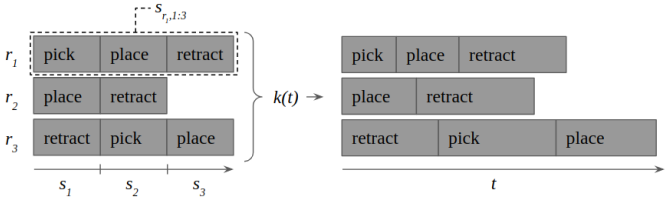



Fig. 2: Illustration of the scheduling function $k$ on a problem with three agents $(r_1, r_2, r_3)$. The scheduling function maps the time $t$ to the active tasks $s_1, s_2, s_3$.

The goal constraint (1e) specifies that at the end, all objects have to be at their target poses. The initial condition $x_0$ contains the initial configuration of all robots as well as the initial poses of the objects. Finally, $c$ is a cost function, such as path length, control cost, or minimal time. In case we are only interested in finding a *feasible* solution, $c = 0$.

### A. Assumptions

We summarize the assumptions we made in the problem formulation:

- Known initial and final position of all objects, and availability of a method to sample configurations for manipulating them.
- Monotonic rearrangement: While it is possible to handle nonmonotonicity in the logic search, we assume in this work that each object is handled a single time. However, we consider re-grasping of objects such as handovers.
- No force and torque constraints for the robots: In this work, we assume that the parts are light compared to the allowable robot-payload. Consequently, every object can be manipulated by a single robot under this assumption.

## IV. Method

Solving the problem described in Eq. (1) in a fully joint and global manner is intractable, and even finding a feasible solution that utilizes all robots is hard. We present our approach

[1]This is slightly different to most TAMP literature, where $s$ is a set of grounded literals that determine, e.g., which robot is assigned to which task, and STRIPS-like rules determine feasible transitions between logical states (task assignments).

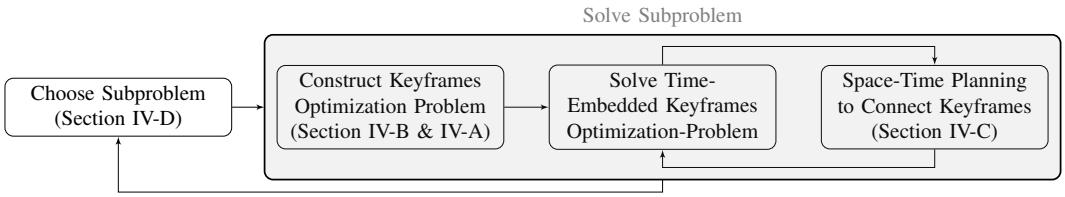



Fig. 3: A high-level description of the steps of our method. The subproblem to solve (i.e. with which agent which part should be placed) is chosen using a heuristic. This subproblem is then solved (gray rectangle) by first finding a set of feasible keyframes that fulfill the constraints for the mode switches. Since other agents might already move on a previously planned trajectory, the method then finds a time-embedded path by repeatedly generating new keyframes with different time-embeddings and attempting to connect the keyframes to each other. Please refer to Section IV for a more elaborate description of each step.

**Algorithm 1:** `plan()`.

tree $\leftarrow \emptyset$, curr_node $\leftarrow \emptyset$
**while** true **do**
  $\mathcal{O} \leftarrow$ `extract_objects`(curr_node)
  $\text{seq}_{\text{prev}}, R_{\text{prev}} \leftarrow$ `extract_prev_attempts`(curr_node)
  $(o, R, \text{seq}) \leftarrow$ `choose_subproblem`$(\mathcal{O}, \text{seq}_{\text{prev}}, R_{\text{prev}})$
  **if** $(o, R, \text{seq}) = \emptyset$
    curr_node $\leftarrow$ `backtrack`(tree)
    **continue**
  $\text{sol}_{\text{sub}} \leftarrow$ `solve_subproblem`$((o, R, \text{seq}))$
  **if** $\text{sol}_{\text{sub}} = \emptyset$
    `mark_as_infeasible`(curr_node, $(o, R, \text{seq})$)
  **else**
    curr_node $\leftarrow$ `add_to_sol`(tree, $\text{sol}_{\text{sub}}$)
  **if** done
    **return** `extract_sol`(curr_node)
**return** Infeasible

**Algorithm 2:** `solve_subproblem`$((o, R, \text{seq}))$.

**for** $k = 1 \dots |\text{seq}|$ **do**
  `goal_sampler` $\leftarrow$ `make_goal_sampler`(seq, $R, \{x_1, \dots, x_k\}$)
  $q_0, t_0 \leftarrow$ `get_available_time`$(R)$
  $x_k \leftarrow$ `ST-RRT*`$((q_0, t_0)$, `goal_sampler`$)$
  **if** $x_k = \emptyset$
    **return** $\emptyset$
**for** $k = 1 \dots |\text{seq}|$ **do**
  $x_k \leftarrow$ `shortcut`$(x_k)$
  $x_k \leftarrow$ `smooth`$(x_k)$
**return** $\{x_1, \dots, x_{|\text{seq}|}\}$

**Algorithm 3:** `ST-RRT`$(x_0$, `goal_sampler`$)$.

$T_a \leftarrow x_0,\ T_b \leftarrow \emptyset$
**while** not stopped **do**
  $t_{\text{lb}}, t_{\text{ub}} \leftarrow$`update_bounds`()
  **if** $\text{rnd}(0, 1) < p_{\text{goal}}$
    $t \leftarrow$ `sample`$(t_{\text{lb}}, t_{\text{ub}})$
    $q_g \leftarrow$ `goal_sampler`$(t)$
    `add_goal`$((t, q_g))$
  $q \leftarrow$ `sample_valid_state`()
  $t \leftarrow$ `sample_valid_time`$(q)$
  **if not** $x_{\text{new}} \leftarrow$ `extend`$((t, q), T_a)$ = trapped
    **if** `connect`$(x_{\text{new}}, T_b)$ = reached
      **return** `extract_path`()
  `swap`$(T_a, T_b)$
**return** $\emptyset$

to decompose the problem into simpler subproblems, and to solve the subproblems such that the solutions together are a feasible solution to the original problem. Algorithms 1 to 3 and the following sections describe

1) A decomposition of the overall problem into subproblems that each contain only a subset of robots and assigned tasks. These subproblems account for the time-embedding in a scene where other robots are already moving, and represent coordination constraints for their respective subset of robots to ensure feasible cooperative manipulations, e.g., handovers.
2) An approach to generate solutions to manipulation constraints defined by the subproblems, such as pick, place, or handover constraints. We use this method to sample goals for the path planning algorithm.
3) A bi-directional space-time RRT path planner to find feasible motions between keyframes taking moving robots into account.
4) A heuristic to prioritize the order of subproblems that the overall system tries to solve, and how everything is integrated with each other.

An illustration of this system can be seen in Fig. 3.

### *A. Decomposition into Time-Embedded, Limited Horizon Subproblems with a Subset of Agents*

A natural decomposition of Eq. (1) into smaller subproblems emerges from the problem specification of rearranging objects, i.e., we consider subgoals of rearranging *one* object with potentially multiple robots. The following description focuses on clarifying the degrees-of-freedom (DoF) for each subproblem.

Assume that we are in step $l$ of the planning process. The set $\mathcal{O}^{l-1} \subseteq \mathcal{O}$ denotes all objects that have been successfully moved to their respective goal locations at previous planning steps, i.e., a task sequence and corresponding trajectory has been planned. The set $\bar{\mathcal{O}}^{l-1} = \mathcal{O} \backslash \mathcal{O}^{l-1}$ denotes the objects that have no plan associated yet. A heuristic (explained in Section IV-D) selects a single new object $o_l \in \bar{\mathcal{O}}^{l-1}$ and a set of robots $R_l \subseteq \mathcal{R}$ that should be involved in rearranging the object $o_l$ to its target pose $p^G_{o_l}$.

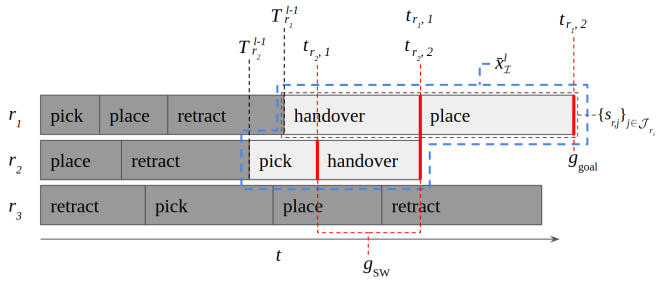



Fig. 4: A problem with three agents, in which we plan for $r_1$, and $r_2$ (and object $o$) with a *handover* sequence. The previously fixed plans are dark grey, the red lines indicates constraints $(g_{\text{sw}}, g_{\text{goal}})$ that have to be fulfilled at the mode-switches, and the tasks and corresponding paths that have to be planned are light-grey. The blue box indicates the DoF $\bar{x}_{\mathcal{I}_l}$ for the current planning problem. The active indices $\mathcal{I}$ are $\{r_1, r_2, o\}$.

