Originalarbeit

Elektrotech. Inftech. (2023) 140:572–577
https://doi.org/10.1007/s00502-023-01161-1

e+i elektrotechnik und informationstechnik

Approaching flexible production with planning methods

Marco De Bortoli · Peter Kohout · Dominik Lampel · Leo Fürbaß · Anna Masiero · Stefan Moser · Martin Nagele · Tobias Frick · Lukas Knoflach · Gerald Steinbauer-Wagner

Received: 26 May 2023 / Accepted: 23 August 2023 / Published online: 27 September 2023

Abstract Flexible production systems are becoming more and more important for the industry. Higher requirements for flexibility in the production process demand further improvements in regards to flexible automation. In this perspective, the RoboCup Logistics League competition was designed to provide a testbed for dynamic production domains. The related challenges range from robust navigation over manipulation to efficient production scheduling. In this paper, we provide an update on how the team GRIPS approches the flexible production challenge using modern planning methods. This includes a new long-term planning and scheduling strategy as well as a hierarchical structure which further develops such plans into Behavior Trees.

Keywords Flexible production · Shop floor · Autonomous robots · Planning and plan execution · Navigation

Ansatz für eine flexible Produktion mit Planungsmethoden

Zusammenfassung Flexible Produktionssyteme werden immer wichtiger für die Industrie. Höhere Anforderungen an die Flexibilität des Produktionsprozesses verlangen immer mehr Verbesserungen der flexiblen Automatisierung. Mit der Überlegung, ein Testbed für dynamische Produktionsumgebungen abzubilden, wurde der Wettbewerb „RoboCup Logistics League“ entworfen. Die Herausforderungen reichen von robuster Navigation über Manipulation bis hin zu effizienter Planung. In dieser Arbeit geben wir ein Update, wie das Team GRIPS an die Probleme der flexiblen Produktion mit modernen Planungsmethoden herangeht.

Schlüsselwörter Flexible Produktion · Autonome Roboter · Planung und Planausführung · Navigation · Shop Floor

M. De Bortoli (✉) · P. Kohout · D. Lampel · L. Fürbaß · A. Masiero · S. Moser · M. Nagele · T. Frick · L. Knoflach · G. Steinbauer-Wagner
Institute for Software Technology, Graz University of Technology, Inffeldgasse 16B/II, 8010 Graz, Austria
mbortoli@ist.tugraz.at

1 Introduction

As a result of the increasing e-commerce as well as the growing need for highly configurable products and swift delivery, there is a pressing requirement for enhanced flexibility in production processes. This emerging phenomenon, referred to as flexible production or Industry 4.0, asks for a higher level of automation to maintain reasonable pricing, consistent product quality, and rapid availability. Nevertheless, the rising demand for flexibility poses a challenge to the conventional rigid automation, thereby posting novel research questions within domains such as Robotics, Internet of Things (IoT), multi-agent systems, planning, and scheduling.

To tackle these challenges and establish a platform for both research and education in the field of flexible production, the RoboCup Logistics League (RCLL) competition was established [9]. The RCLL provides an abstracted environment that mimics a flexible production facility and serves as an attractive platform for the development and evaluation of innovative production concepts.

The GRIPS team, consisting of students and researchers from Graz University of Technology, has been actively participating in the RCLL since 2016. In [8], we have described the overarching architecture of the software utilized by our team to address the chal-

lenges posed by the RCLL, along with a more detailed exploration of the specific challenges introduced by the league. In this current paper, we present the recent advancements in the GRIPS software, which have resulted in improved productivity within the RCLL setting. In particular we report improvements related to the integration of temporal planning, advanced plan dispatching concepts and flexible intermediate skills.

2 RoboCup Logistics League

The objective of the league is to promote the advancement of intelligent production automation through the organization of robotics competitions. In a physical setting, a team of autonomous mobile robots collaborates to manufacture a variety of products by interacting with production machines. The orders for specific products, including a desired configuration and delivery schedule, are generated incrementally during the game. Configurable products are mimicked by a stack composed of a base, up to three rings, and a cap, with the number of rings indicating the product's complexity. Special machines are designated for each of the intermediate production steps, and additional pieces are required to attach some of the rings, which must be provided to the corresponding machine. The assembly of the products typically requires several stages of refinement by different machines. The complexity of a delivered product determines the number of points awarded. Details about how the participating teams tackle the competition can be found in the champion papers of the Carologistics [6] and GRIPS [10], together with [8] for a detailed overview of the different challenges presented by the league.

