2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Robust Task and Motion Planning for Long-Horizon Architectural Construction Planning

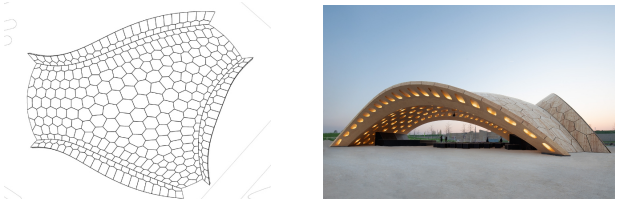
Valentin N. Hartmann[1], Ozgur S. Oguz[1,2], Danny Driess[1,2], Marc Toussaint[1,2] and Achim Menges[3]

***Abstract*— Integrating robotic systems in architectural and construction processes is of core interest to increase the efficiency of the building industry. Automated planning for such systems enables design analysis tools and facilitates faster design iteration cycles for designers and engineers. However, generic task-and-motion planning (TAMP) for long-horizon construction processes is beyond the capabilities of current approaches. In this paper, we develop a multi-agent TAMP framework for long horizon problems such as constructing a full-scale building. To this end we extend the Logic-Geometric Programming framework by sampling-based motion planning, a limited horizon approach, and a task-specific structural stability optimization that allow an effective decomposition of the task. We show that our framework is capable of constructing a large pavilion built from several hundred geometrically unique building elements from start to end autonomously.**

I. INTRODUCTION

Building construction is the largest industry worldwide. It consumes around 40% of global resources and energy, and produces 50% of all waste [1]. Yet, current construction processes, their planning and supervision, as well as the building designs are far from optimal. Building information modeling (BIM), an increasingly employed method in construction, comprises digital tools designed to manage established, primarily manual construction processes with conventional building elements. We believe that AI and robotics have the potential to revolutionize the area, moving towards integrated reasoning about the robotized construction process jointly with the building design, resource consumption, and uncertainties.

This paper aims to make a first step in this direction. We address the problem of computing a possible robotic construction sequence for a given building design. We reason on the kinematic, geometric and static stability level, neglecting the dynamic constraints and actual control problem of execution. While assuming the final design as given in this work, the long term goal is to reason jointly over the construction process and the building design: Such a complementary design-for-robotic-assembly framework should suggest modifications that lead to more efficient construction, and help discover and explore construction processes and related possible designs that are beyond current designer's intuition and conventions. We aim to think of the design of the architecture and the construction process as a joint problem. In its current form, our framework supports an architect in reviewing a potential robotic construction process for his design, possibly yielding insights to modify the design.



(a) Schematic top view. (b) Assembled pavilion.

Fig. 1: BUGA Wood pavilion built by our framework in simulation.

For the demonstration of the planning framework in this paper we consider a long-span pavilion design: the BUGA Wood pavilion (Fig. 1), which was conventionally built in Heilbronn (DE) at the Bundesgartenschau 2019 [2]. This example is particularly suitable, as it showcases how the differentiation of building elements, enabled by robotic prefabrication, results in a high-performance, materially efficient structure. However, the assembly and construction remained entirely manual and conventional.

The problem setting is challenging from the perspective of robot task and motion planning due to several reasons. The overall construction requires to assemble 376 geometrically unique pieces – in our model this is done by coordinating 2 robots: one crane and one mobile robot for final placement. A core issue when scaling TAMP to long horizons is the trade-off between decomposing the problem and aiming for joint optimality: Previous work on optimization-based TAMP treated the whole manipulation path jointly which will not scale to realistic construction domains. Additionally, for such long horizon problems, the TAMP-solver needs to be robust over a wide range of arising motion planning problems, while still providing solutions in a reasonable timeframe. Clearly, we can treat many aspects roughly independently or incrementally provided that certain objectives, e.g., static feasibility, are satisfied. As such, it should be possible to achieve close to linear scaling of the planning-complexity with the number of required manipulation steps. We therefore believe that one of the core technical challenges is to identify the (inter)dependence between manipulations/individual pieces, which in turn would determine planning jointly or

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

[1]Machine Learning & Robotics Lab, University of Stuttgart, Germany {firstname}.{lastname}@ipvs.uni-stuttgart.de
[2]Max Planck Institute for Intelligent Systems, Germany
[3]Institute for Computational Design and Construction, University of Stuttgart, Germany achim.menges@icd.uni-stuttgart.de

sequentially.

We propose to follow a limited horizon approach along with task specific objectives as a form of decomposing the problem, and introduce sampling-based approaches in combination with optimization-based methods to form a robust TAMP-solver. We demonstrate this approach in the Logic-Geometric Programming (LGP) framework introduced in [3], which formulates the TAMP as an optimization problem.

The key contributions of this paper are:

- A limited (receding) horizon planning approach as a form of decomposition in task and motion planning,
- Planning for multiple robots with mixed capabilities,
- Introducing methods to robustly solve an LGP problem, such as sampling-based motion planning and stochastic restarts,
- Integrating static stability estimates in TAMP planning,
- Demonstrating TAMP methods on architectural construction.

II. RELATED WORK

A. Task and motion planning

A full discussion of the TAMP literature is beyond our scope. In relation to problem decomposition, Kaelbling et al. introduced 'Hierarchical Planning in the Now' [4], which decomposes the problem strictly hierarchically, thereby limiting the need for long-term decision making, while still being suitable for long-horizons. In [5] a highly scalable symbolic planner is incorporated in the TAMP framework to achieve longer planning horizons. Recent work also investigates learning heuristics to guide the high-level planning in order to scale these methods for long horizon sequential manipulation problems [6]. However, currently proposed benchmarks lack very long-horizon problems [7], and to the best of our knowledge, there has not been any long-horizon application comparable to the one investigated in this paper demonstrated so far.