The optimization problem we solve in step $l$ therefore only optimizes over a part of the path $x$. We use $\bar{x}^l_{\mathcal{I}_l}$ to denote the degrees of freedom in the subproblem, where $\mathcal{I}_l = \{R_l, o_l\}$ is the set of indices of the path $x$ that correspond to the robots $R_l$ and the object $o_l$. Not all degrees of freedom in $\bar{x}_{\mathcal{I}_l}$ necessarily become active at the same time, since some of the robots might be involved in previously planned motions up to different times (Fig. 4: $r_1$ and $r_2$ become active at different times). Similarly, the subproblem in step $l$ is temporally embedded into a scene where robots and objects *not* part of the current planning problem may follow previously computed plans ($r_3$ in Fig. 4).

To define the optimization problem for the subproblem, we therefore also need to specify how the inactive indices of $x$ are defined. In order to do so, let $T^{l-1}_r, T^{l-1}_o \in \mathbb{R}$ denote the time until which paths for robot $r$ and object $o$ have been planned in the $l-1$ previous planning steps. If no path has yet been planned for a robot/object, its time is set to zero. This allows us to define the $i$-th component of the path variable $x^l$ in the planning step $l$ at time $t$ as

$$x^l_i(t) = \begin{cases} x^{l-1}_i(t) & t \leq T^{l-1}_i \\ x^{l-1}_i\left(T^{l-1}_i\right) & t > T^{l-1}_i,\ i \notin \mathcal{I}_l \\ \bar{x}^l_i(t) & t > T^{l-1}_i,\ i \in \mathcal{I}_l \end{cases}. \tag{2}$$

Therefore, the degrees-of-freedom $\bar{x}^l_{\mathcal{I}_l}$ in planning step $l$ are those of $R_l$ and $o_l$ from the point in time where they have no associated planned trajectory yet (i.e., $t > T^{l-1}_i,\ i \in \mathcal{I}_l$). In the other cases, they move according to previously computed plans ($t \leq T^{l-1}_i$) or remain at the last planned configuration ($t > T^{l-1}_i,\ i \notin \mathcal{I}_l$), i.e., they correspond to inactive degrees of freedom. The same holds for the scheduling function $k$ which has the effective degrees of freedom $\bar{k}_{\mathcal{I}_l}$. Similarly, the search over the symbolic task state for the selected robots happens over $\left\{s_{r,K^{l-1}_r+1:K^l_r}\right\}_{r\in R_l}$ only. The complete task state sequence of robot $r$ is then the concatenation $s_{r,1:K^l_r} = \left(s_{r,1}, \ldots, s_{r,K^{l-1}_r}, s_{r,K^{l-1}_r+1}, \ldots, s_{r,K^l_r}\right)$ of the sequences that have been determined in the steps up until step $l-1$ and the new sequence. We show an illustration that serves to explain the free and fixed parts respectively in planning step $l$ in Fig. 4.

This leads to the following limited horizon optimization problem in step $l$ for the chosen object $o_l$ and robots $R_l$

$$\min_{\substack{\bar{T}^l,\ \bar{x}^l_{\mathcal{I}_l}(\cdot),\ \bar{k}^l_{\mathcal{I}_l}(\cdot) \\ \left\{s_{r,K^{l-1}_r+1:K^l_r}\right\}_{r\in R_l}}} \int_{\bar{T}^{l-1}}^{\bar{T}^l} c\left(x^l(t), \dot{x}^l(t), \ddot{x}^l(t)\right) \mathrm{d}t \tag{3a}$$

$$\text{s.t.} \quad x^l \text{ as defined in Eq. (2)}$$

$$\forall t \in \left[\bar{T}^{l-1}, \bar{T}^l\right]: \quad g\left(x^l(t), \dot{x}^l(t), s^l_{k(t)}\right) \leq 0 \tag{3b}$$

$$\forall r \in R_l: \quad s_{r,1:K^l_r} \in \mathbb{S}(R_l, \{o_l\}) \tag{3c}$$

$$g_{\text{goal}}\left(x^l(\bar{T}^l), p^G_{o_l}, \{o_l\}\right) \leq 0. \tag{3d}$$

Here, $\bar{T}^{l-1} = \min_{r\in R_l} T^{l-1}_r$ is the earliest time for which no plan of a robot in $R_l$ exists yet. If $R_l$ contains more than one robot, the final time $\bar{T}^l$ that is being optimized for is the maximum time of all robots $R_l$ that are involved in the current planning step, as one robot could fulfill all its constraints earlier than the others. Consequently, the final times $T^l_j \leq \bar{T}^l$ are assigned by extracting the minimum times where each individual robot and object $j \in \mathcal{I}_l$ fulfill their constraints.

### *B. Time-Embedded Keyframes Optimization to Jointly Solve for Sequential Transition Constraints*

Equation (3) is nonconvex due to collision avoidance, manipulation constraints, the time-embedding, and the discrete task-sequence. To robustly find *feasible* solutions to Eq. (3), we combine a search for a valid sequence with nonlinear optimization and a sampling-based planner. The optimizer solves for configurations at the transition between two task assignment states, while the motion planner, described in Section IV-C, iteratively finds paths between the keyframes.

Assume we are given robots $R$ and the sequences of discrete states $\left\{s^l_{r,j}\right\}_{j\in\mathcal{J}^l_r}\ \forall r \in R$ with $\mathcal{J}^l_r = \left\{K^{l-1}_r + 1, \ldots, K^l_r\right\}$. The problem

$$\min_{x^l_{\mathcal{I}_l}(\cdot)} \sum_{\substack{r\in R_l, \\ j\in\mathcal{J}_r}} c_d(x^l_{\mathcal{I}_l}(t_{r,j})) \tag{4a}$$

$$\text{s.t.} \quad \forall r \in R_l\ \forall j \in \mathcal{J}_r: \quad g_{\text{sw}}(x^l_{\mathcal{I}_l}(t_{r,j}), s_{r,j}, s_{r,j-1}) \leq 0 \tag{4b}$$

$$\exists r \in R_l \exists j \in \mathcal{J}_r: \quad g_{\text{goal}}(x^l_{\mathcal{I}_l}(t_{r,j}), p^G_{o_l}, o_l) \leq 0 \tag{4c}$$

thus describes the configurations of the involved robots and the object $o_l$ at the mode switching times $t_{r,j}$. The constraint (4b) is the discrete version of (3b) at the transition from $s_{r,j-1}$ to $s_{r,j}$. The cost function $c_d$ is the discrete version of $c$. Solving Eq. (4) (using, e.g., [59]) generates a set of keyframes that jointly fulfill interdependent constraints, e.g., finding consistent hand positions in pick and place poses.

To solve Eq. (4), we first sample the times $t_{r,j}$ where the transition from the discrete state $s_{r,j-1}$ to $s_{r,j}$ occurs uniformly between $t_{\text{lb},j}$ and $t_{\text{ub},j}$ with $t_{r,j-1} < t_{r,j}$ and $\bar{T}^{l-1} < t_{r,j}$ for $j \in \mathcal{J}_r$. The lower bounds of the intervals are estimated by using the minimum possible arrival time and the upper bound is a fixed multiple of it. The intervals are gradually enlarged during path-planning if no solution

can be found. We then use an optimization based solver to find configurations fulfilling the constraints. Optimization-based solvers are strong to resolve equality constraints, but are prone to local optima. We alleviate this problem by *repeatedly* solving Eq. (4) to generate various consistent keyframes. There are two reasons for why the solution to Eq. (4) is randomized and generates varieties of solutions: First, whenever we solve Eq. (4), we sample the times $t_{r,j}$, which leads to a different time embedding and corresponding constraints. Second, we initialize the optimizer with randomly sampled configurations, which helps to find various local optima.

In this view, solving Eq. (4) generates feasible keyframes that can be used as goals for bi-directional path planning.

### C. Bi-directional Space-Time Path Planning to Connect Keyframes

We compute the path for a given task sequence in a sequential manner (Fig. 5): the path planner first aims to find a path between the first two keyframes; when one is found, it moves on to find a path to a third keyframe that is consistent with the first two. It can always query the keyframes optimizer for more keyframes consistent with given previous keyframes, i.e., Eq. (4) is solved for the *remaining* mode-switch configurations. Solving the full remaining problem, instead of only the constraints implied by the next mode-switch, excludes keyframes that are feasible in the next mode switch, but lead to an infeasible problem later on, e.g., a *pick* configuration, which does not correspond to a feasible *place*-configuration.

Since the arrival time at which the robot can reach a goal keyframe configuration is unknown, we uniformly sample a range of candidate time-embeddings $t_{r,j}$ as input to the keyframes optimizer. During the planning process, the path planner continuously extends the range of time-embedding samples to allow for consideration of larger time-spans, i.e., to enable 'waiting'. The arrival times that are found in this fashion correspond to the scheduling function $k$ for the robots involved in the current subproblem.

