Automatic Planning of Manufacturing Processes using Spatial Construction Plan Analysis and Extensible Heuristic Search

Ludwig Nägele, Andreas Schierl, Alwin Hoffmann and Wolfgang Reif
Institute for Software & Systems Engineering, University of Augsburg, 86159 Augsburg, Germany

Keywords: Manufacturing Planning, Construction Plan Analysis, Heuristic Planning.

Abstract: When automating small-batch manufacturing processes, the time spent for process planning and robot programming becomes more important. This paper proposes an automated process including construction plan analysis, process planning and execution to reduce the amount of manual work required. The process starts by analyzing the structure of the desired product and deriving required process step results, then uses heuristic search to find possible production steps and task assignments, and concludes by simulating or executing the resulting production plan. The approach is evaluated on a case study with a simulated robot automatically building different LEGO® DUPLO® structures starting from a 3D model defining the desired product.

1 INTRODUCTION

When new products transfer from design to production, not only the supply of resource materials has to be ensured but also manufacturing processes have to be defined that consist of many single production steps. When automating production with robots to reduce cost, a suitable robot cell configuration needs to be provided, robots have to be programmed and teached, end-effectors such as welding tongs are to be integrated. Additionally, proper flow of production parts through the manufacturing process is necessary.

Thus, before a product finally comes to production, a lot of time-consuming work has to be performed by engineers, technicians and programmers. While this effort usually is profitable for high lot sizes, the process and program definition for smaller lot sizes or even single individual product manufacturing forms an enormous challenge to the producer.

In this paper, we present a modular approach for automatic planning of manufacturing processes aiming for speeding up the entire development process up to the final robot program definition. In a first step, an engineer's construction plan is parsed and analyzed to generate a structural model. From this model, possible process definitions and their order are derived which describe manipulator independent manufacturing steps. Appropriate robots are assigned to each process step in order to form an executable action as part of the final program. Besides, techniques are used for optimizing both planning time and the result.

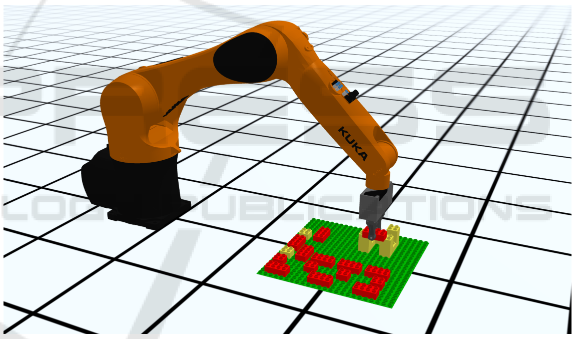

Figure 1: Simulated KUKA KR 6 robot and Schunk WSG 50 gripper performing LEGO brick placement.

The approach is suitable for production of high lot sizes just as for manufacturing unique products.

As a main contribution, the paper presents an approach for spatial analysis of a construction blueprint resulting a detailed structure model of the product. Furthermore, the work introduces a technique for deriving an optimized order of process steps necessary for manufacturing a given structure model.

After an overview of other approaches to perform automatic planning of manufacturing tasks in section 2, an introduction to the case study with LEGO® DUPLO® is given in section 3 (see Fig. 1). Section 4 presents the elements and steps of the approach in detail, followed by a description of the heuristics used for finding proper sequences of process steps and for planning with appropriate robots. As an evaluation of the approach, its application to the LEGO domain

Nägele, L., Schierl, A., Hoffmann, A. and Reif, W.
Automatic Planning of Manufacturing Processes using Spatial Construction Plan Analysis and Extensible Heuristic Search.
DOI: 10.5220/0006861705760583
In *Proceedings of the 15th International Conference on Informatics in Control, Automation and Robotics (ICINCO 2018) - Volume 2*, pages 576-583
ISBN: 978-989-758-321-6

for building structures is presented in section 6. Experimental results from a survey of different heuristic variants for the LEGO example are discussed in section 7 before the work concludes in section 8.