While simplifying problems is a common approach to make problems tractable in path planning [8] and optimization [9] – the two areas from which we leverage methods – it is not yet investigated in detail within TAMP settings. In [10] a team of multiple robots assembles a chair, where the planning problem is simplified by decomposing the problem into smaller subproblems by regrasping. The research in [11] suggests the incorporation of hierarchies to solve a problem with less decisions before moving on to the actual problem, while noting that the worst-case scenario is still the same.

B. Robotics in Fabrication, Architecture, and Construction

As the interest in autonomous construction increases, [12] identifies some challenges (e.g., cluttered working environments, high reliability to be useful, uncertainty in sensing) and current use cases (e.g., bricklaying, masonry).

Computational and robotic approaches in architecture and design are becoming more relevant as well and are often already deployed in reality: [13] discusses the assembly of specially designed wooden structures using two robotic arms for precise positioning. The application of a mobile robot for semi-autonomous construction of a brick wall at the DFAB house, a demonstrator building for research concepts, is presented in [14]. In [15], flying vehicles are used for the construction of a tower consisting of foam-bricks in an art project. In [16], use-cases in prefabrication and various mobile robots are presented, while noting that few of them are currently economically viable due to their specialized use cases.

While demonstrating use cases of robots in construction, these projects lack the explicit use of incorporating the robotic planning in the design process. The robots are used for the construction, and made fit, rather than informing the design of what is feasible, which is what we are aiming for.

III. BACKGROUND

A. Logic-geometric programming

In this section, we formulate Logic-Geometric Programming (LGP) [3], [17], as the underlying TAMP framework for the purpose of this work. The main idea of LGP is to introduce a discrete variable $s \in \mathcal{S}$ (the state of a symbolic domain $\mathcal{S}$) that parameterizes the costs and constraints of a nonlinear trajectory optimization problem (NLP) over the (continuous) path x. Let $\mathcal{X} \subset \mathbb{R}^n \times SE(3)^m$ be the configuration space of m rigid objects and n articulated joints of potentially multiple agents with initial condition x_0. Given a goal description g (details in Section IV-B), LGP tries to find a sequence of symbolic states $s_{1:K}$, which we call a *skeleton*, and a path $x : [0, KT] \to \mathcal{X}$ in the configuration space that minimizes

$$
\begin{aligned}
\min_{\substack{x,K,\\ a_{1:K},s_{1:K}}} \quad & \int_0^{KT} f_{\text{path}}(\overline{x}(t)) + \psi(x(t), s_{\tilde{k}(t)})\,\mathrm{d}t + f_{\text{goal}}(x(KT), g) \\
\text{s.t.} \quad & x(0) = x_0,\ s_0 = \tilde{s}_0,\ h_{\text{goal}}(x(KT), g) = 0 \\
& \forall t \in [0, KT]: \quad h_{\text{path}}\big(\overline{x}(t), s_{\tilde{k}(t)}\big) = 0 \\
& \qquad\qquad\qquad\quad g_{\text{path}}\big(\overline{x}(t), s_{\tilde{k}(t)}\big) \leq 0 \\
& \forall k \in 1, \dots, K: \ h_{\text{switch}}(\hat{x}(t), a_k) = 0 \\
& \qquad\qquad\qquad\quad g_{\text{switch}}(\hat{x}(t), a_k) \leq 0 \\
& \qquad\qquad\qquad\quad a_k \in \mathcal{A}(s_{k-1}) \\
& \qquad\qquad\qquad\quad s_k \in \text{succ}(s_{k-1}, a_k) \\
& \qquad\qquad\qquad\quad s_K \in \mathcal{G}(g),
\end{aligned} \tag{1}
$$

where $\overline{x} = (x, \dot{x}, \ddot{x})$ and $\hat{x} = (x, \dot{x})$. This path consists of $K \in \mathbb{N}$ phases (kinematic modes [17]) induced by the sequence $s_{1:K}$, each of length $T > 0$ which should be chosen large enough to allow sufficient time for all actions. T can be scaled optionally for each phase individually, using the technique from e.g. [18]. The number of phases is part of the optimization problem itself. We assume the path to be globally continuous and two times continuously differentiable within each phase. f_{path} describes path costs, e.g., squared accelerations. The constraints h_{path} and g_{path} in phase k of the path are parameterized by s_k with $k = \tilde{k}(t) = \lfloor t/T \rfloor$. Therefore, the symbolic state determines the objective in each phase, while the optimization problem tries to find a globally consistent path fulfilling these requirements. The

transitions of s_{k-1} to s_k are determined by a first-order logic language (similar to PDDL) through $\text{succ}(\cdot,\cdot)$ as a function of s_{k-1} and the discrete action $a_k \in \mathcal{A}(s_{k-1})$. The two functions h, g_{switch} impose transition conditions between the modes. $\tilde{s}_0 \in \mathcal{S}$ is the initial symbolic state. The goal is defined symbolically by $s_K \in \mathcal{G}(g)$ and kinematically by f_{goal} and h_{goal} for the goal specification g (see Section IV-B for more details). We extend the formulation of [17] to include a stability cost function ψ, which we will explain in Section IV-C.

