Selfoptimized Assembly Planning for a ROS Based Robot Cell

Daniel Ewert, Daniel Schilberg and Sabina Jeschke

Abstract In this paper, we present a hybrid approach to automatic assembly planning, where all computational intensive tasks are executed once prior to the actual assembly by an Offline Planner component. The result serves as basis of decision-making for the Online Planner component, which adapts planning to the actual situation and unforeseen events. Due to the separation into offline and online planner, this approach allows for detailed planning as well as fast computation during the assembly, therefore enabling appropriate assembly duration even in nondeterministic environments. We present simulation results of the planner and detail the resulting planner's behavior.

Keywords Assembly Planning · Cognitive Production Systems · ROS

1 Introduction

1.1 Motivation

The industry of high-wage countries is confronted with the shifting of production to low-wage countries. To slow down this development, and to answer the trend towards shortening product life-cycles and changing customer demands regarding individualized and variant-rich products, new concepts for the production in high-wage countries have to be created. This challenge is addressed by the Cluster of Excellence "Integrative production technology for high-wage countries" at the RWTH Aachen University. It researches on sustainable technologies and strategies on the basis of the so-called polylemma of production [1]. This polylemma is spread between two dichotomies: First between scale (mass production with limited product range) and scope (small series production of a large variety of products), and second between value and planning orientation. The ICD) "Self-optimizing Production Systems"

D. Ewert (✉) · D. Schilberg · S. Jeschke
IMA/ZLW & IfU, RWTH Aachen University, Dennewartstr. 27, 52068 Aachen, Germany
e-mail: daniel.ewert@ima-zlw-ifu.rwth-aachen.de

Originally published in "ICIRA 2012", © Springer 2012.
Reprint by Springer International Publishing AG 2016,
DOI 10.1007/978-3-319-46916-4_2

focusses on the reduction of the latter dichotomy. It's approach for the reduction of this polylemma is to automate the planning processes that precede the actual production. This results in a reduction of planning costs and ramp-up time and secondly it allows to switch between the production of different products or variants of a product, hence enabling more adaptive production strategies compared to current production. Automatic replanning also allows to react to unforeseen changes within the production system, e.g. malfunction of machines, lack of materials or similar, and to adapt the production in time. In this paper we present the planning components of a cognitive control unit (CCU) which is capable to autonomously plan and execute a product assembly by relying entirely on a CAD description of the desired product.

1.2 Use Case Description

The CCU is developed along a use case scenario for an assembly task in a nondeterministic production environment [2]. This scenario is based on the robot cell depicted in Fig. 1.

Of the two robots of the robot cell, only Robot2 is controlled by the CCU. Robot1 independently delivers parts in unpredictable sequence to the circulating conveyor belt. The parts are then transported into the grasp range of Robot2 who then can decide to pick them up, to immediately install them or to park them in the buffer area.

