Combining Assembly Planning and Geometric Task Planning

Raphaël Lallement, Juan Cortés, Mamoun Gharbi, Alexandre Boeuf, Rachid Alami, Carmelo J Fernandez-Agüera, Iván Maza

Chapter 5.1
Combining Assembly Planning and Geometric Task Planning

Raphaël Lallement, Juan Cortés, Mamoun Gharbi, Alexandre Boeuf, Rachid Alami, Carmelo J. Fernandez-Agüera, Iván Maza

Abstract This chapter deals with the integration of different planners that solve problems from different domains in a common context. In particular, a joint solution for structure assembly planning, symbolic task planning and geometric planning is presented and analyzed in the construction of structures with a team of aerial robots equipped with on-board manipulators in places where the access is difficult. Geometric reasoning is present at different levels in our joint solution in order to reduce the computational complexity derived from the highly dimensional space due to the many degrees of freedom of the robots and the complexity of the tasks and also to produce more robust plans even for strongly intricate problems.

1 Introduction

The motivation for the work described in this chapter comes from context of the ARCAS European Project [1] funded by the European Commission. One of the goals of this project is to build a structure by using a team of aerial robotic manipulators (AROMAs). The practical interest of this system can be found in situations where it is required to build a structure in places with difficult access by conventional means. Actually, the problem of automatically finding a proper place for the construction of the structure is also addressed in the project.

Assembly planning is the process of creating a detailed assembly plan to craft a whole product from separate parts by taking into account the final product geometry, available resources to manufacture that product, fixture design, feeder and tool descriptions, etc. In this context, different planning problems such as structure assembly planning, task planning and motion planning have to be solved in a highly dimensional space due to the many degrees of freedom of the robots and the complexity of the tasks. To have simple yet interesting structures to assemble the project

[1] http://www.arcas-project.eu

focused on structures made of bars. An example can be seen in Fig. 1. The simplicity comes from the clipping mechanism: when two pieces are brought together they clip ensuring a strong link. On the other hand the complexity comes from the need for cooperative transport of certain long bars requiring two (or more) AROMAs to strongly cooperate. Moreover the robots are using arms which must be compliant to avoid any problems. To carry out the assembly of the structures, the complete system must exhibit a set of properties:multi-robot plans, cooperative transport, perception and localization and visual servoing.

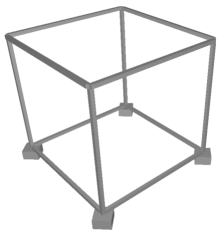

Fig. 1 An example of the type of structures considered in the ARCAS project. There are 12 parts composing a cube, but there also exist a variant with four additional diagonal bars for a total of 16 parts.

Assembly, symbolic and motion planning traditionally belong to decoupled different research areas and have been widely studied separately in the last decades. However, there is a growing interest on how to properly connect these planners or how to design coupled planners.

This chapter proposes a joint solution for these planning problems which embeds geometric reasoning at different levels in order to reduce the computational complexity. The chapter is structured as follows. Section 2 presents related work on the connection of assembly planners to task planners. The assembly planning problem covers three main assembly subproblems: sequence planning, line balancing, and path planning. This chapter is focused on a single AROMA, and hence, the line balancing subproblem related to the allocation of assembly operations to robots is out of the scope of our work. The approaches followed to solve the other subproblems are described in Section 3. Section 4 describes in more detail the proposed architecture and how each component relates to each other. It also defines the information exchanged between the planner and introduces the heuristic used during the experiments to guide Assembly Planning. Section 5 describes how the approach works for a team of AROMAs whereas Section 6 presents how the symbolic plans can be refined to deal with intricate constraints. Finally, the conclusions in Section 7 close the chapter.

2 Related Work

Efficient assembly plans can reduce time and costs significantly. The assembly planning problem has been shown to be an NP-complete problem [18] and covers two main assembly subproblems in a single robot context: sequence planning and path planning.

The Assembly Sequence Planning (ASP) problem concerns with finding a sequence of collision-free operations that bring the assembly parts together, having given the geometry of the final structure. A systematic overview on the ASP is presented in [15], which includes a survey of the elements of sequence planning, such as finding a feasible sequence, determining an optimal sequence according to one or more operational criteria, representing the space of feasible assembly sequences in different ways, applying search and optimization algorithms, and satisfying precedence constraints existing between subassemblies. This problem is addressed in Section 3.1.

A taxonomy for assembly and task planning is described in [12]. However, the taxonomy presents both planners in a decoupled manner. The problems that an autonomous robotic assembly system must tackle both regarding assembly planning and task planning are summarized in [30]. This paper also reviews the approaches that use Petri Nets as a formalism to develop the corresponding planners, but the coupling between both is not described.

In [23], the output of an assembly planner is used directly as input to a symbolic planner that decomposes it. Once given the assembly plan, the system allows them to complete motion planning in real time. However, since there is no feedback, the system must assume that the given assembly plan is compatible with geometric restrictions. Manipulation primitive nets are introduced in [32] to deal with uncertainties occurring during assembly task execution.

In [19], the assembly planner computes a blueprint by reasoning about the holes used to attach the parts together. It then sends the blueprint to the symbolic planner using an object-oriented language which is then translated into a PDDL problem and solved using the Fast-Forward solver. This builds a set of actions which is then dispatched in the group with sometimes coordinated (multi-robot) manipulations. So the geometry is only taken in account when computing the blueprint, not to ensure the feasibility for the coordinated motion: a robot could block or be blocked by others.