B. Multi-Bound Tree Search (MBTS)

The logic introduces a decision tree, where each leaf node, i.e., a node that reaches a symbolic goal state $s_K \in \mathcal{G}(g)$, corresponds to an NLP as a candidate for the overall TAMP problem. Solving the LGP (1) therefore includes a tree search over the discrete actions such that a symbolic goal state is reached and the NLP is feasible. Using the full path optimization problem (1) is too expensive to guide the tree search. Therefore, a key contribution of [3] is to introduce a sequence of relaxations i of (1) in terms of lower bounds $\mathcal{P}_i$, i.e., computationally simpler problems that serve as a lower bound on the cost of the original problem and as a necessary condition on the feasibility. For a given skeleton, (1) is solved by discretizing x in time. This work relies on the following three bounds proposed in [3]: The bound $\mathcal{P}_{\text{pose}}$ optimizes only the final pose for $t = KT$, and $\mathcal{P}_{\text{seq}}$ the switching poses, i.e., x is discretized with configurations corresponding to the switching times $t = 0, T, \dots, KT$ only. The full path optimization problem is called $\mathcal{P}_{\text{full}}$, where x is discretized with n_t points for each phase.

For scaling LGP to an architectural setting, we additionally introduce two new bounds: a stability bound and a sampling-based motion-planner bound in Section IV-C and Section IV-D, respectively.

IV. SCALING LGP TO ARCHITECTURAL CONSTRUCTION

A. Overview

LGP is a very general way to formulate sequential manipulation problems. However, the existing solver (MBTS) is fundamentally limited in scalability: while it leverages approximate bounds to guide tree search over skeletons, it eventually always tries to optimize the full joint path over the whole sequence. In our architectural construction application we need a long sequence of steps, i.e., on the order of thousands of actions a_k, to complete the full task – optimizing jointly across that many steps is not scalable. Further, while the optimization based approach provides many benefits (e.g., in dealing with higher dimensional multi-robot systems jointly, or optimizing interactions such as handovers jointly with the motions), the non-convexity of the motion problems ultimately lead to local optima. To robustly solve large scale problems we need better guarantees of the sub-problem solvers: We cannot tolerate that a sub-problem is labelled infeasible only because an optimizer is trapped in a local optimum. Sampling-based methods can help to gain robustness for such non-convex problems.

In this section we describe key aspects of our solver to address such challenges:

1) We propose an iterative limited-horizon approach to solve long-horizon LGPs, where the formulation of subgoals is a key aspect.
2) We propose a sampling-based motion-planner bound and stochastic restarting to address the challenges of non-convexity.
3) We propose a novel approach to approximate the static stability of the construction, leveraging a constrained optimization solver, which is used as an additional bound to guide tree search.

The last point is specific to architectural construction, where the order of assembly should be guided by static stability. To integrate this reasoning in our framework we need to have computationally efficient approximations, which we propose here.

B. Subgoals & Limited-Horizon LGP

We first propose to enable the planner to decompose the full problem into subgoals, each of which implies a limited-horizon LGP problem, akin to receding horizon planning.

Specifically, in our case, the overall goal specification $g = \bigcup_{i=1}^{m} \{(q_i, p_i)\}$ means that m parts q_i have to be placed at their specified target poses p_i, which translates to the constraint on the last symbolic state

$$s_K \in \mathcal{G}\left(\bigcup_{i=1}^{m} \{(q_i, p_i)\}\right), \tag{2}$$

where $\mathcal{G}(\cdot)$ extracts the symbolic goal state from the goal specification. Instead of attempting to solve (1) with the complete goal (2), we introduce a so-called horizon length n_h and replace the symbolic goal constraint with

$$s_K \in \bigcup_{\{g_i \subseteq g \,:\, |g_i| = n_h\}} \mathcal{G}(g_i). \tag{3}$$

While with the constraint (2) a symbolic goal state is only reached if *all* parts are placed, the subgoal formulation (3) terminates if at least n_h parts are placed. Therefore, the optimization horizon is automatically limited. The overall goal is then achieved by solving (1) with the constraint (3) iteratively for a specified horizon length n_h. Parts that have already been placed in previous iterations are, of course, excluded from the goal specification in the current iteration. Note that neither the subgoals nor their order is given explicitly, but subject to the planner itself. This requires solving each subproblem robustly, while ensuring that those solutions result in physically and kinematically feasible intermediate goals.

C. Static Stability Bound

In our construction setup, we additionally have to prioritize which parts of the building should be placed first based on a stability criterion of the already placed parts. This is

expressed with the static stability term $\psi : (\mathcal{X}, \mathcal{S}) \to \mathbb{R}$, which is added to the path costs in (1) as a function of the current configuration $x \in \mathcal{X}$ and the symbolic state s. ψ contributes to the costs only if in the symbolic state s_k a part is placed.

While existing applications of LGP mainly focused on solving the underlying TAMP problem, i.e., finding a feasible solution, here we explicitly want to minimize this additional cost term. In order to realize this efficiently, we introduce another lower bound, the stability bound $\mathcal{P}_{\text{stab}}$. This bound evaluates ψ for each subgoal, i.e., placement of a part, without taking into account any other constraints or costs that are induced by parts of the configuration space not corresponding to placed objects. The stability term also helps guiding the solver to avoid configurations that make later placement of parts infeasible.