2 RELATED WORK

Manufacturing and assembly with robots is still an ongoing topic in research. Some approaches concentrate on discrete planning techniques to solve allocation of resources and task assignment to specific robots. For this, the STRIPS planner developed in Stanford (Fikes and Nilsson, 1971) and the Planning Domain Definition Language (PDDL) by McDermott et al. (1998) which bases on STRIPS are commonly used. In PDDL, physical entities are specified as objects and domain-specific predicates are defined which hold situational properties of these objects. These predicates are used to describe an initial state and a goal specification of the planning problem. With actions that are defined with preconditions and are used to manipulate a situation by influencing the value of predicates, the planning problem can be solved. However, our planning domain is highly dependent on geometric reasoning with pre- and postconditions calculated during planning time which is difficult to express in PDDL.

In the european project SkillPro (Pfrommer et al., 2015), skills are used to describe functionality of available resources. Together with manufacturing requirements of the final product, executable process actions can be generated. While actions can be dynamically orchestrated and integrated into the manufacturing process at runtime, in our project we want to focus on an overall optimization of the entire manufacturing process considering products' as well as resources' characteristics and performances.

In the last few years a trend has evolved to consider motion planning together with classical task planning for robot based manufacturing. One approach by Kaelbling and Lozano-Pérez (2011) proposes stepwise planning and execution of successive actions in order to reduce complexity. Planning and decomposition of tasks is done only in limited depth. The intermediate results are then executed and planning on task level is subsequently continued based on the new situation. In a further work, Levihn et al. (2013) introduce concepts for foresight with a belief state and reconsideration as improvements to their first approach. For our project, we plan to integrate the concept of interleaved execution while planning, but use simulation techniques instead.

Many approaches exist which try to bring predefined process steps to execution on a robot cell. But researchers also pay attention to automatic definition of process steps by raw construction models or 3D data. In an approach where KUKA youbots are used to assemble an IKEA furniture, Knepper et al. (2013) present a geometric preplanning strategy to extract a final assembly configuration (blueprint) from the form and quantity of available parts. In a second step, a symbolic planner is used to find a sequence of operations to assemble the blueprint. The presented separation of blueprint analysis, operation sequence examination and finally execution will be a paradigm for our project.

In their work about an offline programming platform for automatic programming of manufacturing tasks in the domain of carbon fiber-reinforced polymers (CFRP), Nägele et al. (2015) present techniques to derive executable robot programs from an initial CAD model of the final product. The CAD model is parsed and decomposed into a hierarchical task representation with pre- and postconditions for each leaf. The concept of task contribution units is introduced in a further work by Macho et al. (2016) which allows for dynamic planning of each task basing on a set of various exchangeable modules. Such a module can provide generic calculation functionality, apply domain-specific knowledge or involve interaction with a human expert. While this offline programming platform is mainly focused on the domain of CFRP and the presented planning problem is almost sequential, the idea of parsing and analyzing some kind of blueprint is taken as inspiration for our project.

3 CASE STUDY

Building up and programming manufacturing systems often includes generic challenges on the one hand and domain-specific problems on the other hand. As a case study, an application in the domain of construction toy LEGO® DUPLO® is presented. Its interchangeable and often symmetric raster-sized bricks keep domain-specific problems simple and allow for focusing on generic challenges of automatized production design.

In our first scenario, we use a base plate of 24×24 studs, 13 colored bricks with dimensions of 2×2 studs and 13 bricks with a raster of 2×4 studs. As an initial setup, all bricks are randomly placed planarly on the base plate as shown in Fig. 2 with a minimum distance of one raster width. The goal – the manufacturing product – is a yellow house topped with a red roof, having a space for an entry door and a window area on all three other sides (see Fig. 2).