Finally, [14] uses a 3-layer architecture: an assembly planner generates the assembly graph and finds a good sequence. Then a two-stage executive layer turns the task tree into instances which in turns are transformed into robot behaviours. The last step is the execution on the different robots, which is controlled by the behaviours layer. The system uses motion planning and an exception-handling mechanism to ensure its robustness. Although it will not be as optimal as a planning-based system, the authors claim it to be significantly faster.

Dealing with and merging symbolic and geometric constraints when planning is a growing field that has been a focus to a number of researchers over the few last years. Different approaches have been proposed like [29] where the search at

geometric level is guided by the symbolic planner. Other approaches focused on different aspects like [28] where a *placement graph* is built in other to apply a Satisfiability Modulo Theories solver on the actions represented in the graph. Similarly, [9] builds a conditional-reachability graph representing different configurations reachable by the robot (with or without object in hand) and the FastForward algorithm is used to guide the search and find solutions in the graph.

In [7] and [6], the authors use *semantic attachments*, which are external procedures called by the symbolic planner to assess the feasibility of a symbolic action at the geometric level (e.g. if a collision free path exists between 2 positions). [17] extends this approach by adding a geometric backtracking: in order to assess the feasibility of an action, the geometric planner can change the geometric choices made for the previous actions.

Reference [16] uses a goal-driven task planner that enables calls to external procedures during the planning phase. In case of failure, this procedure provides a reason of the failure to the planner and enables it to find an alternative plan. The idea of using failure reasoning to guide symbolic planning is also stressed out in [8] where a causal reasoner is used to find a symbolic plan, then a motion planner tries to solve the underlying motion problem, and if the latter fails, a temporal constraint is added to the symbolic domain and the causal reasoner tries to find a new plan. [31] uses off-the-shelf planners and formulates a novel representational abstraction enabling to combine them: each action in the symbolic plan can have multiple instantiations at geometric level corresponding to the different geometrical choices possible.

In [20] a task-plan is computed, and interval bounds which are constraints on object/robot positions are used to prune out geometric choices. Another approach, proposed in [22], is to formalise the problem as geometric constraints between agents and objects, and to use a Constraint Satisfaction Problem solver to find the solution. Reference [24] addresses the combination of a symbolic hierarchical task network planner and a constraint satisfaction solver for the vehicle routing problem in a multi-robot context for structure assembly operations. Each planner has its own problem domain and search space, and the article describes how both planners interact in a loop sharing information in order to improve the cost of the solutions.

We have proposed various contributions to the problem of combining symbolic and geometric reasoning in order to produce pertinent and feasible robot plans. In Asymov [13, 2] we essentially proposed a principled way to link the two planners thanks to a geometric level able to tackle the so-called “manipulation planning problem” [1] and that allows to explicitly take into account the topological changes occurring, in the configuration space, when a robot grabs or releases an object. Asymov provided a well founded translation of pick and place actions (and similar actions) into ’transit’ and ’transfer’ motion planning requests even in multi-object and multi-robot contexts.

More recently we focused on a complementary approach [5, 3]: exploiting the capacity of the Hierarchical Task Network (HTN) [27] techniques to encode domain knowledge, and developing a geometric motion planner capable of planning actions with several levels of abstraction opening to more elaborate action instantiations. Such a combination provides several key features: a clean interface which corre-

sponds to the anchoring problem and allows to better exhibit and master the links between the incremental processes of producing the symbolic plan and its geometric counterpart.

In [5] we presented a geometric backtracking algorithm which allows to reconsider the previous geometric choices by trying different alternatives to the previously computed actions and tests the validity of this alternatives by computing their geometric effects. In [11], a more elaborate management of the ramification problem is proposed allowing the integrated planner to exploit: (1) the ability of the geometric planner to provide several instantiations of the same action, (2) the ability to encode domain expert knowledge at both levels, (3) well-informed cost estimation of actions.

3 Description of the Planners

The overall planning approach is composed of three planners: (1) an assembly sequence planner that computes the assembly sequence from the assembled structure, (2) the symbolic planner, HATP (Hierarchical Agent-based Task Planner), is responsible for the task allocation, and finally (3) a motion planner, to compute the trajectories for each action in the symbolic plan. These planners are described in this section.

3.1 Assembly Sequence Planner

Figure 2 shows the architecture of the assembly planner [25] developed to take into account the effect of external forces such as gravity and physical interactions between the parts of the structure to be built. It uses simulations to predict the stability of partially assembled structures and applies that information to expand the graph of possible plans.

The assembly planner is built on top of two main components: A classical planning engine and a physics simulation engine.

- Planning engine: a generic planning library has been developed. The library allows transparent substitution of simple search strategies such as depth-first or breadth-first. It provides support for heuristics for more advanced strategies as well. At a technical level, the framework takes the form of a set of template classes that can be inherited or instantiated with specific policies, and abstracts the generic process of planning. A higher level component is responsible for writing a dynamic graph tree, selecting a search strategy, and representing states and actions that describe the application specific problem set. In the case of assembly planner, an action would be the removal of a piece from the structure along a discrete direction and a state would correspond to a partial assembly of the structure that is stable and self-sustaining.