D. Sampling-Based Motion Planning Bound

Solving the full path optimization problem $\mathcal{P}_{\text{full}}$ in our setting is challenging due to collision avoidance in the complex geometries of our building scenario. Therefore, we introduce another lower bound on $\mathcal{P}_{\text{full}}$ that utilizes sampling-based motion planning algorithms, specifically RRTs, to solve this issue. This bound relies on the solution of the switching configurations from $\mathcal{P}_{\text{seq}}$, which allows us to combine the advantages of jointly optimizing these configurations including multi-agents and handovers with the path finding capabilities of the sampling-based planner. In this work, we use RRT-connect [19] , and thus call the bound $\mathcal{P}_{\text{RRT}}$.

Let x^{seq} be the solution of the sequence bound $\mathcal{P}_{\text{seq}}$. The bound $\mathcal{P}_{\text{RRT}}$ now aims to connect each consecutive switching configurations $x^{\text{seq}}_{k-1} = x^{\text{seq}}((k-1)T)$ and $x^{\text{seq}}_k = x^{\text{seq}}(kT)$ for $k = 1, \dots, K$ by solving

$$\begin{aligned} &\text{find } x_{\text{RRT}} : [(k-1)T, kT] \to \mathcal{X} \\ &\text{s.t. } x_{\text{RRT}}((k-1)T) = x^{\text{seq}}_{k-1}, \quad x_{\text{RRT}}(kT) = x^{\text{seq}}_k \\ &\qquad \forall t \in [(k-1)T, kT] : g_{\text{path}}(x_{\text{RRT}}(t)) \leq 0 \\ &\qquad\qquad h_{\text{path}}(x_{\text{RRT}}(t)) = 0. \end{aligned} \tag{4}$$

Equality constraints need special attention in sampling based planners [20]. In our setup, the path constraints h_{path} and g_{path} only consider x, not $\overline{x}$, i.e., we neglect the dynamic constraints for the RRT bound. We note that sampling-based planners have previously been integrated in logic based task planners directly as the solver for the arising motion planning problem [21], but not explicitly as a bound of (1).

E. Stochastic Restarts

Since even the lower bounds for (1) are still non-convex, and the solver might converge to an infeasible local optima, we introduce stochastic restarting of the underlying optimizer (KOMO) [22], which is used to solve the NLPs, with randomized initial conditions for better reliability.

Specifically, we use restarting when solving the problems $\mathcal{P}_{\text{pose}}$ and $\mathcal{P}_{\text{seq}}$. In both cases, we do not sample the variables in the full dimensional space $\mathcal{X}$, but select a lower dimensional subspace to sample from, i.e., the subspace that influences the basin of attraction of the NLP the most, and project them up into the full configuration space (see Section V-D.4 for an example). The other variables are initialized from the current state as solved in the previous iteration.

One has to choose carefully how many restarts are allowed, before a given NLP is declared *infeasible*. We tackle this issue by introducing a computational budget t_{max} for the solution-attempts, and declare a problem as infeasible in case no solution was obtained.

F. Algorithm and Hierarchy of Bounds

This section describes how the bounds are used to efficiently solve the LGP problem for each subgoal and hence the overall long-term TAMP problem.

By construction, the hierarchy of our bounds is

$$\mathcal{P}_{\text{stab}} \prec \mathcal{P}_{\text{pose}} \prec \mathcal{P}_{\text{seq}} \prec \mathcal{P}_{\text{RRT}} \prec \mathcal{P}_{\text{full}}, \tag{5}$$

where $\prec$ means a lower bound with respect to a necessary condition of the feasibility of the following bound. Let $\mathcal{B}$ be the set of all symbolic state sequences that reach the symbolic goal constraint (3) for horizon length n_h. The overall idea of the algorithm is to select $s_{1:K} \in \mathcal{B}$ and then test its feasibility by computing the bounds (5) until either one bound is infeasible or $\mathcal{P}_{\text{full}}$ is feasible and hence a solution has been found.

However, due to the combinatorial complexity, determining $\mathcal{B}$ fully may not be possible even for short horizon lengths. To obtain possible action sequences, the logic-tree is expanded by breadth-first search until a sequence $s_{1:K}$ is reached that fulfills (3), and $\mathcal{P}_{\text{stab}}$ is satisfied as detailed in Section IV-F.1. For the best candidate according to $\mathcal{P}_{\text{stab}}$ we then attempt to compute the pose and sequence bounds, $\mathcal{P}_{\text{pose}}$ and $\mathcal{P}_{\text{seq}}$, respectively.

According to (5), the RRT-bound $\mathcal{P}_{\text{RRT}}$ is a lower bound to $\mathcal{P}_{\text{full}}$, and should thus be computed first. However, since the worst-case run-time of $\mathcal{P}_{\text{RRT}}$ is only limited by a computational budget, which has to be high enough in order not to miss feasible solutions, solving $\mathcal{P}_{\text{RRT}}$ can get expensive. In comparison, $\mathcal{P}_{\text{full}}$ either gives a solution or returns infeasibility much faster. Since a local optimizer is not guaranteed to find a solution for a path between the poses obtained from $\mathcal{P}_{\text{seq}}$ due to non-convexities, the 'infeasibility' assignment by $\mathcal{P}_{\text{full}}$ contains false positives. Therefore, we first solve $\mathcal{P}_{\text{full}}$ (initialized with $\mathcal{P}_{\text{seq}}$) and only if this returns infeasibility we utilize the RRT bound, which, in case it can find a solution, is used as an initialization for a final smoothing step.