*Path-Planning:* We finally present the bi-directional path planner to connect the keyframe-configurations. Following the standard notation for time-embedded path planning [32], [33], let $\mathcal{Q}_{R_l}$ denote the configuration space for the robots $R_l$, and $\mathcal{T} \subset \mathbb{R}_{\geq 0}$ the time dimension. Our path planning problem is the problem of finding a collision-free path through the combined space-time $\mathcal{Y} = \mathcal{Q}_{R_l} \times \mathcal{T}$ from an initial keyframe configuration $(x(t_{r,j-1}), t_{r,j-1})$ at time $t_{r,j-1}$, to a *set* of candidate goal keyframe configurations, each a pair $(x(t_{r,j}), t_{r,j})$ in space-time with varying $t_{r,j}$.

The free configuration space $\mathcal{Q}_{\text{free}}$ for the robots $R_l$ is time-dependent, as objects and other agents might move on previously planned paths. Additionally, it can be the case that some of the agents we are currently planning for move on a fixed path for some time, i.e., they only become a degree of freedom at time $t > \bar{T}^{l-1}$. We deal with this by using a constrained path-planner [60], with the constraints defined in Eq. (2).

Specific care has to be taken due to the time-dimension, and the direction-dependent distance function in the configuration space:

$$d(y_1, y_2) = \begin{cases} \lambda d_{\mathcal{Q}_{R_l}}(q_1, q_2) + (1-\lambda)(t_2 - t_1), \\ \qquad\qquad \text{if } t_1 < t_2,\ v \leq v_{\max}; \\ \infty, \qquad\qquad \text{else.} \end{cases} \tag{5}$$

where $d_{\mathcal{Q}_{R_l}}$ can be any valid metric in $\mathcal{Q}_{R_l}$, and $v$ is an estimate of the velocity. We use $\lambda \in (0, 1]$ to describe the importance of the path-length and the needed time respectively. This distance-metric encodes the inability to move from a configuration $q_1$ at time $t_1$ to a configuration $q_2$ at time $t_2$ if either the required speed $v$ is too high, or the robot would need to move backwards in time.

We then extend bi-directional rapidly-exploring random trees (RRT) [61] to space-time[2]: This extension consists of the previously described keyframe sampler which generates the goals, time dependent collision queries, and configuration-sampling bounded by the sampled keyframe time. Specific care has to be taken when connecting edges using the goal-centered tree: we move 'backwards' in this case, and thus the distance function has to be adapted. This bi-directional space-time RRT formulation allows us to efficiently find paths.

After finding a path using the outlined approach, we post-process the path by shortcutting [63] the obtained path, and smoothing it. Shortcutting of the path works by repeatedly choosing two discrete states of the path and checking if they can be connected with a straight line while fulfilling the constraints, e.g., collision, kinematics, or velocity limits. If that is the case, the straight line path replaces the part of the path between the chosen discrete states.

For smoothing, we use an optimizer [64] that takes the constraints and the cost function of the original problem as input and is thus able to take the dynamics and constraints into account. Taking this approach is similar to separating path planning and trajectory planning, and is a common approach to find feasible paths in cluttered environments [65], [66].

As in the planning itself, care has to be taken to shortcut and smooth in a constrained manner, i.e., parts of the path that have previously been fixed can not be altered.

### D. Prioritization of Subproblems and Search over Skeletons

In each step $l$, the limited horizon formulation Eq. (3) requires a selection of object $o_l$ and robots $R_l$. Due to the assembly tasks we consider, the order in which objects have to be rearranged (in particular being placed) is not obvious. For example, objects need sufficient support in order to be placed, while placing some objects too early might obstruct later object placements.

In planning step $l$ we have the set $\mathcal{O}^{l-1} \subseteq \mathcal{O}$ of previously rearranged objects, and the set of objects $\bar{\mathcal{O}}^{l-1} = \mathcal{O} \backslash \mathcal{O}^{l-1}$ that have no plan associated yet. From previous task assignments and trajectory planning, the times $T_r^{l-1}$ until which a trajectory is already assigned to robot $r$ are known. Based on this planning state, the algorithm has to

(i) choose an object $o_l \in \bar{\mathcal{O}}^{l-1}$ subject to object order constraints (expressed by $\phi(o_l; \mathcal{O}^{l-1}) \leq 0$),

[2]In this work, we used an early version of the algorithm presented in [62].

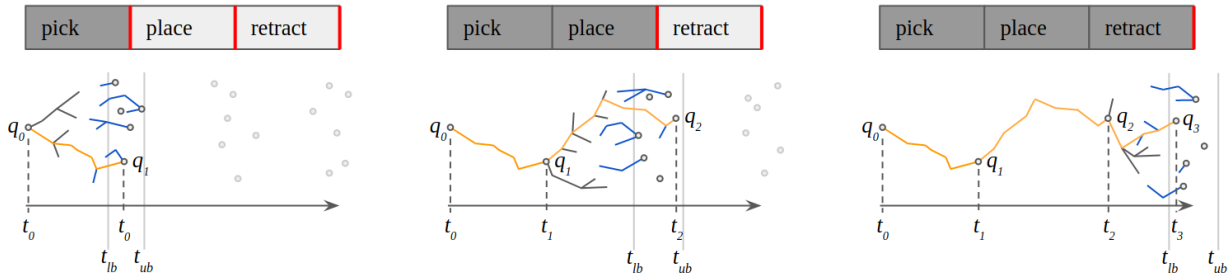
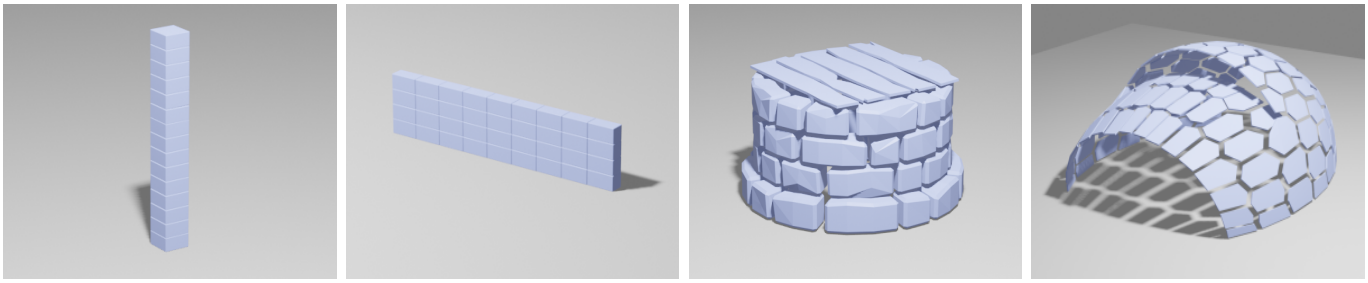



Fig. 5: Illustration of how plans are computed sequentially for a task sequence using the keyframe sampler and the bi-directional space-time planner. In the task sequence, red shows the constraints that are part of the optimization problem to generate the keyframes, dark grey indicates the tasks for which a path is fixed. In the path planning, the nodes are the keyframes that are computed, where the light grey ones are the keyframes that are computed for the later mode-switches. The edges of the forward tree are dark grey, and the ones of the reverse tree are blue. The final path for a task is orange. Please refer to Section IV-C for an in-depth explanation.



(a) Tower (b) Wall (c) Well (d) Pavilion

Fig. 6: Final configurations of the models we use for the experiments and demonstrations.

(ii) choose a subset of robots $R_l \subseteq \mathcal{R}$ to rearrange object $o_l$,
(iii) search through the space of assignment sequences $\left\{s_{r,K_r^{l-1}+1:K_r^l}\right\}_{r\in R_l}$ to find a sequence that is logically feasible, leads to the goal (i.e., object placement), and for which the keyframes optimizer and path planner can find solutions. If no such sequence is found, choose another robot assignment $R_l$ (that was not chosen yet).
(iv) If no possible choice of $R_l$ leads to a feasible subproblem (Eq. (3)), we backtrack to rewind previous object placements, and attempt to place $o_l$. Backtracking is repeated until a valid solution to place $o_l$ is found.

This can be seen as a depth-first tree-search over the objects, robots, and assignment sequences. These steps are explained in more detail in the following:

*1) and 2) Selection of Object and Robots:* The objects $o_l$ in step 1) and robots $R_l$ in step 2) are selected by a strict prioritization: We prioritize the selection of objects by minimizing a heuristic $h$:

$$h(o_l; \mathcal{O}^{l-1}) \quad \text{s.t.} \quad \phi(o_l; \mathcal{O}^{l-1}) \leq 0, \tag{6}$$

where the constraint $\phi(o; \mathcal{O}^{l-1}) \leq 0$ is application specific. As example, it could express that intermediate construction states need to be stable, or put limits on deflections of objects. Objects violating the constraint are excluded from the prioritized search.

The selection of robots $R_l$ is prioritized by minimizing

$$\max_{r\in R_l} T_r^{l-1}, \tag{7}$$

i.e., the latest busy time of all involved robots. This assumes (very conservatively) that the work on this subproblem starts only when the last of the involved robots becomes free. The robot-selection could also be prioritized by, e.g., minimizing the estimated finishing time of the subproblem.

*3) Path Planning and Task Sequence/Robot Rejection:* Finding a task sequence is realized as a breadth-first search to check if there exists a sequence

$$\left\{s_{r,K_r^{l-1}+1:K_r^l}\right\}_{r\in R_l} \in \mathbb{S}(R_l, \{o_l\}) \tag{8}$$

which is logically feasible and leads to the symbolic goal.