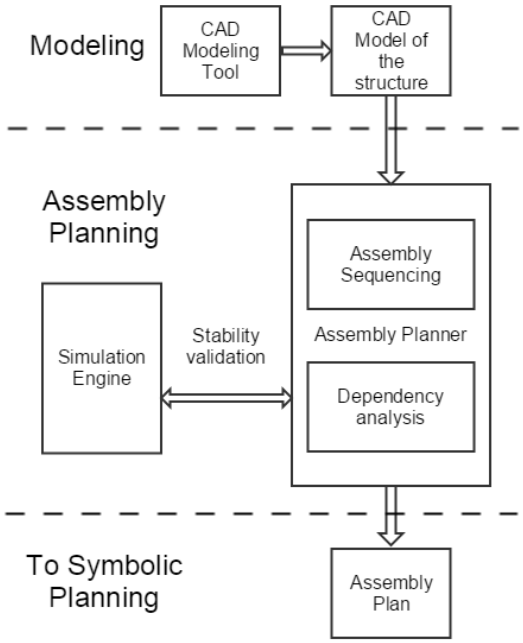

Fig. 2 Architecture of the assembly planner developed to take into account the effect of external forces such as gravity and physical interactions between the parts of the structure to be built

- Physics engine: This component is built using the Bullet Physics library[2]. It is applied to compute whether a subassembly is stable (i.e. it is self-sustaining and can remain indefinitely in the given configuration without external support). When given a set of pieces and their locations in the structure, it performs a rigid body simulation on them and checks the results. If after a fixed period, all the pieces of the structure remain in their starting configuration (position and orientation), the subassembly is considered stable.

The planner actually solves the problem of disassembly, and then returns the reverse plan as a valid assembly plan. It uses a best-first search strategy, based on heuristics that can be refined for successive iterations of re-planning. The search graph used for planning is given in implicit form. This avoids the need to precompute and store the whole tree in memory. In order to get the possible actions that can be performed at a given state, the planner invokes the simulator and retrieves a list of all the pieces that can be safely removed from the remaining set.

[2] http://bulletphysics.org

As a final step, the planner will try to combine successive actions into parallel operations that can be executed at the same time by separate agents. The engine tests consecutive operations combined into a single action. This process also lets the engine compute conditional relations between operations. Let it be two consecutive operations A, the first, and B, the latter. If the order of execution of A and B can be reversed, then they can also be carried out in parallel. If the order can not be reversed, then operation A is considered to be a precondition for operation B. Note that preconditions computed in this manner are actually a super set of the absolute precondition relations between all actions (i.e. there may be alternative plans where B can be performed before A). However, this technique guarantees that all variants of the plan resulting from recombining the operations according to these relations are also stable. This property can be highly valuable for integration with planners in other domains.

3.2 Symbolic Planner

A symbolic planner is required to get the actual set of actions to execute the assembly plan. It will compute the best sequence of actions to fulfill the plan for a given set of agents and their properties (payload, actuator, and so on).

Our system uses Hierarchical Task-Network (HTN) planning [10], for its expressiveness and efficiency. An HTN planner computes the symbolic plan from a domain that is a description of the different possible ways to solve the problem, and it is provided by a domain expert. The domain is composed of tasks, and task can be of one of two types: *action* or *method*. An action is a task that the system can actually achieve (such as go to, pick, place, and so on). On the other hand, a method is composed of several sets of partially-ordered tasks (again actions or methods), each set represents a way to solve the method at hand. The domain is composed of several methods encapsulating other methods and actions, hence creating a hierarchy.

The planning goal is expressed as a task to accomplish. The principle of HTN planning is to decompose this top-level task into one of its possible decompositions (i.e. as set of partially ordered tasks). This process goes on until all tasks are decomposed into actions. In addition to this process, there is also a mechanism of variable binding: it is indeed possible to let some task parameter as free variable, the planning process will then try different values and keep the best one. Finally the choices for which decomposition and which value for a variable are done randomly and it creates backtrack points. The most famous HTN planner is SHOP2 [26].

Our implementation is called HATP [21], which stands for Hierarchical Agent-based Task Planner. It is a total-order HTN, which means that we can have the current symbolic state anytime during planning. One of HATP main characteristic is its object-oriented and user-friendly language which ease the development of new domain. It uses a state variable representation, and every object in the environment is represented with an *entity*. The language is rich and contains many features such as set manipulation. Please refer to [4] for more details.

Our planner is "agent-based" because agents are special entities used to compute streams of actions in addition to the usual solution (tree) decomposition. Each stream represents an agent. It contains its actions and causal links to order them, and to link them with the other agents stream.

HATP is implemented in C++ and special care was taken when designing the internal structures to ensure that it is easy to extend the planner with new functionalities. The most notable one is the tight integration with a Geometric Task Planner.

3.3 Interleaved Task and Motion Planning

GTP (Geometric Task Planner) encapsulates a motion planner, including a level of abstraction. For instance, it allows to plan a 'placeReachable' action, which is a combination of fast trajectory, approach trajectory, ungrasp and escape trajectory.

Using this planner, we developed a system that can check the geometric feasibility of symbolic plan *as it is built* [3]. The principle is that, when the symbolic planner adds an action to the symbolic plan, its geometric counterpart is tried in the current geometric state. If the geometric action is feasible, the planning process continues, otherwise a backtrack is triggered: either change the symbolic plan or try another solution for one of the previous geometric actions.

In addition to the symbolic plan, a geometric counterpart is built and maintained to ensure that the current geometric state matches the current symbolic state. To update the symbolic state, some literals are computed by GTP (e.g. an object *isOn* a table, and so on) and sent to HATP to update the symbolic state. This ensures the consistency of the symbolic state and allows to tackle the ramification problem: when an unexpected side effect occurs, it is detected. We also implemented some other functionalities, as further described in [11].