1) Details about stability optimization: Ideally, one would solve

$$\operatorname*{argmin}_{s_{1:K}} \sum_{k=1}^{K} \psi(x(kT), s_k) \text{ s.t. } s_{1:K} \in \mathcal{B} \tag{6}$$

to decide which sequence $s_{1:K}$ to evaluate further. However, we only have access to the set $\mathcal{B}' \subseteq \mathcal{B}$, which is grown through the expansion of the tree. To balance the tree expansion and the optimality of the solution, we propose

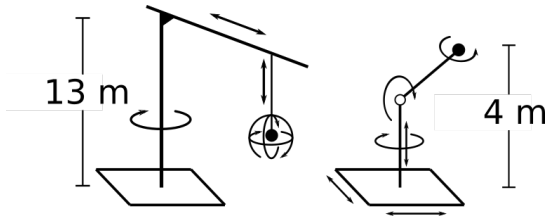


Fig. 2: Schematics of the crane with 6 DoF, and the mobile robot with 6 DoF used in the problem setting.

a scheme that provides such a trade-off, by proceeding with the solution of $\mathcal{P}_{\text{pose}}$ if

$$\sum_{i=1}^{K_1} \psi(x^{(1)}(iT), s_i^{(1)}) - \sum_{j=1}^{K_2} \psi(x^{(2)}(jT), s_j^{(2)}) < \epsilon(|\mathcal{B}'|), \tag{7}$$

where $s_{1:K_1}^{(1)}$ and $s_{1:K_2}^{(2)}$ are the best and the second best sequence found so far that reach the symbolic goal (3), respectively. ϵ is a function of the number of currently found sequences. We additionally impose a minimum size of the set $\mathcal{B}'$, and a maximal computational budget for the expansion. The choice of ϵ is dependent on the behavior of ψ, and the tolerable sub-optimality for the specific problem.

V. EXPERIMENTS & RESULTS

A. The BUGA Wood Pavillon

The BUGA Wood pavilion is made of 376 unique wooden elements which are precisely fabricated by robots. It spans 30 meters, and was assembled by two human operators and one crane in 10 days. For the fabrication of the timber-parts, robotic constraints were taken into account, whereas the construction was manually planned.

In this work, we scale the elements of the pavilion to 80% of their true size to avoid difficulties for the motion planning algorithm when placing the part at its final position, as this positioning is not the main focus of this paper.

B. Robots

In this problem setting, we use two robots with different sets of capabilities (Fig. 2), which mimic how the construction process was done in reality before:

- One crane, which is used to lift the parts, and move them to the handover position. The rotation of the end-effector of the crane around the x and y axis is limited to only allow for small rotations.
- One robot with a mobile base, used for final positioning and placement of the parts. The translation on the z-axis is limited in height, and the rotation of the arm is limited to not bend further down than to a 90 degree angle.

C. Assumptions

We assume that the final pose and position of each part, and thus the whole design of the structure is known before the construction process starts. Hence, the termination criterion of the planner is that each piece has to be placed at its corresponding final position. We also assume:

- No noise is present in the movement for both the crane and the mobile robot,
- Stable grasps by touch for both the crane and the mobile robot.

TABLE I: Predicates to impose constraints on the path optimization

(touch X Y)	distance between X and Y equals 0
(stable X Y)	create stable (constrained to zero velocity) free (7D) joint from X to Y
(postLift X Y)	X is above Y with a distance > 0
(preHandover X Y Z)	X is above Y, which is the goal of Z, with a distance > 0
(fitPose X Y)	pose of X (7D) is exactly at pose of Y

TABLE II: Action operators and the path constraints they imply (brackets indicate a *non-persistent* predicate).

(pick X Y)	[touch X Y] (stable X Y)
(liftUp X Y)	(postLift X Y)
(handover X Y Z)	[touch X Y] (stable X Y) (preHandover X Y Z)
(place X Y)	[stable Y X] (fitPose X Y)

D. Modeling details

1) Logic: We describe the details of the used logic predicates for the specific task of building the BUGA wood pavilion. Most of the implementation details are the same as in [3], [17], with small changes to account for the differences in robots and task domain: The predicates, their implicitly imposed path constraints and the action operators are listed in Table I and Table II, respectively. They are slightly different from the ones used in a normal pick-and-place problem, as we impose that the part has to be placed from above, such that the mobile robot is only 'finalizing' the position of the part.

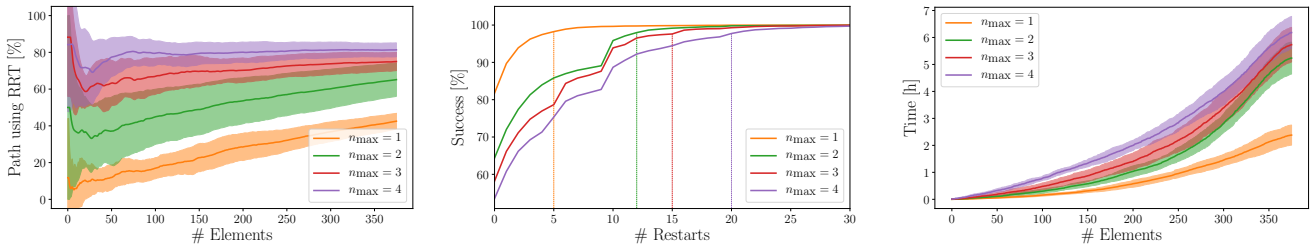
2) Subgoals: We define the subgoals g_i as having placed M ($M \ll m$) parts q_j of the structure successfully, which is equivalent to a horizon length of $n_h = M$. This translates to a minimum of $4 \times M$ necessary actions to fulfill the subgoal. The final goal g is the full assembly of the whole building, i.e., all parts being placed by the robots.