With Eq. (3) fully defined by the choice of $R_l$, $o_l$, and $\left\{s_{r,K_r^{l-1}+1:K_r^l}\right\}_{r\in R_l}$, we try to find a valid path (as detailed in Section IV-B and Section IV-C). Before attempting to solve the full problem, it is possible to evaluate *lower bounds*, i.e., simpler subproblems, which, if they are infeasible, guarantee that there is no solution to the full problem. In our case, examples for this are *i)* attempting to find a placement pose, and *ii)* attempting to find the configurations at the mode switches. For a more thorough description of the notion of lower bounds, we refer to [67].

If a problem is infeasible for a chosen $\left\{s_{r,K_r^{l-1}+1:K_r^l}\right\}_{r\in R_l}$, we first attempt to solve the problem using different $\left\{s_{r,K_r^{l-1}+1:K_r^l}\right\}_{r\in R_l}$ multiple times, and if still no feasible solution can be found, we restart from *2)*, excluding infeasible task sequences and robots.

In general, the methods we use for generating keyframes or motion paths can not prove infeasibility of a specific discrete

assignment $R_l$, $o_l$, $\left\{s_{r,K_r^{l-1}+1:K_r^l}\right\}_{r\in R_l}$ due to non-convexity of the optimization problem, or due to finite runtime of the path planning. Hence, we keep a list of all assignments that we determined to be infeasible before, and revisit them in a deprioritized manner if still no solutions in the remaining discrete assignments can be found. This allows us to explore more promising decisions, while still guaranteeing that a solution will be found eventually, if it exists. For brevity, this was left out of the algorithm.

*4) Backtracking:* At this stage, all robots $R_l$, and possible task assignments were checked. Thus, any infeasibility at this stage must be caused by previously placed objects, since a selected part $o_l$ fulfills all the constraints to be able to be placed, i.e., $\phi(o_l; \mathcal{O}^{l-1}) \leq 0$, and placing additional objects can never make placement of $o_l$ feasible. Instead, we backtrack to rewind previously placed objects until a valid solution to place object $o_l$ can be found.

## V. Demonstrations & Results

We analyze the scalability of the algorithm, and how some scenarios benefit more than others from better parallelizability. We demonstrate the robustness of the algorithm on long-horizon scenarios, show the ability to coordinate multiple robots in a scenario where a handover sequence is necessary, and a real robot experiment. Finally, we compare the algorithm to a modified version of a classical TAMP-solver.

### A. Setup

We test the algorithm on several construction scenarios[3][4]:

- A tower, consisting of 15 pieces, where the placements of the parts have to be in strictly sequential order.
- A wall, consisting of 36 bricks used to analyze how well a task can be parallelized.
- A well, consisting of 52 pieces, used to demonstrate scalability.
- A pavilion consisting of 113 unique wooden cassettes, used to demonstrate scalability.

The final configuration of the models is visualized in Fig. 6. We assume that the pieces form a rigid body with the neighbouring pieces as soon as they are placed, and thus neglect both the fastening process, and the structural support that would be necessary. We are first and foremost interested in finding a *feasible* solution to Eq. (1), i.e. we use $c = 0$. For the real robot experiment, we minimize the acceleration in the smoothing-step.

*1) Robots:* We demonstrate our approach using 3 different robots (Fig. 7): a mobile manipulator, a KUKA-arm on a mobile base, and a crane. The robots we are using for the demonstrations are holonomic. If not stated differently, we model manipulation by gripping-by-touch: On construction sites, vacuum grippers are commonly used, which can be approximated as gripping-by-touch.

[3]Simulated using https://github.com/MarcToussaint/rai

[4]The experiments were run on a single core of Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz with 16GB RAM using Ubuntu 18.04.

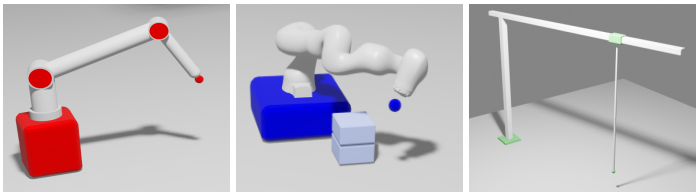


Fig. 7: An illustration of the robots (from left to right): a mobile manipulator, a KUKA-arm on a mobile base, and a crane.

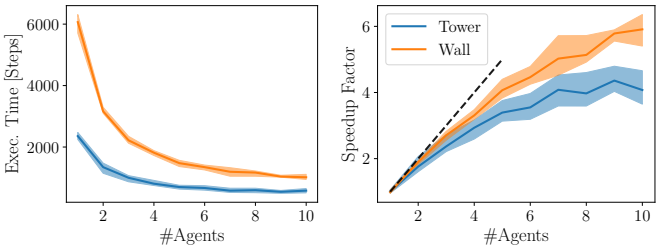



Fig. 8: The execution time (left), and the speedup factor of the execution time (right) when using multiple agents for the tower and the wall models over 10 runs. The speedup factor is the factor by which the execution time is sped up by using $m$ agents compared to only one agent. The line is the median and the shaded area is bounded by the 25 and 75 percentile.

*2) Task Assignments:* The discrete task assignments are *pick*, *place*, *retract* and *handover*. We require a *place* task to always be followed by a *retract* task, since in general, after placing an object, it is not desirable to stay in the same configuration, and possibly block the placement configurations of other agents.

*3) Ordering Heuristic:* We represent the buildings as a graph, with the nodes being parts, and edges between connected parts. The heuristic $h$ in Eq. (6) is chosen to find the object that maximizes the number of previously placed neighbours. The placeable parts are the set of nodes which are connected to at least one node in the graph that is already placed. This is encoded in the constraint $\phi$ in Eq. (6).

Due to the heuristic and placement constraint, no backtracking was required in our experiments. As such, there is no specific demonstration, or mention of how many times backtracking was needed in the examples.

### B. Experimental Results

We provide analysis on several quantitative metrics in this section. The experiments for the analysis were done using the mobile manipulator utilizing *pick and place*-sequences. Videos of the assembly processes for various models with different team-sizes and various constellations of robot types can be found in the supplementary material.

*1) Execution Time:* The execution time, i.e. the real time of the planned movements, is expected to decrease the more agents are used for the rearrangement task. Figure 8 plots the factor by which the execution is sped up against the number of agents for the tower and the wall, and highlights how the *separability* of the model influences how adding more

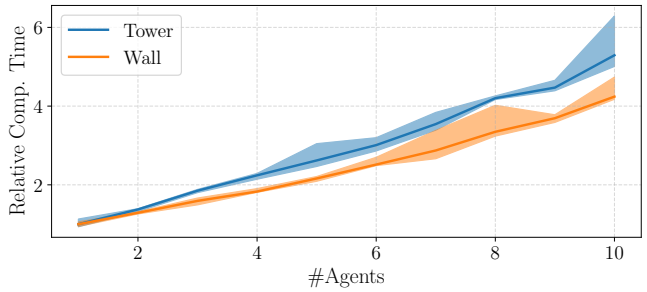



Fig. 9: The relative computation time for different numbers of agents for the tower and the wall scenario over 10 runs. The line is the median and the shaded area is the 25 and 75 percentile, respectively.

TABLE I: Median computation time over 10 runs for Tower and Wall, 5 for Well and Pavilion for the components of the algorithm for all models, with number of agents $m$. The total time additionally contains e.g. pre-processing of the model, deciding on task sequences etc. The super- and subscripts are the difference to the 25, and 75 percentile.

| | | Time [s] | | | |
|---|---|---|---|---|---|
| | $m$ | Keyframes-opt | Path-planning | Postprocessing | Total |
| Tower (15 obj.) | 1 | $13.0^{+2.0}_{-1.6}$ | $0.7^{+0.1}_{-0.0}$ | $4.3^{+0.3}_{-0.2}$ | $19.3^{+2.5}_{-1.3}$ |
| | 5 | $19.7^{+0.3}_{-1.3}$ | $4.7^{+0.4}_{-0.2}$ | $15.8^{+0.8}_{-0.6}$ | $50.6^{+8.2}_{-3.0}$ |
| | 10 | $25.8^{+4.7}_{-1.4}$ | $13.3^{+10.9}_{-0.7}$ | $44.4^{+2.0}_{-3.6}$ | $102.3^{+19.3}_{-5.5}$ |
| Wall (36 obj.) | 1 | $42.1^{+3.0}_{-2.9}$ | $4.2^{+0.2}_{-0.3}$ | $23.9^{+0.7}_{-0.8}$ | $75.4^{+4.3}_{-4.2}$ |
| | 5 | $62.0^{+6.1}_{-0.7}$ | $17.0^{+1.1}_{-1.9}$ | $53.7^{+1.2}_{-3.1}$ | $163.0^{+3.7}_{-5.4}$ |
| | 10 | $102.7^{+6.0}_{-4.8}$ | $50.1^{+4.2}_{-6.1}$ | $107.6^{+5.3}_{-2.4}$ | $319.7^{+37.4}_{-5.0}$ |
| Well (52 obj.) | 1 | $256.8^{+15.1}_{-0.7}$ | $53.1^{+11.2}_{-4.1}$ | $131.7^{+26.2}_{-3.5}$ | $450.2^{+15.5}_{-13.0}$ |
| | 5 | $342.2^{+45.9}_{-16.1}$ | $84.1^{+11.8}_{-1.6}$ | $222.6^{+38.9}_{-1.1}$ | $722.1^{+123.3}_{-15.4}$ |
| Pavilion (113 obj.) | 1 | $734.7^{+113.0}_{-38.5}$ | $311.5^{+26.5}_{-27.3}$ | $511.1^{+86.3}_{-34.2}$ | $1597.7^{+94.7}_{-37.1}$ |
| | 5 | $937.6^{+11.5}_{-75.8}$ | $339.7^{+21.2}_{-8.0}$ | $669.0^{+36.4}_{-28.4}$ | $2336.1^{+86.2}_{-195.2}$ |

robots leads to diminishing returns in the speedup factor. These diminishing returns occur more quickly for the tower, where all agents need to place the objects in close vicinity.