4 Integration of Symbolic-Geometric Planning with Assembly Planning

Integrating the assembly planner and the Symbolic-Geometric Planner (SGP) together, we were able to compute symbolic plans with associated trajectories that could be executed on the system. However, there was a combinatorial problem, since the assembly plan was given as a dependency tree where each part would specify which other part must be assembled before the symbolic planner had to find an actual assembly sequence. For instance in the cube structure in Fig. 1, the four first bars that compose the first level can be assembled in $4! = 24$ different orders. With the two other levels. this number explodes. Indeed, the worst case scenario is $n!$ where n is the number of parts in the structure. The proposed solution is to have a tighter integration between SGP and the assembly planner.

4.1 Principles

As it has been explained in the previous section, the assembly planner takes as input a model of the structure as it should be assembled, and computes the assembly sequence using an assembly-by-disassembly approach: it tries all the ways to disassemble the structure to guess the assembly sequence and the dependencies between the parts. To improve this search, two additional tools are used, a physics engine to ensure the stability and self-sustainability of the structure at each step, and a heuristic search. The heuristic used consists in computing the risk of collision in order to find an assembly sequence with less chances of collisions (self-collisions between the parts or collisions with the obstacles). To obtain this information, bounding volumes are placed around the objects and collisions between those volumes are used. Once a solution is found, the assembly sequence is sent to HATP along with the dependencies between the parts. Thanks to the heuristic search, this assembly plan is supposedly the most likely to succeed.

It is possible, when solving the problems individually, that solutions to one problem are incompatible with the other domains (e.g. it might not be possible to compute paths for an otherwise correct assembly plan). Thus, it is important to keep a loop of information between the planners. This loop can also hint each planner into the right direction to get better plans. For example, information from the symbolic planner can help the assembly planner choose a sequence of operations with better odds of succeeding on later phases of planning. At the other end of the system, a complete assembly plan can save the symbolic and geometric planners a good deal of computation, because it provides them with a smaller search space, with far less degrees of freedom.

The approach followed in this work, illustrated in Fig. 3, is to feed a structure to the assembly planner, which will generate a single assembly plan, and to feed this plan to the symbolic planner, which in turn will invoke the geometric planner as needed. The assembly planner is guided by an heuristic that can be modified with information from HATP. When an assembly plan results unfeasible during HATP or GTP phases, information about the causes of the failure can be fed back to AP to improve the heuristic and have better chances of finding a successful assembly plan.

We decided to send only one plan at a time, in order to reduce the search space HATP has to work with. Sending an explicit graph representation (e.g. an AND/OR graph) of all plans is impractical for large structures in general. The number of possible states follows a growth rate of the order of $\mathcal{O}(n!)$ with n being the number of pieces in the structure. Hence, a simple structure as the cube used in this work (16 pieces) can easily get a graph with too much nodes to be able to compute it, reaching the trillions. In addition to that, evaluating each plan has an important cost, because we have to try it in both GTP and HATP, which can be time consuming.

Our approach not only reduces the search space for both planners, it also allows each of them to address the part of the problem it fits better. For instance, GTP would have a hard time figuring out stability issues within a structure, but the assembly planner is specifically designed for that. Complementary, the assembly planner can only solve linear straight paths for assembling each piece, but GTP can find more

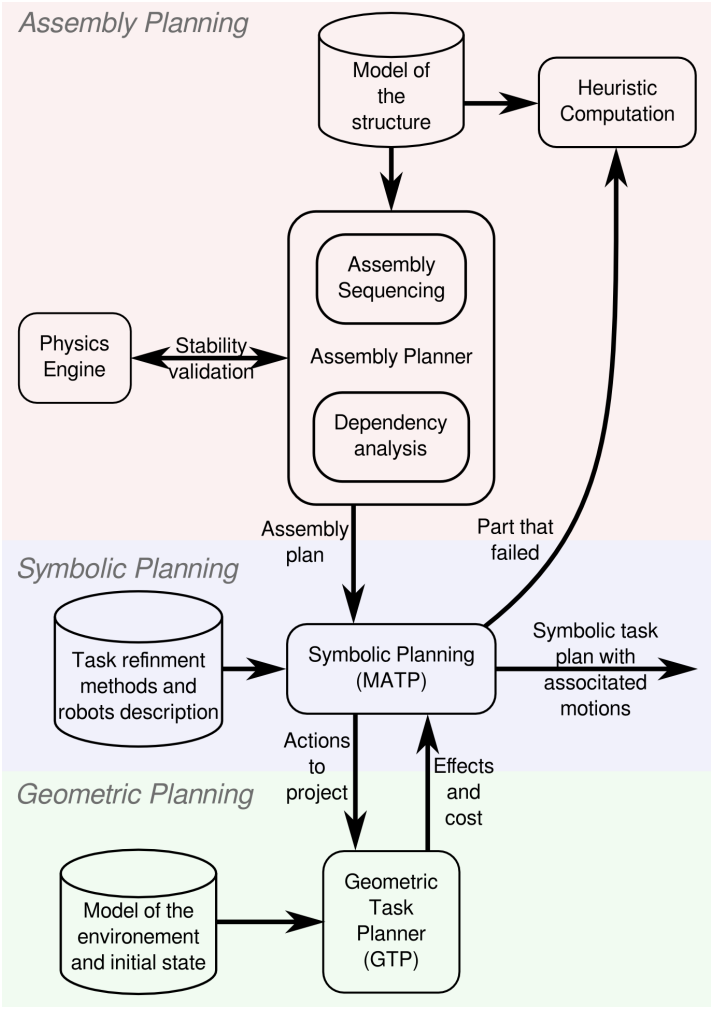

Fig. 3 Complete planning architecture including assembly planning, symbolic planning and geometric task planning. The assembly planner sends the assembly plan computed thanks to the heuristic, then HATP plans the symbolic action and assesses their feasibility at geometric level thanks GTP. If the symbolic plan can not be found or if it is not geometrically feasible, the part that prevents it is sent to the heuristic, which is updated, and a new assembly plan is computed.