We introduce an adaptive approach of choosing the limited horizon length: We start the algorithm with $n_h = n_{\text{max}}$, and decrease it if within a computational budget no feasible solution has been found. In case we were able to solve the problem for two consecutive subgoals with horizon length n_h, we increase the horizon length given that $n_h < n_{\text{max}}$.

3) Stability bound: We define two stability functions $\psi_{\text{statics}}(x, s)$, and $\psi_{\text{neighbors}}(x, s)$, which can be seen as a fast but inaccurate approximation of $\psi_{\text{statics}}(x, s)$, and compare them. From x we determine the set $\mathcal{U}$ of previously placed parts, and from s the set $\mathcal{W}$ of potentially placeable parts in the evaluated action sequence. We additionally define $\mathcal{N}(w)$ as the set of already placed neighbors of part w:

- $\psi_{\text{statics}} = \frac{1}{|\mathcal{U} \cup \mathcal{W}|} \texttt{Torques}(\mathcal{U} \cup \mathcal{W})$, i.e., we compute[1] the

[1]For simple structures, this can be done analytically. For the BUGA pavilion, we numerically simulate the loads, and extract the torques.



(a) Relative usage of the sampling-based planner throughout runs for different maximum horizon length $n_{\max}$.

(b) Average number of required restarts before finding a solution to the sequence bound $\mathcal{P}_{\text{Seq}}$. The dashed line shows where at least a success rate of 97.5% is attained.

(c) Total computation times for different horizon-lengths $n_{\max}$.

Fig. 3: Usage of the sampling-based planner, required restarts, and computation time using the stability bound ψ_{neighbor} and different $n_{\max}$.

sum of all torques between neighboring parts, and take the average over the building to normalize for number of placed parts.

- $\psi_{\text{neighbors}} = -\frac{1}{|\mathcal{W}|}\sum_{w\in\mathcal{W}} |\mathcal{N}(w)\cap\mathcal{U}|$, i.e., the average number of neighbors per part.

We use both versions of the stability bound with a threshold on the cost to make the bound a necessary condition to avoid attempting to place free floating pieces.

Additionally, we set the maximum size of $|\mathcal{B}'| = 20$ to impose an upper bound on the computational budget of the tree-expansion. Relying on ϵ from (7) alone was not robust enough for the wide range of different scenarios that occur over the construction period.

4) Stochastic restarts: For the mobile robot, we randomly sample the translational coordinates from an uniform distriution, and do not change the other variables. Changing these coordinates leads to convergence of the switching-positions from different sides of the pavilion, which helps the optimizer avoid getting stuck in an infeasible configuration. For the crane, no variables are randomized.

E. Experimental results

In the following section, we analyze several different scenarios, and provide detailed analysis to demonstrate the necessity and the effectiveness of our novel extensions on the default LGP formulation. If not stated differently, the experiments[2] were run several times with different random seeds and the evaluated metrics were averaged to reduce the stochastic effects of the sampling and restarting. A video of a full run with $n_h = 1$ and $\psi_{\text{neighbors}}$ is part of the supplementary material.

1) Robustness: We report the percentage of times the sampling-based planner was invoked, and the required restarts over the whole run in Fig. 3a, and Fig. 3b, respectively. As the environment gets harder for the optimizer to solve for a motion path (i.e., a solution to $\mathcal{P}_{\text{full}}$ using KOMO is not found), especially with higher horizon lengths, our framework reliably switches to the RRT-based motion planner.

2) Stability function: The results of applying different strategies highlight the capability of the developed framework to enable an exploratory functionality that can support users, and specifically architects for this use-case. In effect, different sequences arise with two different stability functions (Fig. 4c, Fig. 4d), compared to the baseline without ordering (Fig. 4b), where we only impose the existence of one connection to a neighbouring part as the sufficient condition. The order used when assembled in reality is shown in Fig. 4a.

3) Horizon length: We compare the required computation time between different horizon lengths n_h over the course of a run (Fig. 3c). We note that, while longer horizons ($n_h > 4$, which translates to a minimum of 20 logical actions) are theoretically possible, our analysis on this is limited due to computational constraints, namely restrictions in the logical search which exhibits combinatorial complexity.

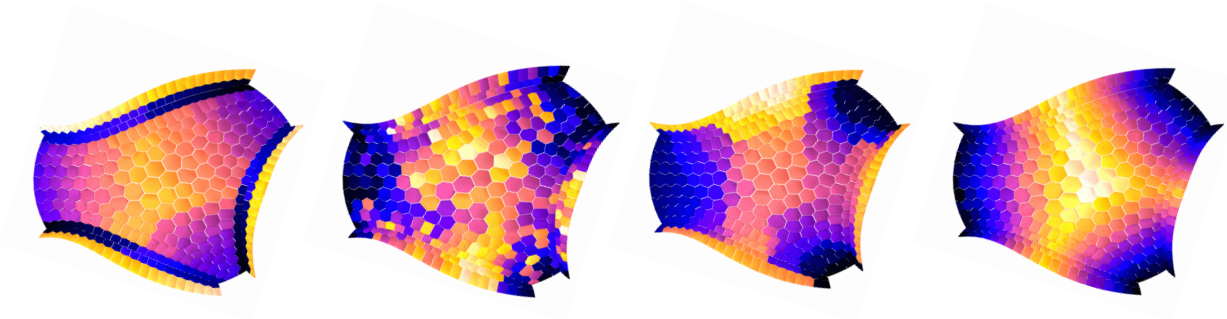
Note that while this pavilion was feasible to build with all horizon lengths, the longer horizon lengths n_h can be necessary for successful planning of other assemblies. The longer horizon lengths enable reasoning about the constraints that are imposed by placing parts further into the future, and thus a more robust method to plan. In addition to enabling more complex construction sequences, a longer horizon length also enables to optimize the resulting trajectories more.