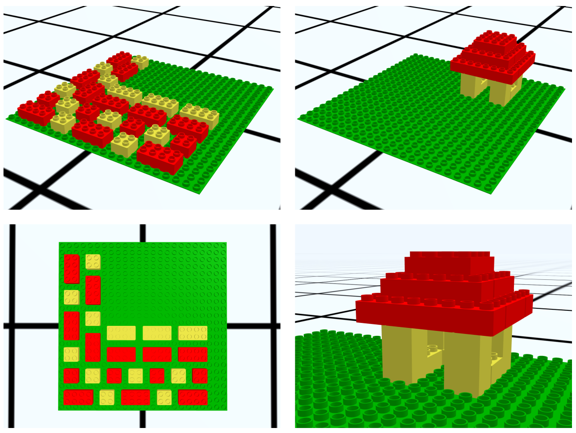
Figure 2: Initial setup (left) and goal structure (right) for building up a LEGO house.

For manipulation, a KUKA KR 6 robot with a mounted Schunk WSG 50 gripper is used. Both actuators, as well as all LEGO parts, are integrated in the Robotics API (Angerer et al., 2013) which allows for object-oriented behavior programming and provides visualization by an inbuilt engine. For evaluating conceptual and algorithmic results of this paper, the inbuilt simulation feature of the Robotics API is used to execute actuator behavior (see Fig. 1).

4 APPROACH

One main challenge in automation of manufacturing is to bring a construction plan (3D model, written script, etc.) to a stable, efficient and at best cost-optimal manufacturing program for a robot cell. There are several development phases and decisions which all influence the final result with respect to quality, resource efficiency and performance of both product and production system. We propose an extensible module based approach which is meant to ease this task by automatic programming of such processes. Different modules can be contributed to provide specific knowledge and computation, ranging from automation expertise for programming robots and process expertise regarding manipulation and material behavior, over to detailed knowledge about the specific application domain. The presented approach splits the development of manufacturing programs into four parts: structure analysis, process planning, task assignment and simulation and execution.

4.1 Structure Analysis

The overall goal of structure analysis is to extract as much information out of a given initial construction plan as possible. We assume that a construction plan describes all parts to be assembled as well as further information such as their relative positions to each other or applied treatments and processes. We name such a piece of information *Attribute*, which describes one or relates multiple parts of the product. Examples for attributes are that a product is painted or that multiple parts are plugged or glued together.

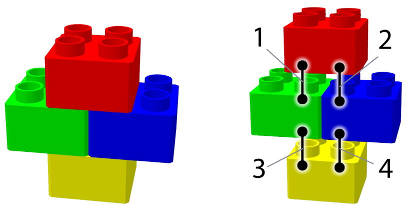

Figure 3: LEGO diamond structure with four attributes.

Fig. 3 shows a simple structure which is an expressive showcase for general planning challenges. Four 2×2 bricks are assembled in a diamond-like shape. The four connection lines illustrate attributes describing which bricks are in a stacked relationship to each other and which transformation between the respective bricks is applied.

Attributes are called *executionally equivalent* if they cannot be established independently. Assuming all three lower bricks in Fig. 3 are already stacked by attributes 3 and 4. A following placing of the top brick would result in simultaneously establishing attribute 1 and 2. An independent establishment of only one of these attributes is not possible. The property of executional equivalence is not valid in general on a specific set of attributes but depends on the particular setting: When attributes 2 and 3 are already established, the set of attributes 1 and 4 emerges to be executionally equivalent. However, with attributes 1 and 3 being established, attributes 2 and 4 are not executionally equivalent due to their contradicting stacking direction relating to their joint (blue) brick.

To automatically derive attributes from an initial construction plan, we propose *Analyzer Modules* which take a construction plan and return a set of attributes. They automatically compute attributes from existing properties or relationships, for example through geometrical inspection. Multiple analyzer modules can be provided for general purposes and different domains. For instance, one analyzer module determines parts which need to be bolted together while another one identifies glued parts and calculates minimal needed adhesive strength using an internal static analysis. Once analyzer modules are available for all properties and relationships of certain domain, they result a complete set of attributes when applied to a construction plan of the respective domain.

A data structure describing which attributes are established and which are not at a certain moment is called *Situation*. When manufacturing a product, at least two situations are known: a goal situation containing all established attributes to form the final product, and an initial situation which describes the setup before assembly begins. Here, a set of attributes describes initial placements of parts, for example.