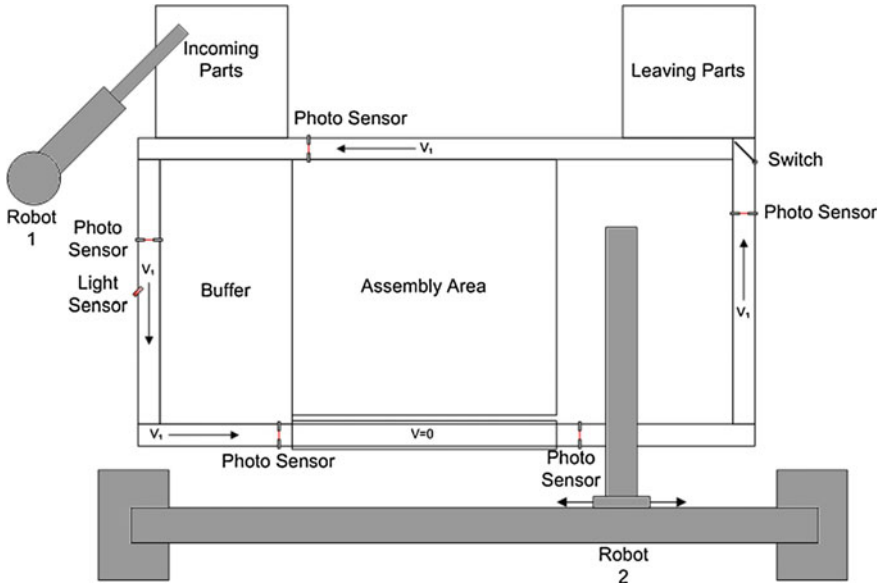

Fig. 1 Schematic of the robot cell

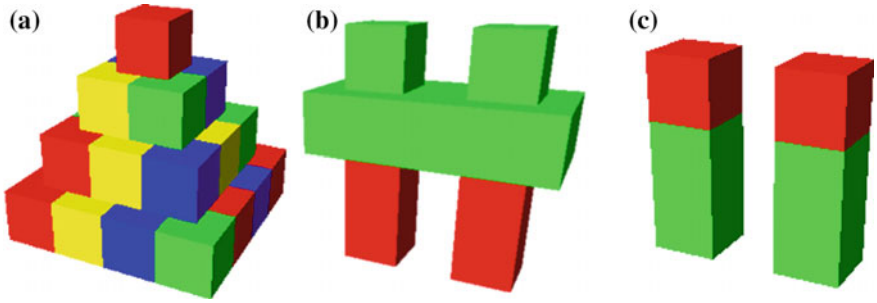

Fig. 2 Toy model products for planner evaluation

The scenario also incorporates human-machine cooperation. In case of failure, or if the robot cannot execute a certain assembly action, the CCU is able to ask a human operator for assistance. To improve the cooperation between the operator and the machine, the operator must be able to understand the behavior and the intentions of the robot [3]. Therefore, machine transparency is a further major aspect in our concept.

The only sources of information to guide the decision making of the CCU are a CAD description of the desired product, the number and types of single parts currently in the buffer and on the conveyor belt and the current state of the assembly within the Assembly Area. The planner is evaluated with the figures described in Fig. 2. The pyramid construct (a) serves here as a benchmark for the computational complexity of our planning approach and has been used in different sizes (base areas of 2×2, 3×3, and 4×4 blocks). Construct (b) and (c) are used to demonstrate the planner's behavior.

2 Related Work

In the field of artificial intelligence planning is of great interest. There exist many different approaches to planning suitable for different applications. Hoffmann developed the FF planner, which is suitable to derive action sequences for given problems in deterministic domains [4]. Other planners are capable to deal with uncertainty [5, 6]. However, all these planners rely on a symbolic representation based on logic. The corresponding representations of geometric relations between objects and their transformations, which are needed for assembly planning, become very complex even for small tasks. As a result, these generic planners fail to compute any solution within acceptable time.

Other planners have been designed especially for assembly planning and work directly on geometric data to derive action sequences. A widely used approach is the Archimedes system by Kaufman et al. [7] that uses And/Or-Graphs and an "Assembly by Disassembly" strategy to find optimal plans. U. Thomas [8] follows this strategy,

too, but where the Archimedes system relies on additional operator-provided data to find feasible subassemblies, Thomas uses only the geometric information about the final product as input. However, both approaches are not capable of dealing with uncertainty.

Other products for assembly planning focus on assisting product engineers set up assembly processes. One example is the tool Tecnomatix from Siemens [9], which assists in simulating assembly steps, validates the feasibility of assembly actions etc. All of the mentioned works do not cover online adaption of assembly plans to react on changes in the environment. One exception is the system realized by Zaeh et al. [10], which is used to guide workers through an assembly process. Dependent on the actions executed by the worker, the system adapts its internal planning an suggests new actions to be carried out by the worker. The CCU uses the same technique for plan adaption.