complex solutions if they exist. While geometric analysis might be used to predict some precedence relationships between assembly operations that must be fulfilled, it misses a family of limitations of mechanical nature such as balance and friction issues (where extracting a piece in a given direction might drag other pieces out of place). The case shown in Fig. 4 demonstrates a structure where pure geometrical reasoning can not anticipate the cause of instability, since the behaviour depends on physical properties of the bodies (i.e. their masses) and there is no direct contact

between the critical pieces. Since AP performs rigid body simulations on the resulting subassemblies, the problem would be detected early, and would never reach the HATP or GTP phases, thus saving a lot of computation time that would be wasted otherwise. Note that this is different from physics based geometric planning, since we are studying a stability condition inherent to the structure itself and independent from the path that agents follow while actually executing the assembly sequence. In a similar fashion, HATP can handle multi-agent domains and allocate tasks in parallel easily, while adding this to the assembly planner would increase complexity and computation time.

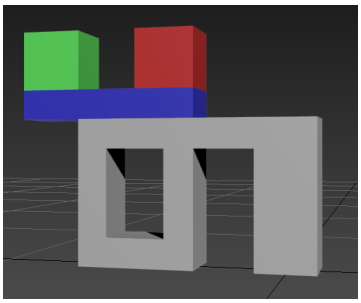

Fig. 4 Example of structure with potential instability. The stability depends on the order of operations where there is no contact between critical pieces. If the red block was to be removed before the green one, the structure would become unstable and the blue block would fall.

4.2 Formalisation

A planning session starts with HATP requesting an assembly plan from the assembly planner for a given structure. Such request would simply contain the name of the structure that the assembly planner needs to analyze, and a request ID that must be the same for further requests in the same session.

The assembly planner will retrieve the current world state (mainly obstacles) from the same model that GTP uses, and a description of the structure as it should be (from a CAD 3D model), and generates an assembly plan for it. Then, it will answer with a response in the form of a tuple $PR = \langle S, D \rangle$, where:

- $S = \{s_1, ..., s_n\}$ is the ordered sequence of assembly operations s_i that must be performed to assemble the structure. Each element s_i describes the type of operation and the pieces involved. At the moment, only putting the pieces in place is supported as a type of operation, but this can be generalized to other operations as screwing them in or securing them in other ways. An operation has the form $s_i = (id, type, pieces...)$, where the first element is a number to identify the mo-

ment when the operations should be done (the smaller the number the earlier it must be done), then comes the type of operation, and finally the list of pieces.
- $D = \{d_1, ..., d_m\}$ is a set of dependencies between assembly operations. Each element d_j is a tuple made of the target operation and the set of all other operations that must necessarily be performed before that one, with $d_j = \langle t_l, p_k \rangle$ which reads as the set of operations p_k that must be achieved before the operation t_l, where t_l is just the identifier of an operation s_l. The set p_k contains the list of identifiers for operations from S.

Making precedence relationships between operations explicit allows the symbolic planner to perform more than one operation of the sequence at the same time in multi-agent environments, with one or more assembly operations starting before all previous operations in the sequence are finished.

See below an example of such plan, for the cube in Fig. 1, where in order to assemble the vertical bar we need the four operations that put the horizontal (bottom) bars:

$$PR = \langle (0, place, horizontal\_bottom\_bar\_0), \ldots, (4, place, vertical\_bar\_5), \ldots \\ \ldots, \langle 4, 0, 1, 2, 3 \rangle, \ldots \rangle$$

When HATP starts, it requests the assembly plan and upon receiving it, starts the decomposition. The first step is to detect which operation to achieve first and to do so it uses the identifier, starting at 1 and going up to the number n of operations. For each operation an *Assemble* task is called: it detects the type of operation and execute it for all the parts listed. Then it uses the dependencies D to add causal links and in multi-agent scenario uses those links to parallelize some operations.

4.3 Heuristics

The assembly planning search space is tremendously high even for structures with a small number of pieces (ten to twenty), so we use a heuristics based best first to navigate it efficiently. While choosing heuristics to guide the assembly planning process, the goal is to reach an assembly plan with high probability of being successful during the symbolic and geometric planning phases. By feeding information from the later phases back into the assembly planner's heuristic, the odds are improved but both domains remain decoupled.

Different heuristics can be used in assembly planning for this purpose. We used a measurement of the free space available around each piece in the directions of assembly, in order to give preference to the least occluded one. The reasoning behind this heuristic is that occluded pieces are more likely to introduce difficulties during assembly by posing obstacles to the agents. For any given piece, its geometry is projected along various directions (possible assembly directions) and each collision

detected will increase the cost associated to the removal of that piece. If replanning is needed, these costs can be increased according to information received from the symbolic planner. In this fashion, occlusion would be more costly if the associated piece is directly linked with the reason why the first plan was discarded.