Table III gives a summary of experiments over a range of horizon lengths $n_{\max}$. In the following, we will discuss some of the results in more detail:

- Comparing RRT usage over horizon-length: The results indicate that the necessity of using a robust motion planner becomes more important when dealing with longer horizon lengths. This is intuitive, since optimization can not solve the whole path planning problem even if only one of the 'sub-problems' is infeasible. Hence, the increased reliance on the RRT-planner from $n_h = 1$ to $n_h = 4$ is expected.
- Restarting compared over horizon length: While roughly 80% of all problems for $n_h = 1$ can be solved

[2]The experiments were run on a single core of Intel(R) Xeon(R) Gold 6148 CPU @ 2.40GHz per experiment.

[3]Percentage of optimization instances where a restart was necessary to obtain a feasible solution in the optimization problem.



(a) The assembly order used in reality.

(b) Without optimization function.

(c) Using ψ_{neighbor}.

(d) Using ψ_{statics}.

Fig. 4: Sequences arising from the different stability criteria (dark parts are placed first, light parts at the end) with horizon length $n_h = 1$.

TABLE III: Average time usage and restart rate for different combinations of stability bounds ψ and maximum horizon length $n_{\max}$.

				Time [h]				
ψ	$n_{\max}$	Restarts[3] [%]	RRT [%]	Logic & $\mathcal{P}_{\text{Stab}}$	$\mathcal{P}_{\text{Pose}}$	$\mathcal{P}_{\text{Seq}}$	$\mathcal{P}_{\text{RRT}}$ & $\mathcal{P}_{\text{Full}}$	Total
statics	1	10.0	33.4	3.81	0.09	0.79	0.83	5.5
	2	22.3	64.3	2.87	0.07	2.54	1.40	6.9
	3	25.9	77.4	2.70	0.07	3.88	1.53	8.2
neighbors	1	17.9	42.3	0.02	0.12	1.03	1.21	2.4
	2	28.7	65.1	0.01	0.09	3.42	1.71	5.2
	3	29.2	75.1	0.01	0.07	3.95	1.70	5.7
	4	30.7	80.8	0.27	0.07	4.22	1.63	6.2

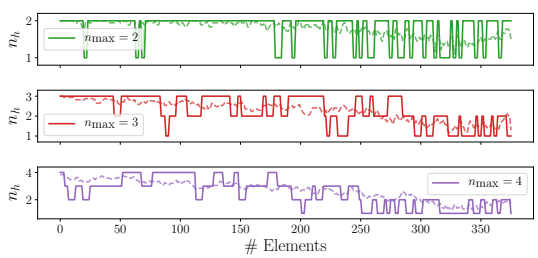


Fig. 5: Average (dashed) and exemplary (full) trajectories of horizon length n_h for various $n_{\max}$ through a full run.

without any restarts, the utility of the restarting scheme becomes critical when dealing with larger n_h.

- Comparison of the stability functions: The stability function imposes the order of the parts (Fig. 4). This is helpful for the feasibility of the following bounds in the MBTS, which in turn leads to a decrease in the necessary time for the *sequence* ($\mathcal{P}_{\text{Seq}}$) and *motion* ($\mathcal{P}_{\text{RRT}}$ & $\mathcal{P}_{\text{Full}}$) bounds. This comes with a stark trade-off of necessary computation time. In sum, the total computation times when using ψ_{statics} are approximately 1.5 to 2 times longer.
- The switching scheme for the adaptive horizon is aggressive in trying to get back to the maximum allowed horizon length. It can be seen in Fig. 5 that a longer horizon fails, and oscillatory behavior emerges between the short and the longer horizon length. This can potentially lead to wasted effort, which could be avoided by using a more nuanced switching scheme.

VI. DISCUSSION

A. Limitations

The limitations that arise in our demonstration can be grouped in two major areas: *i*) issues that prevent scaling of this approach to other, larger problem instances:

- Decomposing the long-horizon problem into a problem with multiple disjoint subgoals is not feasible for some problems that are more reliant on previous decisions of the TAMP problem. Similar to Model Predictive Control (MPC), the limited horizon approach we introduce here decreases the set of feasible solutions for future decisions. The stability-bound helps mitigate, but does not resolve possible challenges from this completely.
- Sampling-based approaches do not scale favourably over multiple agents in the naive compound-configuration-space formulation. Planning for multiple agents is still feasible and performant enough for the case of two agents, but scaling this approach as is to parallel construction using multiple agents on the same building is computationally inefficient.

and *ii*) caveats to the solutions that we introduced for the increase in robustness of the LGP approach:

- Using sampling-based motion planners offers a solution to the problem that optimizers fail in cluttered environments, but they come with their own set of problems, namely not being able to declare a problem as infeasible, and computing paths that might be feasible when ignoring system-uncertainties, but are actually infeasible.
- The introduced method of restarting the optimizer currently still suffers from a similar issue as the sampling-based planner: in a non-linear optimization problem it is, in general, not known if a solution can be found, or if a different action sequence should be followed.