3 Autonomous Assembly Planning

3.1 Hybrid Assembly Planning

The overall task of the CCU is to realize the autonomous assembly in a nondeterministic environment: Parts are delivered to the robot cell in random sequence and the successful outcome of an invoked assembly action cannot be guaranteed. While assembly planning is already hard even for deterministic environments where all parts for the assembly are available or arrive in a given sequence [8], the situation becomes worse for this unpredictable situation. One approach to solve the nondeterministic planning problem would be to plan ahead for all situations: Prior to the assembly all plans for all possible arrival sequences are computed. However, this strategy soon becomes unfeasible: A product consisting of n parts allows for n! different arrival sequences, so a product consisting of 10 parts would already result in the need to compute more than 3.6 million plans. Another approach would be to replan during the assembly every time an unexpected change occurs in the environment. This strategy, however, leads to unacceptable delays within the production process.

Therefore, our approach follows a hybrid strategy. All computational intensive tasks are executed once before the actual assembly. This is done by an Offline Planner component. The results of this step serve as basis of decision-making for the Online Planner component, which adapts planning to the actual situation and unforeseen events. Due to this separation, our approach (see Fig. 3) allows for detailed planning as well as fast computation during the assembly, therefore enabling appropriate assembly duration even in nondeterministic environments. The Offline Planner contains a CAD Parser which derives the geometric properties. The currently supported format is STEP [11]. This data is then processed by the graph generator. The details of this process are explained in Section 3.2. The Online Planner consists of the components Graph Analyzer, Parallelizer and Cognitive Control, which are detailed in Section 3.3.

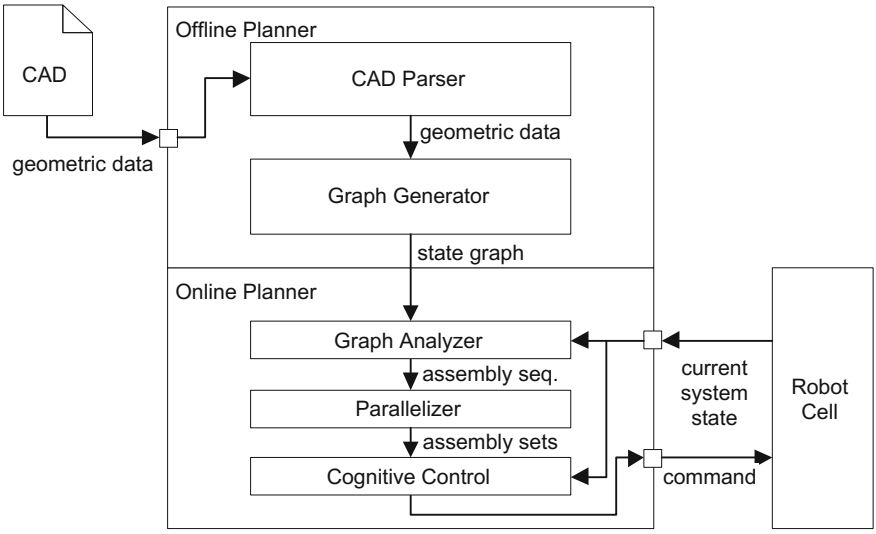

Fig. 3 System overview of the hybrid approach

3.2 Offline: Graph Generation

The Offline Planner receives a CAD description of the desired final product. From this input it derives the relations between the single parts of the product via geometrical analysis as described in 0. The results are stored in a connection graph. Assembly sequences are now derived using an assembly-by-disassembly strategy: Based on the connection graph, all possible separations of the product into two parts are computed. The feasibility of those separations is then verified using collision detection techniques. Unfeasible separations are discarded. The remaining separations can then be evaluated regarding certain criteria as stability, accordance to assembly strategies of human operators or similar. The result of this evaluation is stored as a score for each separation. This separation is recursively continued until only single parts remain. The separation steps are stored in an and/or graph [12], which is then converted into a state graph as displayed in Fig. 4 using the method described in Ewert D., D. Schilberg, and S. Jeschke [13]. Here nodes represent subassemblies of the assembly. Edges connecting two such nodes represent the corresponding assembly action which transforms one state into the other. Each action has associated costs, which depend on the type of action, duration, etc. Also, each edge optionally stores information about single additional parts that are needed to transform the outgoing state into the incoming state.