*2) Computation time:* Fig. 9 shows approximately linear scaling with the number of agents for the computation time. This supports that our approach of planning each agent separately, and assuming the previously planned agents as fixed, scales well. Since the tower has a bottleneck where all agents have to come together within a relatively small region, it scales worse than the wall. The objects making up the wall are distributed over a larger space, and thus the agents do not impede each other as much, i.e., the planning problem is easier, since fewer agents per volume are present. Table I breaks down the average necessary computation time for the different steps of the planning process. Specifically, we want to highlight that solving the keyframe-problem takes most of the time of the whole planning process.

### C. Comparison to fixed-time-sampling

We did not put an emphasis on comparison with other methods, as the other methods we are aware of ( [8], [11], [12], [25], [58], [68]) do not scale to the number of robots and objects we consider. We compare our method to a fixed-time sampling scheme[5], which decomposes the problem as the presented approach does, and uses prioritized planning, but does not allow for variable durations of the tasks, i.e. all mode-switches take place at a multiple of $T$. We first note that the fixed-time-sampling approach is extremely sensitive to the choice of $T$. If there are tasks with different duration, or agents that have different maximum speeds, this approach will not allocate the right time, and take an unnecessarily long time for short tasks, or be infeasible altogether.

We compare the schedules generated by our approach, and the fixed-time-sampling approach for building the tower using 7 agents in Figure 10: It is clear that our method achieves better utilization in this example. This is due to the fact that a single agent can temporarily block any other placement when building the tower. This happens in the beginning, where blocks tend to be gripped from above, which blocks placement of another block at the same time. The later blocks need to be grabbed from the side to be placed, as the robots would not be able to place the parts that are higher up if they are grabbed at the top. By coincidence, this enables parallelization for the later blocks for the fixed-time sampling in this example. Placing at a slightly different time would work but is not possible with this approach.

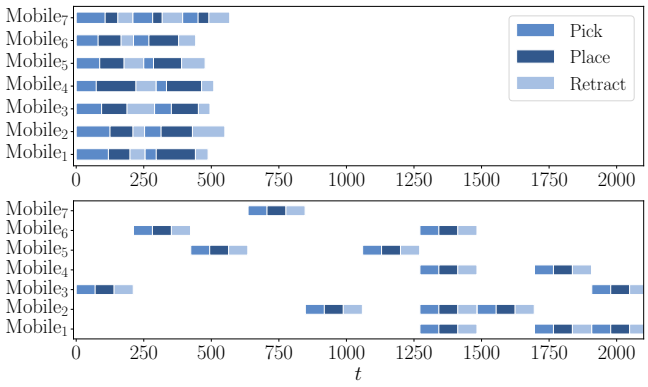



Fig. 10: Illustration of the schedules obtained by our method (top), and fixed-time sampling (bottom) for the tower with 7 mobile agents.

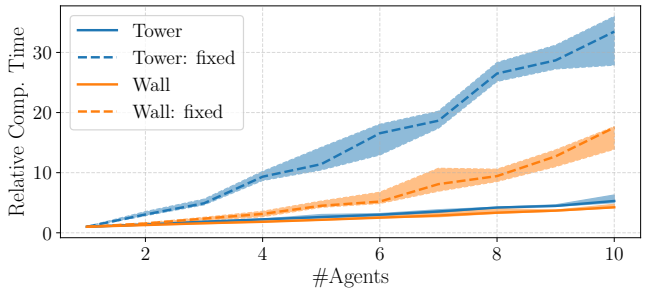



Fig. 11: The relative computation time for different numbers of agents for the tower and the wall scenario over 10 runs for our method and fixed-time sampling. The line is the median and the shaded area shows the 25 and 75 percentile, respectively.

[5]This is similar to what most TAMP solvers would do.

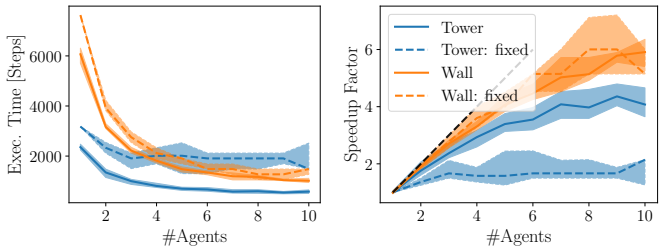



Fig. 12: Comparison of the execution time (left), and the speedup of the execution time when using multiple agents (right) for the tower and the wall models over 10 runs for our method and the fixed-time sampling. The speedup factor is the factor by which the execution time is sped up by using $m$ agents compared to only one agent. The line is the median and the shaded area is bounded by the 25 and 75 percentile.

TABLE II: Median computation times over 10 runs for the *fixed-time* sampling method with number of agents $m$. The total time additionally contains e.g. pre-processing of the model, deciding on task sequences etc. The super- and subscripts are the difference to the 25, and 75 percentile.

| | | Time [s] | | | |
|---|---|---|---|---|---|
| | $m$ | Keyframes-opt | Path-planning | Postprocessing | Total |
| Tower (15 obj.) | 1 | $8.8^{+0.7}_{-0.6}$ | $1.0^{+0.1}_{-0.1}$ | $6.0^{+0.1}_{-0.1}$ | $17.3^{+0.8}_{-0.4}$ |
| | 5 | $141.5^{+69.6}_{-37.2}$ | $23.1^{+9.1}_{-10.7}$ | $16.3^{+7.5}_{-0.4}$ | $197.6^{+48.1}_{-16.6}$ |
| | 10 | $393.4^{+103.9}_{-71.2}$ | $39.2^{+5.8}_{-12.0}$ | $51.1^{+16.0}_{-10.2}$ | $579.1^{+43.8}_{-96.4}$ |
| Wall (36 obj.) | 1 | $28.9^{+5.0}_{-1.9}$ | $4.9^{+0.1}_{-0.3}$ | $30.6^{+0.2}_{-0.2}$ | $70.6^{+3.8}_{-1.6}$ |
| | 5 | $184.2^{+32.3}_{-21.3}$ | $54.1^{+12.2}_{-8.1}$ | $61.2^{+4.2}_{-0.3}$ | $317.8^{+54.5}_{-18.6}$ |
| | 10 | $823.1^{+62.1}_{-104.1}$ | $143.7^{+61.3}_{-65.1}$ | $101.4^{+0.3}_{-1.6}$ | $1235.8^{+9.5}_{-254.8}$ |

We show the same graphs as before, i.e. relative necessary computation time, and relative speedup in Fig. 11, and Fig. 12 respectively, showing that our method both achieves better speedup by using more robots, and better scaling with regard to computation time. Table II shows the absolute computation times broken down to the different parts. While we see that on the Wall-example, a similar speedup can be achieved due to the large space that is available, this is not the case in the Tower-example. The main bottleneck in the fixed-time-sampling is the keyframe generation, which becomes much harder if the space is highly congested, and thus needs more restarts, which leads to a much longer computation time, and much worse scaling of the computation time with the number of agents we plan for.

### D. Demonstrations

*1) Long horizon assembly:* We demonstrate our algorithm on two long horizon-construction tasks using the mobile manipulators, and modeling manipulation constraints as gripping-by-touch:

- The well, using 6 agents: Computation time 14.1 min, execution time 2150 steps.
- The pavilion, using 8 agents: Computation time 43.2 min, execution time 3867 steps.

We show a schedule for the assembly of the pavilion using 8 agents in Fig. 13 to showcase the complexities in coordinating the robot movements.

*2) Handover scenario:* We consider the scenario of the tower again, but this time with three mobile bases with KUKA-arms on top. Manipulation is modeled as gripping-by-touch. Since the KUKA-arms are unable to reach the top of the tower, we add a tower crane. However, the crane is unable to reach the pieces on the floor. Therefore, *handover* sequences are necessary to place the last 3 parts. This scenario demonstrates the ability of our framework to handle and coordinate robots with different capabilities and explore various possible task sequences to fulfill a task. The resulting schedule, and some frames from the process can be seen in Fig. 14 and Fig. 15, respectively.

*3) Real robot experiments:* We demonstrate an experiment with two robotic arms with Robotiq grippers as end-effectors. We model the manipulation constraints of the robot as a two finger gripper. The goal is to stack 6 boxes using the two arms.