Analyzer modules as well as types of attributes are generally contributed by application domain experts who know about technical details, what kinds of attribute types exist and how they can be identified in a plain construction plan. The result of structure analysis is an initial situation describing start setup and a goal situation which analyzer modules extracted out of a construction plan.

4.2 Process Planning

To transition an initial situation to a goal situation, detailed process descriptions (*Process Actions*), which represent abstract robot behavior, and their order are searched for. They furthermore indicate which attributes are established or removed when executing the behavior. For this, *Skill Modules* are introduced which create particular process actions that are applicable in a given situation. Depending on a concrete skill module implementation multiple alternative process actions are conceivable for one situation: If a situation describes three bricks placed on the floor, a grasping skill module provides three grasping actions – one for each brick. At this point, alternative process steps are introduced which later allow for overall variability exploration and optimization.

Skill modules and their respective process actions are contributed by process experts who have fundamental knowledge about process steps which can be taken in specific situations. As a result of process planning, skill modules provide possible process actions which can be performed in specific situations.

4.3 Task Assignment

Process actions describe behavior from the product's point of view. From automation view, process actions need to be performed by concrete manipulators like robots, grippers or tools. For defining the behavior of process actions, skill modules use generic behavior interfaces. These can later be instantiated by concrete manipulators whose capabilities are then used to perform the desired action. This concept is taken from the Robotics API where actuator and sensor interfaces are used to abstract robot actions (e.g. PTP, LIN, GRASP, MEASURE) from the underlying manipulators. With concrete manipulators assigned, a skill module can calculate a new situation that results when executing the process action in a given situation. When a product is being grasped, the gripper is also part of the resulting situation.

The basic purpose of task assignment is to find proper constellations of manipulators which can be used to perform a process action's task based on its behavior descriptions. In a flexible multi-functional robot cell like the one described by Angerer et al. (2015), multiple robots – or even teams of robots – might be applicable to perform the same task. This search is done by *Manipulator Modules* which find all manipulators or teams appropriate for the behavior interfaces of the process action. In the current stage, alternatives of manipulators are provided, but there is no concrete strategy or criteria for selection yet.

Process actions describe their task without knowing the actuator effectively used. This allows to inject concrete robots and to evaluate the appropriateness while planning. Manipulator modules are contributed by process and automation experts. The step of task assignment deals with resource (manipulator) allocation and assignment as well as with finally providing concrete execution plans.

4.4 Simulation and Execution

Once specific manipulators are assigned to a process action, it can be executed in a given situation. The presented approach uses Robotics API activities (Angerer et al., 2013) to execute process actions with robots. The framework allows for both inbuilt simulation enhanced with a graphical 3D visualization (see Fig. 1) as well as execution with real robots on its Robot Control Core (Vistein et al., 2014). For product parts being correctly positioned and visualized, the changes performed by process actions are transferred to the world model of the Robotics API by translating attributes to geometrical or logical properties supported by the framework.

The concepts presented in this work are evaluated in simulation only. However, the Robotics API allows for a transition to real execution. For this, the exact positions of resources, robots and parts need to be reflected back into the world model. In general, processes furthermore need testing and adjustments of process parameters by automation experts.

5 HEURISTIC SEARCH

For defining an overall execution plan, several steps have to be taken using the aspects described in the

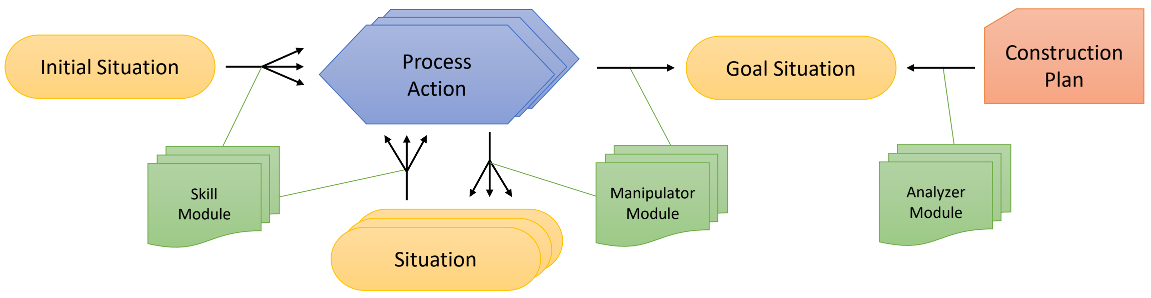

Figure 4: Overview of steps for planning, modules and intermediate artifacts.