The graph generation process has huge computational requirements for time as well for space. Table 1 shows the properties of resulting state graphs for different products. The results show the extreme growth of the graph regarding the number of parts necessary for the given product. However, as can be seen when comparing the

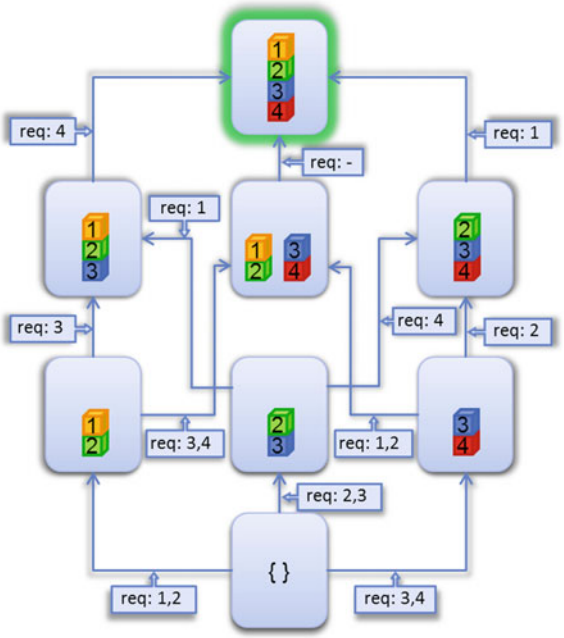

Fig. 4 State graph representation of the assembly of a four blocks tower

Table 1 State graph properties for different products

Product	# Parts	# Nodes of graph	# Edges of graph
Construct (a) (size 2 × 2)	5	17	33
Construct (c)	6	16	24
Construct (b)	14	361	1330
Construct (a) (size 3 × 3)	14	690	2921
Construct (a) (size 4 × 4)	30	141,120	1,038,301

state graphs of both constructs with 14 parts, the shape of a product affects the graph, too: The more possible independent parts are from each other, the more different assembly sequences are feasible. Therefore the graph of the construct (a) with 14 parts has almost twice the size of the state graph resulting from construct (b).

3.3 Online: Graph Analysis

The state graph generated by the Offline Planner is then used by the Online Planner to derive decisions which assembly actions are to be executed given the current situation of an assembly. The Online Planner therefore executes the following process iteratively until the desired product has been assembled: The Graph Analyzer perceives

the current situation of the assembly and identifies the corresponding node of the state graph. In earlier publications [13] we suggested an update phase as next step. In this phase all costs of the graphs edges reachable from that node were updated due to the realizability of the respective action. The realizability depends on the availability of the parts to be mounted. Unrealizable actions receive penalty costs which vary depending on how close in the future they would have to be executed. This cost assignment makes the planning algorithm avoid currently unrealizable assemblies. Additionally, due to the weaker penalties for more distanced edges, the algorithm prefers assembly sequences that rely on unavailable parts in the distant future to assemblies that immediately need those parts. Preferring the latter assembly results in reduced waiting periods during the assembly since missing parts have more time to be delivered until they are ultimately needed. Using the A* algorithm [14] the Online Planner now derives the cheapest path connecting the node matching the actual state with a goal node, which presents one variant of the finished product. This path represents the at that time optimal assembly plan for the desired product. The Parellelizer component now identifies in parallel or arbitrary sequence executable plan steps and hands the result to the Cognitive Control for execution. Here the decision which action is actually to be executed is made. The process of parallelization is detailed in [13].

However, updating all edge cost reachable from the node representing the current state is a computational intensive task. To overcome this problem, the edge cost update can be combined with the A* algorithm, so that only edges which are traversed by A* are updated. This extremely reduces the computational time, since only a fraction of the graphs node is examined. So even for large graphs, the Online Planner is able to derive a decision in well under 100 ms in worst case. Figure 5 shows the nodes reachable and examined by the Online Planner during the assembly of a 4×4 construct. Plateaus in the graph depict waiting phases where the assembly cannot continue because crucial parts are not delivered.