3 Robot platform

Since quick and reliable manipulation is a key factor for an efficient production process, a new gripper system shown in Fig. 1 was developed. All processing related to the gripping procedure is now handled by a programmable logic controller (PLC), which communicates with the Mid-Level of the Software Architecture (see Sect. 4.2) via Open Platform Communications Unified Architecture (OPC UA). The gripper is mounted on a 3-axis positioning system that is equipped with two distance sensors, one facing to the front and one to the bottom. When picking up a product the front sensor scans a profile in the approximate area, from which the actual position of the product can then be estimated. The initial scanning area can be tuned to the accuracy of the alignment between robot and the machine and if no plausible object is found the scanning is repeated at a wider angle. The placing of the product is done in a similar fashion but using the sensor facing down.

Most problems related to grasping arise due to the robots not being identical to each other and relatively

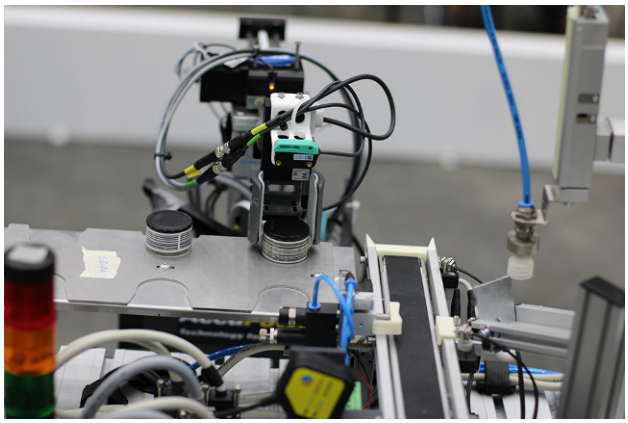


Fig. 1 Robot grasping a piece from a machine

flexible in their geometry. A slight tilt or a height difference in the robot structure can have an effect on the grasping precision, where the tolerance is in the mm range. For this purpose, the robots are checked and some calibrations are set before each competition.

4 Software architecture

In this section we present the novel techniques introducted to the software stack respect to the old version presented in [8]. The general architecture does not differ much, and it is still based on three main layers, namely (1) planning and dispatching (high-level), (2) plan refinement and execution (mid-level), (3) behavior, control and further low-level functionality. We refer the reader to [8] for a more detailed description of the integration between these parts. In order to gain more flexibility and optimally a new high-level planning layer was introduced. Instead of a greedy task allocation algorithm, temporal planning is now deployed. As a result, the system now has an improved long-term outlook, meaning that it optimizes the its production plan considering a long time-window. Other important changes regards the mid-level executive, which relies now on behaviour trees [2] instead of the BDI-System OpenPRS [7]. Further improvement has been implemented in regards of robots behaviour, like on-the-fly machine alignment.

The difference between the high-level dispatching module and the mid-level executive module implemented on the robots are that, while the former has a wider view and dispatches intermediate actions of the derived general plan to the corresponding robots, the latter refine such actions into skills.

4.1 Planning and dispatching

The Planning and Dispatching layers implements a centralized control strategy for multi-agent system in a dynamic domain. It is based on three main components: (1) a Goal Reasoner (GR), (2) a Planner (P), and (3) a Dispatching and Monitoring Module (DM).