previous sections, each offering a variety of options to continue with. Figure 4 gives an overview of the steps, modules and artifacts involved when searching for an overall process plan. In detail, following algorithmic steps have to be taken:

1. Find the goal situation by applying all analyzer modules to the construction plan. Provide initial situation from a description of the initial setup. Choose initial situation as current situation.
2. Apply all skill modules to current situation and derive all possible process actions.
3. For each process action: Use manipulator modules to find all possible sets of manipulator instances which satisfy the process action's needed behavior interfaces.
4. For each set of manipulator instances: Assign it to the process action and derive its resulting situation. Iteratively repeat (2) through (4) with the new situation as current situation until goal situation is reached.
5. Execute each step of the found path using the corresponding manipulators and respectively update the world model after performing process actions.

As hinted by steps (2) and (3), an exponential number of options evolves due to alternative process actions for each situation and different manipulators for each single process action. To find an optimal process plan, we formulate an optimization problem for the A-Star algorithm (Hart et al., 1968). A-Star uses sets of nodes, weighted directed edges and cost-estimations for nodes to iteratively search for a cost-minimal path from an initial to a desired goal node. In the presented approach, situations act as nodes, with initial and goal situation as initial and goal nodes. Starting with initial situation, A-Star requests all edges to proceed with. For this, all possible process actions basing on initial situation are calculated. Each process action is combined with all possible manipulator sets which are applicable for execution. Such a combination of process action and manipulators is used as edge in A-Star. Based on their weights, edges are taken by A-Star and a subsequent situation is computed to continue with. The weight of an edge can be determined as the expense of the process action and the cost of manipulators used, for example, whereas expected remaining cost of a situation to reach goal situation needs to be estimated.

Cost estimation plays a fundamental role for A-Star. The performance is highly dependent on the calculated weights of transitions and the expected distances from nodes to the goal, which might be estimated exactly in rarest cases only. If overestimated, the result is not guaranteed to be cost-optimal. If underestimated, a cost-optimal solution is guaranteed but the search may take exhaustive time. Our approach estimates costs for situations based on their attributes: The cost of a situation is the number of attributes which still have to be established plus the number of attributes which need to be removed in order to reach goal situation. The actual weight of a process action transitioning from a situation to another is calculated as difference between the costs of both situations, but with a minimum of 1. This prevents weight-neutral process actions with no effect from being performed and leads to a cost-underestimation of situations which guarantees cost-optimal solutions.

6 APPLICATION TO LEGO

The planning approach is applied to the domain of LEGO and as a use case, a house as shown in Fig. 2 is built automatically using the presented concepts. In this domain, three kinds of attributes suffice to describe situations for the approach:

- A base plate is statically placed in the world by a *FixpointAttribute* which describes its position.
- *LegoPlacementAttributes* between two bricks describe that these are plugged together with a specific relative transformation.
- A *GraspAttriute* denotes a brick which is grasped by the gripper, as well as its grasping position.

While the spatial positions of bricks are already given for both initial and goal setup (see Fig. 2), initial and goal situations with attributes need to be computed using analyzer modules. Being LEGO domain experts, the authors created a *LegoBrickAnalyzerModule* which geometrically checks each pair of bricks for being in a plugged relationship. For each pair with an adequate spatial distance and rotation, a *LegoPlacementAttribute* is generated. In the diamond example, this analyzer module identifies the four attributes depicted in Fig. 3.