3.4 Planner Behaviour

Figure 6 shows the course of the assembly for the construct (c). Newly arrived parts are shown in the third column. They can either be used for direct assembly (first column) or otherwise are stored in a buffer shown in column 2. The right column depicts the plan that is calculated based on the parts located. Here the number of a given block denotes the position where that block is to be placed. In step 0, no parts have been delivered. The planner therefore has no additional information and produces an arbitrary but feasible plan. In step 1 a new green block is delivered, which matches the first plan step. The related assembly action is therefore executed and the new block is directly put to the desired position. In step 2 a new red block is delivered. Given the current state of the assembly and the new red cube, the planner calculates an improved plan which allows to assemble this red block earlier than originally planned: Now it is more feasible to first mount two green blocks on top

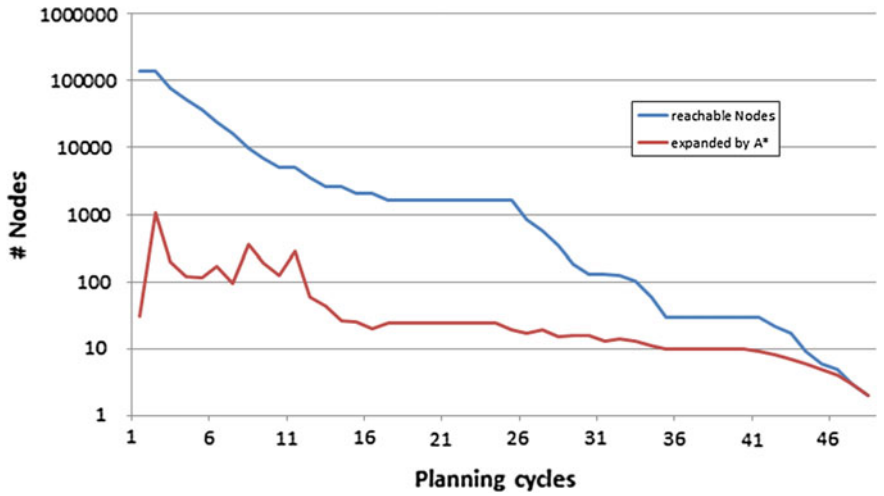

Fig. 5 Number of nodes reachable from the node representing the given situation and number of nodes that are examined by the A*-algorithm. Number of nodes is shown using a logarithmic scale

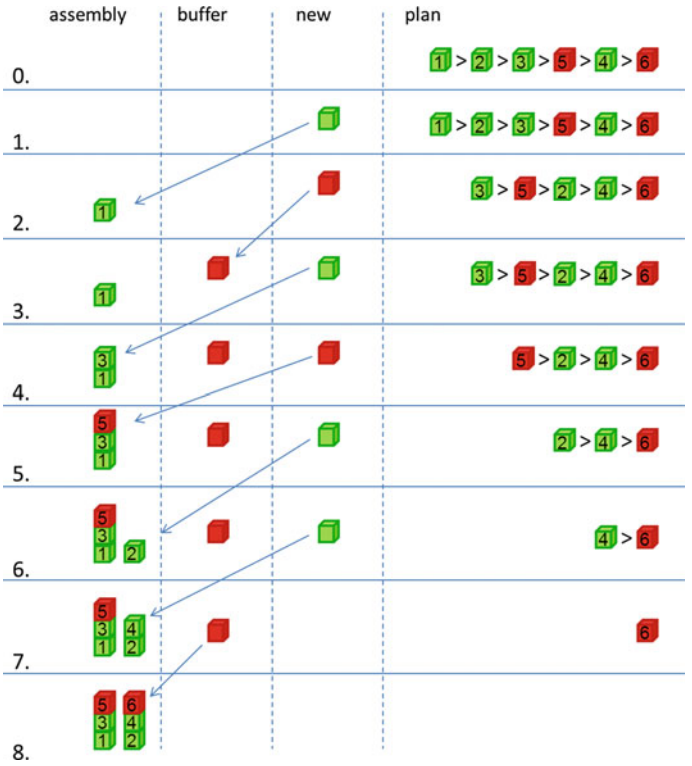

Fig. 6 Exemplary assembly flow for construct (c)

of each other (positions 1 and 3), because then the red block can be assembled, too (position 5). This plan step is executed in step 3 when a second green block becomes available. Now, in step 4, it is possible to mount a red block. From that step on only one feasible assembly sequence is possible, which is then executed.