The average computation times over 10 runs for the different parts of the algorithm were 264.8s for keyframe-optimization, 4.2s for path-planning, and 58.0s for post-processing. The longer computation times for the keyframe optimization can be attributed to the difficulty of finding valid two-finger-grasps with the naive approach that we used here. It is possible to speed this up using more specialized methods as shown e.g. in [69].

We execute the trajectory open-loop; we thus rely on an accurate model in the simulation, such that the trajectory can be executed without adaptions. In the cases where the execution failed, it was mainly due to violating the assumption of having an accurate model: A typical failure mode was due to inaccurate initial placement of the boxes, which lead to inaccurate final placement, and sometimes knocking over the tower during the motion to place the block. Additionally, the cables of the robot were not taken into account initially, leading to pushing over the tower in some cases. The execution time was between 35s and 45s in all attempts.

Figure 16 shows a sequence of images from the two robots placing the blocks. While it is visible that the boxes are not perfectly aligned, it is clear that the algorithm succeeds in effectively coordinating the robots. This experiment showcases that it is possible to account for the constraints arising in real robot experiments and that the plans and trajectories generated by our algorithm can be executed on a set of real robots.

## VI. Discussion & Limitations

Multi-robot assembly is a complex problem which consists of multiple NP-hard subproblems [32]. The method development in this paper is application-driven, aiming to push towards an efficient solution strategy that scales well to challenging scenarios. There are parts of the approach that were not considered in this work, which need to be tackled when deploying this framework to real construction scenarios.

At the moment, the algorithm is too slow to allow for online-replanning. As visible from the experiments, the keyframe-computation take the majority of the time. Smarter generation

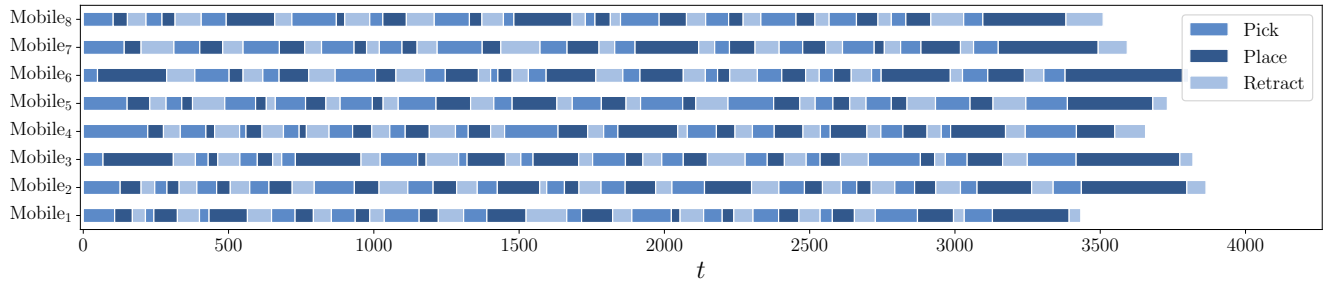



Fig. 13: The schedule for a set of 8 mobile agents assembling the pavilion.

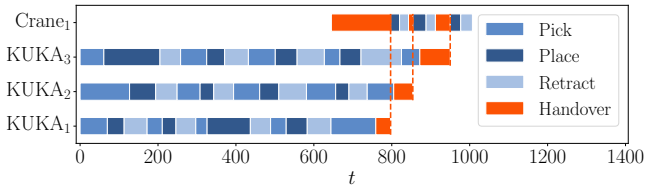



Fig. 14: Illustration of a schedule for the Tower-example using three KUKA-arms on mobile bases and a tower crane, therefore requiring robot-robot interaction. We specifically highlight the times at which constraints are active for two robots (*handover*), and which agents are affected.

of the keyframes, e.g., through online learning of initializations for the optimizer, is one way to decrease the time [69], [70]. Reducing the number of sampled goal states is another possibility to scale down the necessary time. For speeding up the planning, the biggest possible improvement is the usage of multiquery planning. Applying multiquery path planning in a dynamic environment is not straightforward however, and needs to be considered as future work. A first step towards multiquery planning in the construction setting was shown in [71].

While we showed that the paths can be executed by real robots directly in a controlled environment, in case uncertainty is present, this might be more challenging. Hence, the planning done in this framework might need to account for uncertainty in the execution, both in space, and in time. One possibility to do so would be to reserve a 'safe corridor' for an agent, through which no other agent travels for a given time-window. Another one would be to, e.g., consider clearance during the path planning, and not only to minimize time.

Finally, when applying the approach presented here to planning for real world construction scenarios, further static and ordering constraints of the assembly sequencing need to be taken into account. Decoupling of assembly sequencing from the lower level TAMP problem is one way to deal with such additional constraints. Finding valid assembly sequences could be done with, e.g., multi-bound tree search together with assembly/disassembly planning, which could try to find a feasible sequence for a single agent, and then subsequently use the approach presented here to plan the assembly for multiple agents.

While the algorithm was demonstrated on a model of a real pavilion, scaling to larger number of parts might be necessary in other real world scenarios. The demonstrations in the previous sections support our belief that this algorithm is able to scale to larger numbers of objects, but it is part of future work to scale this even further.

### A. Discussion of Theoretical Properties

We briefly discuss the properties of the algorithm, and necessary changes to achieve completeness.

*a) Selection of Subproblems:* The priorization of objects and robots in combination with the backtracking can be seen as depth first search. This means that every possible assignment will be chosen at some point. The methods we use for path-planning can not prove an infeasibility, which is why we need to continue exploring the 'infeasible' nodes.

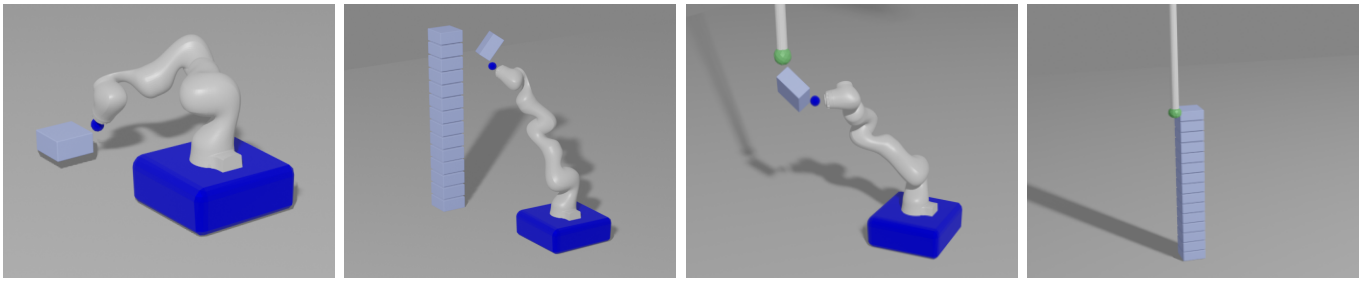
*b) Choice of action-sequence:* We are using a depth-first search over the space defined by a first-order logic-language to compute the available action sequences. Under the assumption that each object can be rearranged to its goal position within a finite number of actions, this implies that we will find a feasible action sequence, if one exists.

*c) Path-planning:* Path-planning in our algorithm works sequentially through an action-sequence, and plans each part of the sequence using an RRT. This means that each segment alone is asymptotically complete in the limit, assuming that the keyframe-sampling is uniformly covering the solution manifold and that we return to a subproblem infinite number of times. It might be possible that we find a path in one part of the action sequence which does not have a corresponding feasible path in the following action, and thus label the sequence as 'infeasible'. The occurrence of this is greatly reduced by jointly sampling all keyframes. Continuously re-expanding the 'infeasible' nodes covers this case.

We finally note that we need to make sure in the algorithm that we do not make subsequent placements impossible by fixing paths that do not allow for a feasible subsequent path, i.e., do not block *pick* or *place* configurations with 'resting' agents.

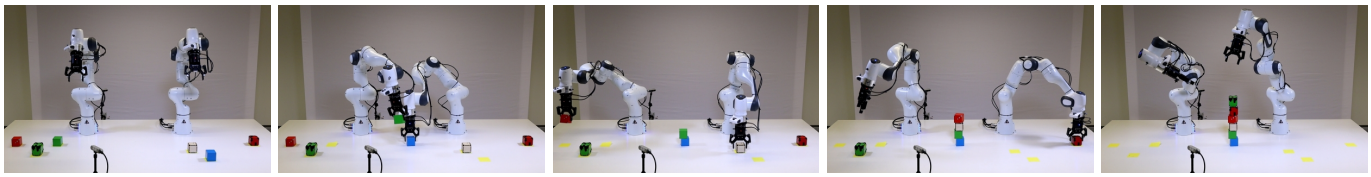
## VII. Conclusion

We presented a planning system to solve long-horizon multi-robot construction assembly problems that integrates several novel components. The approach strongly exploits the factorizations of multi-agent construction assembly problems, by solving simpler subproblems involving only a subset of



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

Fig. 15: Illustration of a handover scenario with a KUKA-arm and a crane. **(a)** KUKA-arm picks up the object. **(b)** The placement position is not reachable. **(c)** A handover to the crane is executed. **(d)** The crane places the object onto tower.