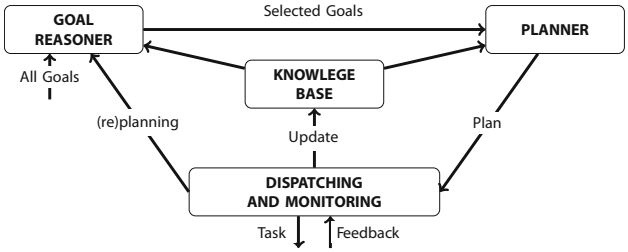


Fig. 2 Planning and Dispatching Architecture

Table 1 Temporal plan for the simplest product. Parameters (except for the agent) have been omitted for better readability

1	0.000:(getBaseFromBS r1)[51.000]
2	0.000:(bufferCapBaseFromCS r2)[89.000]
3	89.001:(getBaseFromCS r2)[52.000]
4	141.002:(deliverProducttoCS r1)[85.000]
5	226.003:(getProductFromCS r1)[52.000]
6	278.004:(deliverProductToDS r1)[73.000]

In Fig. 2, the interaction between the components is shown. The Dispatching and Monitoring module plays the role of the main controller that invokes GR and and P as well as executes the obtained plan. The plan execution is constantly monitored for issues that may require a regeneration of goals or plans, e.g. failed actions or deadline violations.

The planner is responsible for finding a plan for the goals selected by GR while minimizing its makespawn. Planning is performed using the temporal planner TFD [3] that offers numerous PDDL 2.1 features [4]. We chose temporal planning for its ability to handle temporal deadlines and coordinate multiple agents. The downside is the high planning costs. In order to address this issue for the RCLL domain, we abstracted the domain model and introduced an online goal selection. The domain is modeled in PDDL by representing the two main interactions with a station as abstract action: *get* and *delivery* only. The former retrieves a workpiece from a station after processing, while the latter delivers the workpiece to a station for processing. In the RCLL robots can not carry more than one piece at a time, so every $get \circ delivery$ action is interleaved with a *move* action. For this reason, both *get* and *delivered* already include costs and constraints related to the previous *move* action: a *get* action actually models a *move_and_get* macro-action. This significally speed-up the planning process. The result of planning is a temporal plan, which is represented by the schedule σ, formed by triples $\langle a, t_a, d_a \rangle$, where a is an action, t_a is the time when the action a needs to be started, and d_a is the duration of the action. Table 1 shows a temporal plan to build and deliver a simple product in the RCLL.

This representation poorly supports action dispatching, since it is not easy to determine how delays in the execution of an action, affect the other triples of the plan. To address this, we represent the plan as a temporal graph (Simple Temporal Network), which encodes the temporal dependencies and the partial order of actions, without constraining the start of actions to specific time points. An accurate estimation of action durations is crucial for appropriate planning and monitoring plan execution. The interaction time of a robot with a station has been estimated empirically. Regarding the traveling time, we precompute a matrix containing all the estimated costs for traveling between each pair of locations. Since the same path finding algorithm is used as in the robot's navigation skill, the estimated times are close to the real ones. To make the estimation even more precise, we are also considering orientation costs at the start and at the end of each movement action. By adopting this procedure we achieve an accurate estimation of actions duration that is incorporated into the PDDL domain.

As we are interested in dynamic domains, planning needs to be fast, to be able to react to changes and new opportunities in the environment in time. For this reason, GR selects the most rewarding set of goals P the planner may be able to obtain a plan for with a given time budget (e.g., 1 min). For RCLL a single goal is an individual order. In general, planning for all received orders is not possible within a reasonable time window. Thus, we follow the idea of partial satisfactory planning [11]. GR solves a relaxed version of the domain, assuming that each goal can be achieved by a single agent individually, ignoring cooperation or resource management. Giving this setting, the goals selection is formulated as a simple task allocation problem with an overall deadline. We employ the ASP solver CLINGO [5] to compute this optimization task, which consists of finding a subset of goals such that the awarded points are maximized within the given constraints. A drawback of the goal selection heuristic is that the solution of the relaxed problem may be too optimistic. Thus, the planner may not able to find a plan achieving all selected goals while respecting the given deadlines. To mitigate this we exploit parallel planning over multiple sets of goals. The sets are the original set plus sets obtained by either dropping goals or replacing goals with less complex ones. For each of this set, a planning thread is ran for 1 minute. The rational behind this approach is that in general it is more likely to find a feasible plan for simplified goal sets with a bounded time budget. Among the returned feasible plans, we select and dispatch the one with the highest reward.