The described behaviour results in more rapid assemblies compared to simpler planning approaches: A purely reactive planner which would follow a bottom up strategy, would have placed the first two green blocks next to each other (positions 1 and 2). Thus, in step 5 no assembly action would have been possible and the assembly would have to stop until a further green cube would be delivered.

4 Summary

In this paper we presented our hybrid approach for an assembly planner for nondeterministic domains. We described the workflow of the offline planner, which analyses CAD data describing the desired product. The outcome of the offline planner is a state graph which holds all possible (and feasible) assembly sequences. This graph is generated by following an assembly by disassembly strategy: Recursively all possible separations of the final product are computed until only single parts remain. During the actual assembly, this state graph is updated to mirror the current situation of the assembly, specially the availability of newly delivered parts. Using the A* algorithm, the at that time optimal assembly sequence is derived and handed over to the cognitive control unit, which then decides which assembly step gets to be executed. This step is then executed and the outcome of that step is reported back to the planning system. This process is iterated until the product is completed.

5 Outlook

Future work must optimize the described Planners. Using techniques of parallel programming and by incorporating specialized databases which can cope efficiently with large graphs, the planning duration can be improved. Subsequently, the planner will be extended to be able to deal with industrial applications as well as plan and control the production process of a complete production network.

References

1. C. Brecher, F. Klocke, G. Schuh, R. Schmitt, eds., *Excellence in Production*. Apprimus Verlag, Aachen, Germany, 2007
2. C. Brecher, T. Kempf, W. Herfs, Cognitive control technology for a self-optimizing robot based assembly cell. In: *Proceedings of the ASME 2008 International Design Engineering Technical*

Conferences & Computers and Information in Engineering Conference. America Society of Mechanical Engineers, U.S., 2008, pp. 1423–1431
3. B. Kausch, C.M. Schlick, W. Kabuß, B. Odenthal, M.P. Mayer, M. Faber, Simulation of human cognition in self-optimizing assembly systems. In: *Proceedings of 17th World Congress on Ergonomics IEA 2009*. Beijing, China, 2009
4. J. Hoffmann, FF: the fast-forward planning system. The AI Magazine, 2001
5. J. Hoffmann, R. Brafman, Contingent planning via heuristic forward search with implicit belief states. In: *In Proceedings of ICAPS'05*. AAAI, 2005, pp. 71–80
6. C. Castellini, E. Giunchiglia, A. Tacchella, O. Tachella, Improvements to SAT-based conformant planning. In: *Proc. of 6th European Conference on Planning*. 2001
7. S. Kaufman, R. Wilson, R. Jones, T. Calton, A. Ames, Ldrd final report: Automated planning and programming of assembly of fully 3d mechanisms. Technical Report SAND96-0433, Sandia National Laboratories, 1996
8. U. Thomas, *Automatisierte Programmierung von Robotern für Montageaufgaben*, *Fortschritte in der Robotik*, vol. 13. Shaker Verlag, Aachen
9. Tecnomatix, 2011. URL http://www.plm.automation.siemens.com/en_us/products/tecnomatix/index.shtml
10. M. Zäh, M. Wiesbeck, A model for adaptively generating assembly instructions using state-based graphs. In: *Manufacturing Systems and Technologies for the New Frontier*, Springer, London, 2008
11. F. Röhrdanz, H. Mosemann, F. Wahl, HighLAP: a high level system for generating, representing, and evaluating assembly sequences. 1996, pp. 134–141
12. L. Homem de Mello, A. Sanderson, AND/OR graph representation of assembly plans. In: *Proceedings of 1986 AAAI National Conference on Artificial Intelligence*. 1986, pp. 1113–1119
13. D. Ewert, D. Schilberg, S. Jeschke, Selfoptimization in adaptive assembly planning. In: *Proceedings of the 26th International Conference on CAD/CAM Robotics and Factories of the Future*. 2011
14. P. Hart, N. Nilsson, B. Raphael, A formal basis for the heuristic determination of minimum cost paths. IEEE Transactions on Systems Science and Cybernetics **4** (2), 1968, pp. 100–107