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

Fig. 16: Snapshots of the robot experiment, where two Panda arms with two-finger gripper end-effectors are stacking six boxes.

agents to plan for and a single object that has to be placed at its goal location. These solutions to the separate subproblems are then used to construct a feasible solution to the overall problem. To solve the limited-horizon subproblems, we combine sampling based path planning with joint mode-switch optimization to solve for manipulation constraints, and proposed novel methods to find time-embeddings for planned tasks. Path planning between keyframes amidst other moving, previously planned, objects and robots is achieved using a novel bi-directional RRT-planner in space-time.

We demonstrate that our approach scales well to many robots and many objects on a variety of construction tasks. We provide both qualitative and quantitative analysis of the results. Compared to planning task assignments at fixed times, our time-embedding leads to better utilization of the robots and hence lower execution time to achieve the task, as well as lower computation times for planning the movements. Finally, we demonstrated the approach in a real robot experiment. The robotic experiments show that it is possible to execute the paths that our approach generates.

The approach exploits decompositionality and greedily selects the next sub-tasks. This is successful for our application scenarios, but compromises global optimality for efficient planning and execution times, as is crucial to make multi-robot planning work.

We want to push the approach to demonstration on real construction scenarios. In this setting, more realism in the model description is required, including exact physical constraints on static stability.

## VIII. Acknowledgement

The authors thank Christoph Schlopschnat for the model of the wooden pavilion.

## References

[1] McKinsey & Company, "The next normal in construction: How disruption is reshaping the world's largest ecosystem," 2020.

[2] J. M. Davila Delgado, L. Oyedele, A. Ajayi, L. Akanbi, O. Akinade, M. Bilal, and H. Owolabi, "Robotics and automated systems in construction: Understanding industry-specific challenges for adoption," *Journal of Building Engineering*, vol. 26, p. 100868, 2019.

[3] H. J. Wagner, M. Alvarez, O. Kyjanek, Z. Bhiri, M. Buck, and A. Menges, "Flexible and transportable robotic timber construction platform – TIM," *Automation in Construction*, vol. 120, p. 103400, 2020.

[4] R. A. Knepper, T. Layton, J. Romanishin, and D. Rus, "IkeaBot: An autonomous multi-robot coordinated furniture assembly system," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, May 2013, pp. 855–862.

[5] I. Rodríguez, K. Nottensteiner, D. Leidner, M. Durner, F. Stulp, and A. Albu-Schäffer, "Pattern Recognition for Knowledge Transfer in Robotic Assembly Sequence Planning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3666–3673, 2020.

[6] F. Suárez-Ruiz, X. Zhou, and Q.-C. Pham, "Can robots assemble an IKEA chair?" *Science Robotics*, vol. 3, no. 17, p. eaat6385, 2018.

[7] C. Garrett, Y. Huang, T. Lozano-Perez, and C. Mueller, "Scalable and Probabilistically Complete Planning for Robotic Spatial Extrusion," in *Proc. of Robotics: Science and Systems (R:SS)*, Corvalis, Oregon, USA, July 2020.

[8] V. N. Hartmann, O. S. Oguz, D. Driess, M. Toussaint, and A. Menges, "Robust task and motion planning for long-horizon architectural construction planning," in *Proc. of the IEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2020.

[9] S. Leder, R. Weber, D. Wood, O. Bucklin, and A. Menges, "Distributed Robotic Timber Construction: Designing of in-situ timber construction system with robot-material collaboration," *ACADIA – Ubiquity and Autonomy*, 2019.

[10] M. Toussaint, K. R. Allen, K. A. Smith, and J. B. Tenenbaum, "Differentiable Physics and Stable Modes for Tool-Use and Manipulation Planning," in *Proc. of Robotics: Science and Systems (R:SS)*, 2018.

[11] T. Pan, A. M. Wells, R. Shome, and L. E. Kavraki, "A General Task and Motion Planning Framework For Multiple Manipulators," in *Proc. of the IEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2021, pp. 3168–3174.

[12] J. Ota, "Rearrangement planning of multiple movable objects by using real-time search methodology," in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, vol. 1, 2002, pp. 947–953 vol.1.

[13] M. Stilman, J.-U. Schamburek, J. Kuffner, and T. Asfour, "Manipulation

planning among movable obstacles," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*. IEEE, 2007, pp. 3327–3332.
[14] M. Erdmann and T. Lozano-Perez, "On multiple moving objects," *Algorithmica*, vol. 2, no. 1-4, p. 477, 1987.
[15] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez, "Integrated task and motion planning," *arXiv preprint arXiv:2010.01083*, 2020.
[16] K. S. Saidi, T. Bock, and C. Georgoulas, *Robotics in Construction*. Cham: Springer International Publishing, 2016, pp. 1493–1520.
[17] F. Robotics, "Hadrian x begins first commercial building," Youtube, 2015. [Online]. Available: https://www.youtube.com/watch?v=qWGrIQqGwAM
[18] R. L. Johns, M. Wermelinger, R. Mascaro, D. Jud, F. Gramazio, M. Kohler, M. Chli, and M. Hutter, "Autonomous dry stone," *Construction Robotics*, vol. 4, no. 3, pp. 127–140, 2020.
[19] F. Augugliaro, S. Lupashin, M. Hamer, C. Male, M. Hehn, M. W. Mueller, J. S. Willmann, F. Gramazio, M. Kohler, and R. D'Andrea, "The Flight Assembled Architecture installation: Cooperative construction with flying machines," *IEEE Control Systems Magazine*, vol. 34, no. 4, pp. 46–64, 2014.
[20] R. Mascaro, "Towards Automating Construction Tasks: Large-Scale Object Mapping, Segmentation and Manipulation," *Journal of Field Robotics*, 2020.
[21] R. Naboni, A. Kunic, A. Kramberger, and C. Schlette, "Design, simulation and robotic assembly of reversible timber structures," *Construction Robotics*, pp. 1–10, 2021.
[22] S. Parascho, I. X. Han, S. Walker, A. Beghini, E. P. Bruun, and S. Adriaenssens, "Robotic vault: A cooperative robotic assembly method for brick vault construction," *Construction Robotics*, vol. 4, no. 3, pp. 117–126, 2020.
[23] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.
[24] G. Wagner and H. Choset, "Subdimensional expansion for multirobot path planning," *Artificial Intelligence*, vol. 219, pp. 1–24, 2015.
[25] R. Shome and K. E. Bekris, "Synchronized Multi-Arm Rearrangement Guided by Mode Graphs with Capacity Constraints," *arXiv preprint arXiv:2005.09127*, 2020.
[26] K. Solovey, O. Salzman, and D. Halperin, "Finding a needle in an exponential haystack: Discrete RRT for exploration of implicit roadmaps in multi-robot motion planning," *International Journal of Robotics Research*, vol. 35, no. 5, pp. 501–513, 2016.
[27] H. Ha, J. Xu, and S. Song, "Learning a Decentralized Multi-arm Motion Planner," *arXiv preprint arXiv:2011.02608*, 2020.
[28] M. Bennewitz, W. Burgard, and S. Thrun, "Optimizing schedules for prioritized path planning of multi-robot systems," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, vol. 1. IEEE, 2001, pp. 271–276.
[29] W. Wu, S. Bhattacharya, and A. Prorok, "Multi-robot path deconfliction through prioritization by path prospects," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 9809–9815.
[30] J. van Den Berg, J. Snoeyink, M. C. Lin, and D. Manocha, "Centralized path planning for multiple robots: Optimal decoupling into sequential plans." in *Proc. of Robotics: Science and Systems (R:SS)*, vol. 2, 2009, pp. 2–3.
[31] D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock, "Randomized kinodynamic motion planning with moving obstacles," *International Journal of Robotics Research*, vol. 21, no. 3, pp. 233–255, 2002.
[32] J. Reif and M. Sharir, "Motion planning in the presence of moving obstacles," *Journal of the ACM (JACM)*, vol. 41, no. 4, pp. 764–790, 1994.
[33] A. Sintov and A. Shapiro, "Time-based RRT algorithm for rendezvous planning of two dynamic systems," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 6745–6750.
[34] 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.
[35] A. Krontiris and K. E. Bekris, "Dealing with Difficult Instances of Object Rearrangement." in *Proc. of Robotics: Science and Systems (R:SS)*, 2015.
[36] L. S. H. de Mello and S. Lee, *Computer-aided mechanical assembly planning*. Springer Science & Business Media, 2012, vol. 148.
[37] S. Lee, "Backward assembly planning with assembly cost analysis," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 1992, pp. 2382–2391 vol.3.
[38] G. T. Kao, A. Körner, D. Sonntag, L. Nguyen, A. Menges, and J. Knippers, "Assembly-aware design of masonry shell structures: a computational approach," in *Proceedings of IASS Annual Symposia*, vol. 23. International Association for Shell and Spatial Structures (IASS), 2017, pp. 1–10.
[39] E. P. Bruun, R. Pastrana, V. Paris, A. Beghini, A. Pizzigoni, S. Parascho, and S. Adriaenssens, "Three cooperative robotic fabrication methods for the scaffold-free construction of a masonry arch," *arXiv preprint arXiv:2104.04856*, 2021.
[40] J. Seo, M. Yim, and V. Kumar, "Assembly planning for planar structures of a brick wall pattern with rectangular modular robots," in *2013 IEEE International Conference on Automation Science and Engineering (CASE)*, 2013, pp. 1016–1021.
[41] Y. Deng, Y. Hua, N. Napp, and K. Petersen, "A compiler for scalable construction by the termes robot collective," *Robotics and Autonomous Systems*, vol. 121, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889019301897
[42] K. H. Petersen, R. Nagpal, and J. K. Werfel, "Termes: An autonomous robotic system for three-dimensional collective construction," *Robotics: science and systems VII*, 2011.
[43] K. H. Petersen, N. Napp, R. Stuart-Smith, D. Rus, and M. Kovac, "A review of collective robotic construction," *Science Robotics*, vol. 4, no. 28, 2019.
[44] R. Alami, J.-P. Laumond, and T. Simeon, "Two manipulation planning algorithms," in *Workshop on the Algorithmic Foundations of Robotics*. AK Peters, Ltd. Natick, MA, USA, 1994, pp. 109–125.
[45] M.-T. Khoury, A. Orthey, and M. Toussaint, "Efficient sampling of transition constraints for motion planning under sliding contacts," 2020, arXiv:2011.01552 [cs.RO].
[46] W. Vega-Brown and N. Roy, "Asymptotically optimal planning under piecewise-analytic constraints," in *Algorithmic Foundations of Robotics XII*. Springer, 2020, pp. 528–543.
[47] K. Hauser and V. Ng-Thow-Hing, "Randomized multi-modal motion planning for a humanoid robot manipulation task," *International Journal of Robotics Research*, vol. 30, no. 6, pp. 678–698, 2011.
[48] Z. Kingston, M. Moll, and L. E. Kavraki, "Exploring implicit spaces for constrained sampling-based planning," *International Journal of Robotics Research*, vol. 38, no. 10–11, pp. 1151–1178, 9 2019.
[49] J. Mirabel and F. Lamiraux, "Handling implicit and explicit constraints in manipulation planning," in *Robotics: Science and Systems 2018*, 2018, p. 9p.
[50] L. P. Kaelbling and T. Lozano-Pérez, "Hierarchical task and motion planning in the now," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*. IEEE, 2011, pp. 1470–1477.
[51] M. Toussaint, J.-S. Ha, and D. Driess, "Describing physics for physical reasoning: Force-based sequential manipulation planning," *IEEE Robotics and Automation Letters*, 2020.
[52] N. T. Dantam, Z. K. Kingston, S. Chaudhuri, and L. E. Kavraki, "An incremental constraint-based framework for task and motion planning," *International Journal of Robotics Research*, vol. 37, no. 10, pp. 1134–1151, 2018.
[53] W. Thomason and R. A. Knepper, "A unified sampling-based approach to integrated task and motion planning," in *International Symposium of Robotics Research*, 2019.
[54] T. Migimatsu and J. Bohg, "Object-Centric Task and Motion Planning in Dynamic Environments," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 844–851, 2020.
[55] P. S. Schmitt, F. Wirnshofer, K. M. Wurm, G. v. Wichert, and W. Burgard, "Modeling and planning manipulation in dynamic environments," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 176–182.
[56] C. Phiquepal and M. Toussaint, "Combined task and motion planning under partial observability: An optimization-based approach," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 9000–9006.
[57] N. Funk, G. Chalvatzaki, B. Belousov, and J. Peters, "Learn2assemble with structured representations and search for robotic architectural construction," in *Conference on Robot Learning*. PMLR, 2022, pp. 1401–1411.
[58] M. Levihn, T. Igarashi, and M. Stilman, "Multi-robot multi-object rearrangement in assignment space," in *Proc. of the IEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. IEEE, 2012, pp. 5255–5261.
[59] J. Ortiz-Haro, V. N. Hartmann, O. S. Oguz, and M. Toussaint, "Learning Efficient Constraint Graph Sampling for Robotic Sequential Manipulation," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2021.
[60] D. Berenson, S. S. Srinivasa, D. Ferguson, and J. J. Kuffner, "Manipulation planning on constraint manifolds," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*. IEEE, 2009, pp. 625–632.