B. Outlook

Solving the issues stated in Section VI-A would allow our approach to be scaled more efficiently to multiple agents, leading to more versatile construction processes. A more intelligent scheme for the determination of the subgoals should be an area of further research. The incorporation of future costs in the TAMP framework similar to the terminal set and the terminal cost in MPC might alleviate issues that could arise when choosing the subgoals for the decomposition improperly.

Neglecting the noise and dynamics arising in the real world makes many of the planned trajectories suboptimal or even infeasible in the real world. Incorporating measures akin to the stability function for the controllability would enable more realistic planning. Future work also targets incorporating a more structured sampling of the starting points, and a (better) stopping criterion, which have previously been investigated in optimization literature.

VII. Conclusion

We introduced a novel TAMP solver incorporating a limited horizon length and sampling-based planning methods in combination with novel stability analysis to solve complex, long-horizon TAMP problems in the construction domain robustly. Our work can be seen as a first step towards unifying assembly and construction planning, which in turn can enable co-design, i.e., the design is iteratively analyzed and improved in terms of geometric, kinematic, and static feasibility of the whole construction process.

We combine optimization and sampling-based methods to combine the strengths of both approaches. We show that restarts of the optimizer makes our framework more robust, and thus applicable to complex, long-horizon TAMP problems.

Acknowledgements

We thank Hans-Jakob Wagner and Long Nguyen for their help with the creation of the video, and for providing the model of the BUGA pavilion.

We thank the anonymous reviewers for the helpful comments regarding the presentation of the paper.

References

[1] UN Environment and International Energy Agency, "Towards a zero-emission, efficient, and resilient buildings and construction sector. global status report 2017," https://www.worldgbc.org/, 2017.

[2] M. Alvarez, H. Wagner, A. Groenewolt, O. Krieg, O. Kyjanek, L. Aldinger, S. Bechert, D. Sonntag, A. Menges, and J. Knippers, "The buga wood pavilion – integrative interdisciplinary advancements of digital timber architecture," in *Ubiquity and Autonomy - 39th ACADIA Conference 2019*, 10 2019, pp. 490–499.

[3] M. Toussaint and M. Lopes, "Multi-bound tree search for logic-geometric programming in cooperative manipulation domains," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 4044–4051.

[4] L. P. Kaelbling and T. Lozano-Pérez, "Hierarchical planning in the now," in *Workshops at the Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

[5] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling, "Ffrob: An efficient heuristic for task and motion planning," in *Algorithmic Foundations of Robotics XI*. Springer, 2015, pp. 179–195.

[6] H. M. Van, O. S. Oguz, Z. Zhou, and M. Toussaint, "Guided sequential manipulation planning using a hierarchical policy," RSS Workshop on Learning in Task and Motion Planning, 2020.

[7] F. Lagriffoul, N. T. Dantam, C. Garrett, A. Akbari, S. Srivastava, and L. E. Kavraki, "Platform-independent benchmarks for task and motion planning," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3765–3772, 2018.

[8] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," *Artificial Intelligence*, vol. 219, pp. 40–66, 2015.

[9] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein *et al.*, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine learning*, vol. 3, no. 1, pp. 1–122, 2011.

[10] M. Dogar, A. Spielberg, S. Baker, and D. Rus, "Multi-robot grasp planning for sequential assembly operations," *Autonomous Robots*, vol. 43, no. 3, pp. 649–664, 2019.

[11] D. Driess, O. S. Oguz, and M. Toussaint, "Hierarchical task and motion planning using logic-geometric programming (hlgp)," RSS Workshop on Robust Task and Motion Planning, 2019.

[12] H. Ardiny, S. Witwicki, and F. Mondada, "Are autonomous mobile robots able to take over construction? A review," *Int. J. Robot. Theory Appl.*, pp. 10–21, 2015.

[13] A. Adel, A. Thoma, M. Helmreich, F. Gramazio, and M. Kohler, "Design of robotically fabricated timber frame structures," in *Proceedings of the 38th Annual Conference of the Association for Computer Aided Design in Architecture (ACADIA).*, 2018.

[14] K. Dörfler, T. Sandy, M. Giftthaler, F. Gramazio, M. Kohler, and J. Buchli, *Mobile Robotic Brickwork*. Cham: Springer International Publishing, 2016, pp. 204–217. [Online]. Available: https://doi.org/10.1007/978-3-319-26378-6_15

[15] J. Willmann, F. Augugliaro, T. Cadalbert, R. D'Andrea, F. Gramazio, and M. Kohler, "Aerial robotic construction towards a new field of architectural research," *International journal of architectural computing*, vol. 10, no. 3, pp. 439–459, 2012.

[16] T. Bock, "Construction robotics," *Autonomous Robots*, vol. 22, no. 3, pp. 201–209, 2007.

[17] 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*, 2018.

[18] M. Toussaint, J.-S. Ha, and D. Driess, "Describing physics for physical reasoning: Force-based sequential manipulation planning," *IEEE Robotics and Automation Letters*, 2020.

[19] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, vol. 2. IEEE, 2000, pp. 995–1001.

[20] Z. Kingston, M. Moll, and L. E. Kavraki, "Sampling-based methods for motion planning with constraints," *Annual Review of Control, Robotics, and Autonomous Systems*, 2018.

[21] K. Hauser and J.-C. Latombe, "Integrating task and prm motion planning: Dealing with many infeasible motion planning queries," in *ICAPS09 Workshop on Bridging the Gap between Task and Motion Planning*, 2009.

[22] M. Toussaint, "KOMO: Newton methods for k-order Markov constrained motion problems," e-Print arXiv:1407.0414, 2014.