For future work, other heuristics such as using a bounding sphere of the size of the agents may be tested.

5 Multi-AROMA Planning

The current implementation of GTP has a limitation: it does not allow to plan trajectories in parallel, it is only possible to plan trajectories where no other robot move or cooperatively transport one of the parts (which is considered by GTP as one "agent" acting, not two individual trajectories). The method used for trajectory planning of individual AROMAs will be described in Chapter **??**. To cope with this limitation, we generate fully sequential symbolic plans that can later be transformed into plans with parallelism thanks to a post-process, as long as the dependencies between the parts are respected. The actions that GTP can plan are: goto, pick, place and pickAndPlace. The latter is more efficient than a pick followed by a place because the different step to compute the trajectories, grasp and position are interleaved. (It is still important to allow separate pick and place if we need to add actions in between.) In addition to those actions with a geometric counterpart we have a monitor action, which triggers in the supervisor a visual servoing low-level task. It could be interesting to plan it in GTP, so the position of the monitoring UAV is taken into account when planning the others trajectories, but again GTP can not plan actions in parallel. Consequently, there is no point in planning the monitoring trajectory, so in our current implementation it is just considered a symbolic task during the planning process. Figure 5 presents an extract from a plan where a monitoring task must be carried out in parallel of an assembly task. When the robot must place the part, the action is actually called PlaceInSupport because of the domain representation we use when HATP and GTP communicate to assemble parts. The idea is that HATP needs a way to tell GTP where to place the bars, and we refer to the end position as "supports". Those supports are extracted from the final configuration of each part in the assembled structured (from the same model the assembly planner uses). In this particular example, the interesting points are the causal links, they force the monitoring task to last as long as the assembly task (place in support).

Figure 6 shows another extract from a plan, this time however it is focused on the action necessary to execute a cooperative transport. Our strategy is to elect a master and create a group, each other AROMA necessary is added to the group. To assign all the AROMAs in the group (master and others alike) we use the payload of each AROMA and compare the sum to the masses of the parts to transport. However if we only limit ourselves to this comparison, we will end up trying many groups just by changing the order of the AROMAs in the group (for instance a group with UAV1 and UAV2, and another UAV2 and UAV1, two different groups for HATP, but

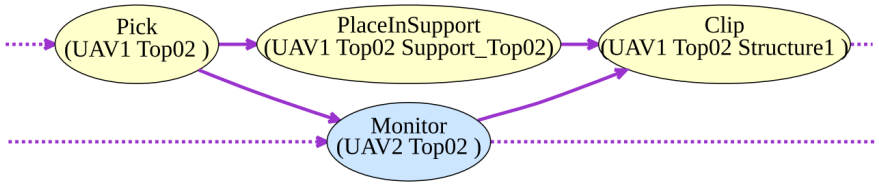

Fig. 5 Extract of a plan with a monitoring task. The AROMA UAV1 picks and places the bar Top02, while the UAV2 monitors this assembly.

with the similar meaning). Testing all the possible orders is useless since the order of the AROMAs in the group does not change the efficiency of the group. Moreover, it represents a very big number of attempts: $N!$ in fact, where N is the number of AROMAs in the problem. To address this issue, we have a mechanism that stores the group that were tested on each part, and prevents to try the same group with a different order on the same piece.

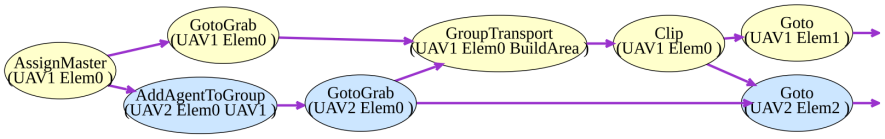

Fig. 6 Extract of a plan with a cooperative transport. In order for a group of AROMAs to cooperatively transport a bar we must assign them into the group, then they must all go to the part and grab it, then the group transports the bar. When it is finally clipped to the structure only then the AROMAs are released and can go do other actions.

6 Refining Symbolic Plans and Dealing with intricate constraints

We illustrate in Fig. 7 the interplay between the symbolic planner and GTP to build a valid plan to assemble two simple structures: the small green and the big blue structures. As the plan is computed, it can fail because geometry constraints not seen by the high-level planner prevent an action (cluttered space, or any other problem) or it can be due to a symbolic choice that leads to impossible execution. During the planning process both “green then blue” and “blue then green” orders are tried, however, only one is feasible. Without geometric projection, there would be no difference at the symbolic level, while in fact, if the big (blue) structure is assembled first, then the small (green) one cannot be assembled because, in this specific con-

text, the robot can not reach the position to place the last horizontal bar. So, only the order depicted in the picture leads to a possible plan.