[61] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, vol. 2. IEEE, 2000, pp. 995–1001.
[62] F. Grothe, V. N. Hartmann, A. Orthey, and M. Toussaint, "ST-RRT*: Asymptotically-Optimal Bidirectional Motion Planning through Space-Time," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2022.
[63] R. Geraerts and M. H. Overmars, "Creating High-quality Paths for Motion Planning," *International Journal of Robotics Research*, vol. 26, no. 8, pp. 845–863, 2007.
[64] M. Toussaint, "KOMO: Newton methods for k-order Markov constrained motion problems," e-Print arXiv:1407.0414, 2014.
[65] K. Kant and S. W. Zucker, "Toward efficient trajectory planning: The path-velocity decomposition," *International Journal of Robotics Research*, vol. 5, no. 3, pp. 72–89, 1986.
[66] Q.-C. Pham, S. Caron, P. Lertkultanon, and Y. Nakamura, "Admissible velocity propagation: Beyond quasi-static path planning for high-dimensional robots," *International Journal of Robotics Research*, vol. 36, no. 1, pp. 44–67, 2017.
[67] M. Toussaint and M. Lopes, "Multi-bound tree search for logic-geometric programming in cooperative manipulation domains," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2017.
[68] J. Chen, J. Li, Y. Huang, C. Garrett, D. Sun, C. Fan, A. Hofmann, C. Mueller, S. Koenig, and B. C. Williams, "Cooperative task and motion planning for multi-arm assembly systems," 2022, preprint at http://web.mit.edu/yijiangh/www/papers/chen2022cooperative.pdf.
[69] J. Ortiz-Haro, J.-S. Ha, D. Driess, and M. Toussaint, "Structured deep generative models for sampling on constraint manifolds in sequential manipulation," in *Conference on Robot Learning*. PMLR, 2022, pp. 213–223.
[70] D. Driess, J.-S. Ha, and M. Toussaint, "Learning to solve sequential physical reasoning problems from a scene image," *The International Journal of Robotics Research*, vol. 40, no. 12-14, pp. 1435–1466, 2021.
[71] V. N. Hartmann, M. P. Strub, M. Toussaint, and J. D. Gammell, "Effort informed roadmaps (EIRM*): Efficient asymptotically optimal multi-query planning by actively reusing validation effort," in *Proceedings of the International Symposium on Robotics Research (ISRR)*, 2022, to Appear.

**Valentin N. Hartmann** received the B.S. degree and the M.S. degree in mechanical engineering from ETH Zurich, Zurich, Switzerland in 2016 and 2019. He is currently working towards the Ph.D. degree at TU Berlin, Berlin, Germany in the LIS group.

His research interests include task and motion planning for robot teams.

**Andreas Orthey** received the B.S. degree and the M.S. degree in computational engineering from TU Berlin, Berlin, Germany in 2011 and 2012. In 2015, he received a Ph.D. degree in computer science from the National Polytechnic Institute of Toulouse, France. Afterwards, he worked as a Postdoctoral fellow at the National Institute of Advanced Industrial Science and Technology in Japan, where he was funded by a JSPS fellowship. In 2018, he worked at the University of Stuttgart, Germany, supported by a fellowship from the Alexander von Humboldt Foundation, and later as a postdoc at the Max-Planck Institute for Intelligent Systems. Since 2021, he is working as staff robotics scientist at Realtime Robotics, and he is a guest lecturer at the TU Berlin.

**Danny Driess** received the B.S. degree and the M.S. degree in simulation technology from the University of Stuttgart, Stuttgart, Germany in 2016 and 2019. He is currently working towards the Ph.D. degree at TU Berlin, Berlin, Germany in the LIS group.

His research interests include learning in task and motion planning with a special focus on connecting perception and planning for sequential manipulation tasks through machine learning.

**Ozgur S. Oguz** received his B.S. and M.S. degrees from Koç University, Istanbul, Turkey, in 2007 and 2010. He received his Ph.D. degree from the Electrical and Computer Engineering Department at TU Munich, Munich, Germany, in 2018. He then worked as Postdoctoral Researcher at the University of Stuttgart and Max Planck Institute for Intelligent Systems.

In spring 2022 he joined the department of computer engineering at Bilkent University, Ankara, Turkey, as Assistant Professor. His research focuses on learning and decision theory with applications in robotics.

**Marc Toussaint** received the Ph.D. degree in neuroinformatics from the Ruhr University Bochum, Bochum, Germany in 2004. He was a Postdoctoral Fellow in the Machine Learning group and the Statistical Machine Learning and Motor Control group at the University of Edinburgh. He then headed the Machine Learning and Robotics group from 2007 to 2010 at TU Berlin, Berlin, Germany supported by the Emmy Noether Proramme. He was Professor at the department of math and computer science at the FU Berlin, Berlin, Germany, until 2012, and Professor at the University of Stuttgart, Stuttgart, Germany since 2012 until 2020.

He is currently a Professor at TU Berlin, Berlin, Germany at the department of electrical engineering and computer science, and a member of the Science of Intelligence excellence cluster.