To determine possible manipulation steps, skill modules along with manipulator instances are used. For LEGO and a robot cell consisting of a robot and a gripper, two kinds of skill modules are relevant:

- *PickupSkillModule* provides process actions for grasping and picking up each brick which is free-standing in a given situation. After execution of this process action, existing *LegoPlacementAttribute*s of the brick are removed and a *GraspAttribute* between brick and gripper is established in the resulting situation.
- *PlaceSkillModule* provides process actions for placing an already grasped brick on the target position derived from the goal situation. The resulting situation contains new *LegoPlacementAttribute*s and is relieved from the *GraspAttribute*.

As a design-decision to this example, the gripper can only grasp one brick at once instead of clusters of bricks. In a first guess, a simple heuristic might build a LEGO structure strictly processing layer by layer from bottom to top starting at the base plate. This straightforward approach is capable in most cases, but due to specific intermediate structure setups, deadlock situations might occur: Assuming a final product with overhanging parts, these bricks will never be plugged from the downside. As another example, bricks needed for the first layer can be blocked by other bricks which will however not be removed.

To overcome such deadlock situations, a more complex heuristic is proposed for which all possible immediate steps are reviewed and rejected where necessary. Both *PickupSkillModule* and *PlaceSkillModule* precisely check geometrical and structural suitability for picking up or placing a brick. In particular, all attributes modified in one step are checked for required executional equivalence, and collisions are identified and excluded, e.g. when grasping a closely surrounded brick with hence no valid grasping point. Furthermore, a situation is rejected if it will eventually lead to a deadlock situation. Assuming the diamond example in Fig. 3 with first attribute 3 and then attribute 1 established. The resulting "C"-like construction excludes the blue brick from ever being inserted since its two attributes 2 and 4 are not executionally equivalent. To identify such eventual deadlocks in the situation where they become inevitable, for example when establishing attribute 1, look-aheads are performed which translate bricks and attributes into graphs and analyze future attributes between sub-graphs for executional equivalence in a depth-first search. Such *early deadlock predictions* are expensive but cut down on the state space significantly. As a further improvement, only established attributes that are transitively connected to a *FixpointAttribute* in both, current and goal situation, contribute to the cost estimation of a situation. This adjustment privileges bottom-up assembly (like the simple first guess does) but does not exclude other strategies. Thus, deadlock situations induced by the bottom-up strategy are avoided while speeding up the planning algorithm for building LEGO products.

7 EXPERIMENTAL RESULTS

For both LEGO structures, diamond and house, processes have been planned with the presented approach and with modules as described in the previous section. All experiments have been run on a Windows machine with an Intel(R) Core(TM) i7-7600U (2.80GHz) and 24 GB RAM. The heuristic search and all modules have been executed by a Java Virtual Machine within a single-threaded context. Table 1 gives a summary of all measured values. For the diamond example, 10,000 experiments have been performed – all resulting with 8 process steps. The shortest experiment investigated 12 states in A-Star to find a result, on average 15.91 states are explored. To plan the house, the calculation time of one state increases by a factor of almost 14 compared to the diamond. The reason is the higher amount of bricks which leads to longer analysis time to find possible next steps and time needed for collision checks. The house exam-

Table 1: Results of experiment runs for diamond and house.

	Diamond	House
Bricks	4	26
Base Plate	1	1
Test Runs	10,000	100
Result Steps	8	52
Avg. Time (Std. Deviation)	0.88 ms (± 0.79 ms)	280.81 ms (± 74.06 ms)
Investigated States (Min - Max)	15.91 (12 - 18)	368.21 (182 - 738)
Time per State	0.055 ms	0.763 ms

Table 2: Tests with different enabled aspects of the heuristic: For each kind of experiment, test runs, average and standard deviation time, average investigated states with min and max and average time per state is measured.