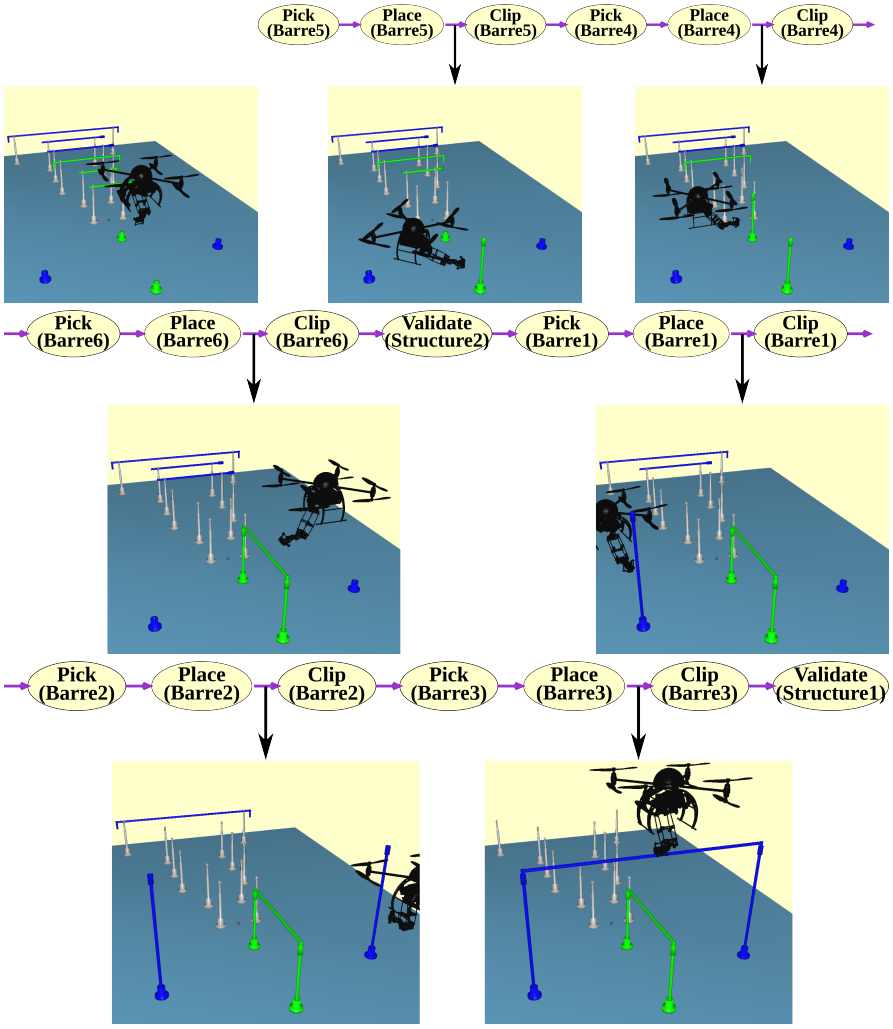

Fig. 7 Action stream and geometric states for the assembly of two structures. The AROMA assembles first the small green structures then proceeds with the big blue. Only this order is feasible because the AROMA must hold the bars in their middle and the long blue horizontal bar prevents it to place the last bar of the small green structure. Please notice that the actions are called `Pick` and `Place` as two separate actions but this is done only for a matter of clarity for the picture. The black arrows are pointing at the geometric state corresponding to the symbolic state.

7 Conclusions

This chapter has presented the algorithms used to solve the cooperative assembly planning problem. All the planners presented have been validated in simulation. The first planner, and the first step to solve the problem at stake, is the assembly planner. It builds an assembly plan that describes how the assembly must be done. However, it does not describe how to carry the assembly out using the available agents. To face this particular issue, we propose an approach based on combining several planning techniques to enrich the planning capabilities. The task planner is based on HTN (for its powerfulness and expressiveness) combined with an object-oriented language to ease the development of new domains. The domain model is such that sub-assemblies, multi-AROMA tasks and heavy parallelism are possible. Thanks to GTP, the geometric feasibility of the actions is ensured, and thanks to the physics engine from the assembly planner, the stability of the structure is guaranteed at each step of the assembly. The symbolic planner adds monitoring actions to increase the chances of success when actually executing the plan. The symbolic solution plan has a motion associated to each action, but is completely sequential. However, it is possible to have parallelism in the plan thanks to a post process as long as the dependencies between the parts are respected.

