Iterative Rule Generation for Assembly Sequence Planning

Ismael Rodríguez[1,2], Korbinian Nottensteiner[1], Daniel Leidner[1], Michael Kaßecker[1], Freek Stulp[1] and Alin Albu-Schäffer[1,2]

***Abstract*— In future manufacturing scenarios, one of the most desired features is to automatically assemble customized products. In addition, robotic systems are favored if they adapt to individual products without increasing the total production time. The framework developed in this work provides a planning system that is able to generate a sequence of robot actions for the assembly of customizable products from a modular construction set. We propose an algorithm that generates symbolic rules from constraints that emerge during simulation. It automatically detects which action sequences fail and exploits this information to improve the planning effort over time.**

I. Introduction

In mass production, customization of products is trending. A commonly requested feature is quickly adaptable production lines without undergoing adverse effects on setup and production times. A major improvement is achievable by algorithmic creation of assembly plans for individually customized products. In the literature, *assembly sequence planning (ASP)* (which is in itself already an NP-hard combinatorial problem [1]) is usually treated as a sequencing problem for the geometry of the parts only. However, it is not only necessary to take into account the properties of the assembly itself, but also the robotic setup and its capabilities.

One of the first works on ASP is the FLAPS system [2], which uses semantic symbols to model the interference, connection and contact of the parts of an assembly. In modern approaches more information is used to generate a sequence that incorporates the restrictions of the agent that executes the assembly. A *disassembly for assembly* strategy combined with an integrated grasp planner was successfully used in our robotic setup [3]. Assembly sequence planning can be seen as a variant of task and motion planning (TAMP) problem. For example, the Asymov system [4] uses symbols to represent places and relations of the world. This system is able to formally express the geometric preconditions to perform a given action and then compute the effects of each action.

To master this increased complexity, we propose to integrate various feedback channels from the robot system into the planning architecture. We present an *assembly sequence planning (ASP)* framework which maps constraints from the robotic execution to rules in a logic planner layer. By a sequential execution in simulation our planner accumulates knowledge of the possible failure situations that may arise during the assembly procedure. That is, our system analyzes the failed executions in order to identify the constraints that prevent the robot from finishing the assembly. It extracts new *symbolic rules* for the logic layer to avoid similar conflicts in the continuing search for feasible sequences. The expert knowledge encoded in the relation between constraints and symbolic rules increases the performance of the ASP. This information is exploited by the logic layer (see Fig. 1) to prune the search for solutions which results in an improved overall planning time.

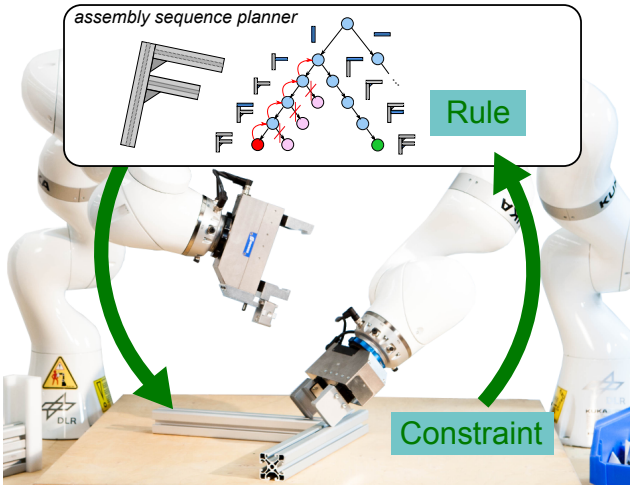

Fig. 1. The presented assembly sequence planner is able to detect constraints in the physical world and converts them into symbolic rules. These rules improve the planning time of new solutions by pruning the search space.

II. Improving Assembly Sequence Planning through Iterative Rule Generation

In this work, a novel approach is presented, in which constraints are automatically discovered by simulations and subsequently converted into symbolic rules. We introduce our iterative planner architecture which uses these rules to improve planning times.

A. Iterative planning architecture

We consider the search for solutions of the ASP problem as an iterative process. A diagram of the principal components of our approach is presented in Fig. 2. The procedure is initiated with an assembly specification and an initial set of optional symbolic rules, provided by the user or any other knowledge entity. Based on these inputs, the system generates a first symbolic solution for the assembly

[1]Institute of Robotics and Mechatronics, German Aerospace Center (DLR), Wessling, Germany.
[2] TUM - Technische Universität München, Faculty of Informatics, Garching, Germany

sequence in the logic layer. Then, the system checks the feasibility of the sequence in the physical layer. Here, the execution of the sequence is simulated in our system performing different inverse kinematic, motion and geometry checks. In case of failure, the situation is analyzed and, if possible, converted into a new symbolic rule (red arrow) as a feedback for the logic layer. The search procedure continues under consideration of the newly acquired rule and generates new possible solutions to be checked in the physical layer. Accordingly, the system reduces the search space continuously and improves search efficiency over time.