	SingleGrasp		MultiGrasp		MultiRobot	No-Deadl.-
	w/ BottomUp	w/o BottomUp	w/ BottomUp	w/o BottomUp		Prediction
Experiment Nr.	#1	#2	#3	#4	#5	#6
Test Runs	100	100	100	100	100	1
Avg. Time	0.281 s	368.477 s	0.389 s	1.005 s	9.112 s	> 1,800.0 s
(Std. Deviation)	(± 0.074 s)	(± 214.223 s)	(± 0.137 s)	(± 0.432 s)	(± 5.598 s)	-
Investigated States	368.21	519,948.70	418.77	956.55	1,427.42	> 1.1M
(Min - Max)	(182 - 738)	(8.58k - 1.06M)	(208 - 752)	(294 - 2,598)	(556 - 3,173)	-
Time per State	0.763 ms	0.709 ms	0.930 ms	1.051 ms	6.384 ms	*

ple has furthermore been used to evaluate different aspects of the heuristic with 100 test runs each. Table 2 compares six different experiments each with different sets of activated features. All terminating experiments returned a sequence of 52 process steps: Picking up 26 bricks and placing 26 bricks.

Two experiments have been performed with *SingleGrasp* which allows grasping of only one brick at once. The first one uses the heuristic as described in the previous section preferring bottom-up assembly. After 0.281 seconds with a standard deviation time of 74 milliseconds a valid process plan was found.

With bottom-up preference being disabled, experiment #2 exhibits an exploded set of states, leading to an overall longer time to find a valid result. In the house example, parts of the house which belong to the upper part might already be assembled on the base plate but can not be grasped in a later step. As an interesting fact, the average time per state decreases slightly. This is because there is no computation whether a randomly picked brick fits into the bottom-up strategy.

Two further experiments have been performed with *MultiGrasp* which allows for picking up and placing whole structures of LEGO instead of just single bricks. Parts of the product (the roof of the house, for example) can be assembled at another location – even before underlying parts are finished. Also here, both variants are tested, starting with bottom-up preference being enabled in experiment #3. All measured values seem similar to these of experiment #1. Due to preferred bottom-up strategy, partial assembly of parts does not occur in practice.

As a combination of disabled bottom-up and enabled grasping of multiple bricks, experiment #4 clearly shows that the large state space of a free (non bottom-up) strategy – where the order of steps is randomly chosen – can be compensated by the grasping of structures. The grasping of whole brick structures allows for assembly of different product parts independently and combining them afterwards.

Experiment #5 uses two robots and grippers, all other variants left to default (#1). The introduction of an alternative manipulator causes a significant increase in time needed to find a result. This is mainly caused by the huge number of states to be investigated, owing to the fact that both robots can grasp each LEGO brick – in parallel and in all combinations. A further conspicuousness is stated by the overproportional standard deviation time of this experiment, which can be explained by the variety of alternatives and the diversity of investigated states. Some test runs seem to find a valid trace in a very early try while others keep searching with stagnating guesses.

When early deadlock prediction is deactivated (see section 6), many states and paths are explored although an eventual deadlock may be certain. Experiment #6 (*No-Deadl.-Prediction*) is aimed to show the effects of this deactivation, but owing to the huge increase of state space no test run did terminate within the first half an hour. Instead, they finally ended up with memory exhaustion at about 1.1 million states. After about 20 minutes, each test run caused the Java garbage collector to continuously occupy CPU while significantly slowing down the overall test execution. Owing to this influence, no valid measurement for time per state can be made.

Besides an immense number of investigated states in the last experiment, a low average time per state is expected due to the deactivation of the prediction algorithm. Situations are not expensively investigated in detail to discover whether a deadlock will be unavoidable. As a scientific finding, even exhaustive algorithms which essentially increase time per state can be profitable and drastically reduce overall planning time as long as they reduce the state space significantly. A comparison of experiment #2 (bottom-up disabled) and experiment #1 (single-grasp and bottom-up) confirms this observation by the increased time per state but drastically reduced overall time.