References

1. Rachid Alami, Thierry Simeon, and Jean-Paul Laumond. A geometrical approach to planning manipulation tasks. the case of discrete placements and grasps. In *The fifth international symposium on Robotics research*, pages 453–463. MIT Press, 1990.
2. Stéphane Cambon, Rachid Alami, and Fabien Gravot. A hybrid approach to intricate motion, manipulation and task planning. *I. J. Robotics Res.*, 28(1):104–126, 2009.
3. Lavindra De Silva, Mamoun Gharbi, Amit Kumar Pandey, and Rachid Alami. A new approach to combined symbolic-geometric backtracking in the context of human–robot interaction. In *IEEE international conference on robotics and automation (ICRA)*, 2014.
4. Lavindra de Silva, Raphaël Lallement, and Rachid Alami. The HATP hierarchical planner: Formalisation and an initial study of its usability and practicality. In *International Conference on Intelligent Robots and Systems*, 2015.
5. Lavindra de Silva, Amit Kumar Pandey, Mamoun Gharbi, and Rachid Alami. Towards combining HTN planning and geometric task planning. *CoRR*, abs/1307.1482, 2013.
6. Christian Dornhege, Patrick Eyerich, Thomas Keller, Michael Brenner, and Bernhard Nebel. Integrating task and motion planning using semantic attachments. In *Bridging the Gap Between Task and Motion Planning*, 2010.
7. Christian Dornhege, Andreas Hertle, and Bernhard Nebel. Lazy evaluation and subsumption caching for search-based integrated task and motion planning. In *IROS workshop on AI-based robotics*, 2013.
8. Esra Erdem, Kadir Haspalamutgil, Can Palaz, Volkan Patoglu, and Tansel Uras. Combining high-level causal reasoning with low-level geometric reasoning and motion planning for robotic manipulation. In *Robotics and Automation (ICRA), International Conference on*. IEEE, 2011.
9. Caelan Reed Garrett, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. FFRob: An efficient heuristic for task and motion planning. In *International Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2014.
10. Malik Ghallab, Dana Nau, and Paolo Traverso. *Automated Planning: Theory and Practice*. Elsevier, 2004.
11. Mamoun Gharbi, Raphaël Lallement, and Rachid Alami. Combining symbolic and geometric planning to synthesize human-aware plans: toward more efficient combined search. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2015, Hamburg, Germany, September 28 - October 2, 2015*, pages 6360–6365, 2015.
12. S. Gottschlich, C. Ramos, and D. Lyons. Assembly and task planning: a taxonomy. *Robotics Automation Magazine, IEEE*, 1(3):4–12, Sept 1994.
13. Fabien Gravot, Stephane Cambon, and Rachid Alami. aSyMov: a planner that deals with intricate symbolic and geometric problems. In *Robotics Research*, pages 100–110. Springer, 2005.
14. Frederik Heger and Sanjiv Singh. Robust robotic assembly through contingencies, plan repair and re-planning. In *International Conference on Robotics and Automation*, 2010.
15. P. Jiménez. Survey on assembly sequencing: a combinatorial and geometrical perspective. *Journal of Intelligent Manufacturing*, 24(2):235–250, apr 2013.
16. Leslie Pack Kaelbling and Tomás Lozano-Pérez. Hierarchical task and motion planning in the now. In *Robotics and Automation (ICRA), International Conference on*, pages 1470–1477. IEEE, 2011.
17. Lars Karlsson, Julien Bidot, Fabien Lagriffoul, Alessandro Saffiotti, Ulrich Hillenbrand, and Florian Schmidt. Combining task and path planning for a humanoid two-arm robotic system. In *Proceedings of TAMPRA: Combining Task and Motion Planning for Real-World Applications (ICAPS workshop)*, pages 13–20. Citeseer, 2012.
18. Lydia Kavraki, Jean-Claude Latombe, and Randall H. Wilson. On the complexity of assembly partitioning. *Information Processing Letters*, 48(5):229 – 235, 1993.
19. Ross A. Knepper, Todd Layton, John Romanishin, and Daniela Rus. Ikeabot: An autonomous multi-robot coordinated furniture assembly system. In *International Conference on Robotics and Automation*, pages 855–862, 2013.

20. Fabien Lagriffoul, Dimitar Dimitrov, Alessandro Saffiotti, and Lars Karlsson. Constraint propagation on interval bounds for dealing with geometric backtracking. In *Intelligent Robots and Systems (IROS), RSJ International Conference on*, pages 957–964. IEEE, 2012.
21. Raphaël Lallement, Lavindra de Silva, and Rachid Alami. HATP: An HTN planner for robotics. In *2nd ICAPS Workshop on Planning and Robotics*, pages 20–27, 2014.
22. Tomás Lozano-Pérez and Leslie Pack Kaelbling. A constraint-based method for solving sequential manipulation planning problems. *IROS*, 2014.
23. H. Mosemann and F.M. Wahl. Automatic decomposition of planned assembly sequences into skill primitives. *Robotics and Automation, IEEE Transactions on*, 17(5):709–718, Oct 2001.
24. Jorge Muñoz-Morera, Francisco Alarcon, Ivan Maza, and Anibal Ollero. Combining a hierarchical task network planner with a constraint satisfaction solver for assembly operations involving routing problems in a multi-robot context. *International Journal of Advanced Robotic Systems*, 15(3):1–13, June 2018.
25. Jorge Muñoz-Morera, Ivan Maza, Fernando Caballero, and Anibal Ollero. Architecture for the automatic generation of plans for multiple UAS from a generic mission description. *Journal of Intelligent & Robotic Systems*, 84(1):493–509, 2016.
26. Dana Nau, Tsz-Chiu Au, Okhtay Ilghami, Ugur Kuter, J. William Murdock, Dan Wu, and Fusun Yaman. SHOP2: An HTN planning system. *JAIR*, 20(1):379–404, 2003.
27. Dana Nau, Malik Ghallab, and Paolo Traverso. *Automated Planning: Theory & Practice*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.
28. Srinivas Nedunuri, Sailesh Prabhu, Mark Moll, Swarat Chaudhuri, and Lydia E Kavraki. Smt-based synthesis of integrated task and motion plans from plan outlines. In *IEEE international conference on robotics and automation (ICRA)*, 2014.
29. Erion Plaku and Gregory D Hager. Sampling-based motion and symbolic action planning with geometric and differential constraints. In *Robotics and Automation (ICRA), International Conference on*, pages 5002–5008. IEEE, 2010.
30. J. Rosell. Assembly and task planning using petri nets: A survey. In *Proceedings of the I MECH E Part B journal of engineering manufacture*, volume 218, pages 987–994, 2004.
31. Siddharth Srivastava, Eugene Fang, Lorenzo Riano, Rohan Chitnis, Stuart Russell, and Pieter Abbeel. Combined task and motion planning through an extensible planner-independent interface layer. *IEEE international conference on robotics and automation (ICRA)*, 2014.
32. Ulrike Thomas and Friedrich M. Wahl. Assembly planning and task planning - two prerequisites for automated robot programming. In Daniel Schütz and Friedrich M. Wahl, editors, *Robotic Systems for Handling and Assembly*, volume 67 of *Springer Tracts in Advanced Robotics*, pages 333–354. Springer Berlin Heidelberg, 2011.