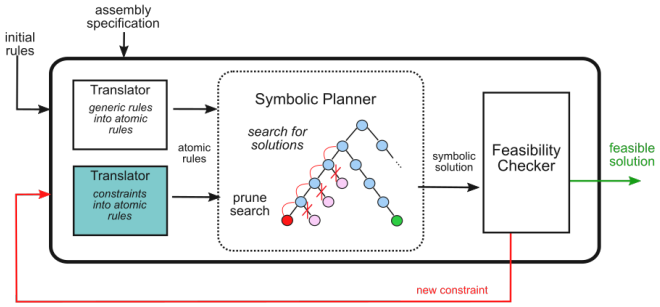

Fig. 2. Diagram of the system architecture. Given the initial rules and the assembly specification the symbolic planner searches for sequences that are then checked for feasibility in the physical layer. Detected constraints are given as a feedback for the continuing search.

B. Symbolic Representation of Assembly Sequence

The main elements of the proposed representation constitute the *parts* to be assembled, their relations and the associated *rules*.

1) Parts: The whole assembly is composed by a set $\mathcal{P}$ of N parts:

$$\mathcal{P} = \{p_1, p_2, ..., p_N\}.$$

Each part p_i is specified by a name, a type and its pose relative to the assembly. This data is initially provided by the user or from CAD data.

2) Relations: The parts in the assembly are in physical relations that can be represented semantically. A basic relation is described by two parts and a label which indicates the type of relation e. g. `in_touch` or `screwed`.

$$Relation : E(p_a, p_b, label),\ with\ p_a, p_b \in \mathcal{P}.$$

We will note $\mathcal{E}$ the set of all relations e of an assembly. Given the pose and the type of the parts, many relations can be deduced automatically e. g. by geometric analysis or prior knowledge. Furthermore, it is also possible to provide type-based generic relations, which the system resolves into pairwise relations between individual parts.

3) Rules: Rules synthesize different types of constraints, which are used to reduce the search space in the logic layer. We formalize two principles, which are sufficient for our assembly sequencing problem. First, we define *precedence* between parts and second, the grouping of parts into *sets*. A precedence represents an ordered pair of parts. It indicates that p_x should be assembled before p_y, forming a partial order in the set $\mathcal{P}$:

$$Precedence : P(p_x, p_y),\ with\ p_x, p_y \in \mathcal{P}.$$

Sets are formally defined as $\mathcal{S} = \{p_{s_1}, p_{s_2}, ...\}$ which contain parts p_{s_i}:

$$Set : S(\mathcal{S}, k),\ with\ \mathcal{S} \subseteq \mathcal{P},\ k \in Z^+.$$

Indicating that parts p_{s_i} should be assembled as close in the sequence as possible. Note that parts can be elements of multiple sets. Therefore, we define the priority k to denote the relevance of each set. The priority will be used in the search such that sets with higher priority are completed earlier.

We make a distinction between generic and atomic rules. The first one refers to abstract types (e. g. profile, angle brackets, slotnuts), whereas the second is defined by a particular group of pieces (e. g. profile A and profile B). Generic rules can be applied to all the parts of an assembly, or only to parts with special properties (i. e. that are related.)

III. Insights and Conclusions

This work proposed an assembly planning framework that is able to generate new planning rules taking into account not only the assembly but also the robotic agent in charge of it. As a result the performance of the system increases with every new rule learned. First experiments indicate that the proposed feedback procedure can significantly improve the time to find *all* feasible solutions compared to an uninformed search algorithm. This is especially true for assembly scenarios with many parts and a high number of possible solutions.

An idea for future implementations is the use of pattern recognition to match symbolic sequence of parts to different constraints detected in the physical layer. Finally, we believe the capability of adapting plans not only to different assemblies but also to available robotic systems in the setup, will play an important role in future production lines.

References

[1] M. F. F. Rashid, W. Hutabarat, and A. Tiwari, "A review on assembly sequence planning and assembly line balancing optimisation using soft computing approaches," *The Int. Journal of Advanced Manufacturing Technology*, vol. 59, no. 1-4, pp. 335–349, 2012.

[2] G. Dini and M. Santochi, "Automated sequencing and subassembly detection in assembly planning," *CIRP Annals-Manufacturing Technology*, vol. 41, no. 1, pp. 1–4, 1992.

[3] K. Nottensteiner, T. Bodenmueller, M. Kassecker, M. A. Roa, A. Stemmer, T. Stouraitis, D. Seidel, and U. Thomas, "A complete automated chain for flexible assembly using recognition, planning and sensor-based execution," in *ISR 2016: 47st Int. Symposium on Robotics; Proceedings of.* VDE, 2016, pp. 1–8.

[4] S. Cambon, R. Alami, and F. Gravot, "A hybrid approach to intricate motion, manipulation and task planning," *The Int. Journal of Robotics Research*, vol. 28, no. 1, pp. 104–126, 2009.