8 CONCLUSIONS AND OUTLOOK

In this paper we introduced an approach for automatic planning of manufacturing processes to build products with robots. It bases on modules which aim for integrating expert knowledge and at the same time separate different domain expertises such as automation, process and application into independent components. The approach contains a heuristic search based on cost estimations to find a suitable manufacturing program. Here, the modules contribute to different steps in order to solve the overall problem: They allow for deriving a structural model from an engineer's construction plan of a goal product and are used to calculate processes to establish the structure. Furthermore, modules can plan concrete robot actions to perform the overall manufacturing of the final product. We applied the presented concept to the domain of LEGO and used it to plan the manufacturing of a LEGO house with a robot. It has been evaluated with different variations of heuristic properties.

As further work, we plan to improve the performance of the heuristic search, for example by reducing the state space through geometric reachability analysis between manipulator and object. Especially when planning with multiple manipulators, a strategy for the selection of alternative manipulators for each process action might be included into the approach. As a major future research question, we will investigate strategies for parallel execution of tasks using independent manipulators. The goal is to make planning for multiple robots in a multi-functional robot cell feasible without the rapidly growing planning time seen in the multi-robot evaluation results of our work. In a further step, a second use case in a different domain – such as CFRP – will be investigated to demonstrate that the presented concepts can be generically applied to other domains and in real world.

ACKNOWLEDGEMENTS

This work is partly funded by the German Research Foundation (DFG) under the TeamBotS grant.

REFERENCES

Angerer, A., Hoffmann, A., Schierl, A., Vistein, M., and Reif, W. (2013). Robotics API: Object-oriented software development for industrial robots. *J. of Software Engineering for Robotics*, 4(1):1–22.

Angerer, A., Vistein, M., Hoffmann, A., Reif, W., Krebs, F., and Schönheits, M. (2015). Towards multi-functional robot-based automation systems. In *Proc. 12th Intl. Conf. on Inform. in Control, Autom. & Robot.*, pages 438–443.

Fikes, R. and Nilsson, N. J. (1971). STRIPS: A new approach to the application of theorem proving to problem solving. *Artif. Intell.*, 2(3/4):189–208.

Hart, P. E., Nilsson, N. J., and Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107.

Kaelbling, L. P. and Lozano-Pérez, T. (2011). Hierarchical task and motion planning in the now. In *Proc. IEEE Intl. Conf. on Robot. & Autom.*, pages 1470–1477.

Knepper, R. A., Layton, T., Romanishin, J., and Rus, D. (2013). Ikeabot: An autonomous multi-robot coordinated furniture assembly system. In *Proc. IEEE Intl. Conf. on Robot. & Autom.*, pages 855–862.

Levihn, M., Kaelbling, L. P., Lozano-Pérez, T., and Stilman, M. (2013). Foresight and reconsideration in hierarchical planning and execution. In *Proc. IEEE/RSJ Intl. Conf. on Intell. Robots and Systems*, pages 224–231.

Macho, M., Nägele, L., Hoffmann, A., Angerer, A., and Reif, W. (2016). A flexible architecture for automatically generating robot applications based on expert knowledge. In *Proc. 47th Intl. Symp. on Robotics*. VDE Verlag.

McDermott, D., Ghallab, M., Howe, A., Knoblock, C., Ram, A., Veloso, M., Weld, D., and Wilkins, D. (1998). PDDL — the planning domain definition language. Technical Report TR 98 003/DCS TR 1165, Yale Center for Computational Vision and Control.

Nägele, L., Macho, M., Angerer, A., Hoffmann, A., Vistein, M., Schönheits, M., and Reif, W. (2015). A backward-oriented approach for offline programming of complex manufacturing tasks. In *Proc. 6th Intl. Conf. on Autom., Robotics & Applications*, pages 124–130. IEEE.

Pfrommer, J., Stogl, D., Aleksandrov, K., Navarro, S. E., Hein, B., and Beyerer, J. (2015). Plug & produce by modelling skills and service-oriented orchestration of reconfigurable manufacturing systems. *Automatisierungstechnik*, 63(10):790–800.

Vistein, M., Angerer, A., Hoffmann, A., Schierl, A., and Reif, W. (2014). Flexible and continuous execution of real-time critical robotic tasks. *Intl. J. of Mechatronics and Automation*, 4(1):27–38.