The Dispatching and Monitoring module dispatches the actions of the obtained plan in time, monitors proper execution (in terms of state and time), and initiates replanning if necessary. These three events may trigger replanning: (1) failed skill execution, (2) successful execution of the actual plan, (3) impossibility to achieve a goal in time.

In this work, we use a dedicated approach to check the temporal constraint of the temporal network reprenting the plan, in order to allow an earlier detection of deadline violations. We use two types of edges: (1) action duration edges $[d_a, d_a]$ between start and end of the action a, and (2) partial order edges $[0, +\infty)$ between the end of an action and the start of the following one. Nodes corresponding to start actions are dispatched by sending the action to the robot for execution, while end nodes are dispatched when a feedback for the successful execution of the corresponding action is received. The partial order encoded through the edges is ensured by the dispatching. Unfortunately, it can not be expected that the execution in the real world perfectly respects the temporal constraint $[d_a, d_a]$ (interval between min and max time). The approach presented in [1] propagates the detected delay by fixing the execution times of already executed actions and updating the remaining deadlines using constraint propagation. In our approach, violations of such edges are tolerated, since our early deadline detection strategy immediately recognizes if such delays will cause a deadline violation in the future. Thus, we use the opposite approach, propagating back the deadlines from the goal nodes to the rest of the network, labeling each node with a relative deadline $rdl(x)$. This represents the latest time point we can dispatch that node without violating the deadline x. Starting from the *goal* vertices, we calculate the relative deadlines *rdl* of each vertex by traversing the edges in the opposite direction, subtracting the lower bound of the edge at each step. If there are more paths from a normal vertex to a *goal* vertex, we keep the more restrictive relative deadline. Every time a node is dispatched, we check if all its relative deadlines are respected, otherwise we trigger replanning. In Fig. 3 a partial result of this approach applied to a temporal graph is shown.

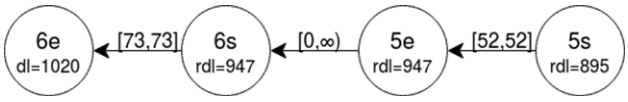


Fig. 3 The deadline propagation process on a part of a temporal network. 1020 is the final deadline for the entire network

4.2 Mid-level executive

In order to ensure reliable execution of tasks, a modification was made to the middle layer, which now utilizes Behavior Trees [2] for action execution. Each dispatched action is divided into smaller, atomic low-level skills, such as moving the gripper or placing an object on a station. This allows for the decomposition of each action into executable skills. The execution of routines is represented as a tree structure composed of different types of nodes. Control nodes are used to alter the program flow, while execution nodes initiate actions. These nodes exchange data through blackboards implemented by the Behavior Tree library. ROS action clients are used to trigger action on the PLC, the robot platform or to gather sensing information. The high-level receives notifications about the execution outcomes. If a tree fails to execute, control is returned to the high-level, and the robot waits for the next action to be assigned. The key advantage of this approach is the user-friendly nature of behavior trees, while still enabling the encoding of complex behaviors.

One notable improvement over the previous system is the introduction of on-the-fly alignment for the robot's interaction with machines. In the previous system, the robot would first navigate towards the closest zone on the correct side of the machine, followed by a separate alignment step with the conveyor. In contrast, the updated system enables the robots to initiate alignment immediately upon detecting the presence of a machine while moving towards the designated zone. For its implementation, a multiplexer-based mechanism is employed, which switches between the velocity provided by the navigation planner to the robot platform and a feedback controller utilized for precise alignments. A proportional controller was chosen as control algorithm. We can reach reference tracking and a low settling time due to the high gain of the controller and a velocity constraint. The velocity constraint is calculated based on the maximum acceptable error divergence (2 cm) and the acceleration of the move base (3 m/s²). The error calculation for the controller determines the signal for selecting the multiplexer values. Once a threshold is reached, indicating that the robot can safely align with the machine, the multiplexer switches from the move base velocity to the alignment velocity. This on-the-fly alignment approach significantly reduces skill transition delays by minimizing network interactions and tree initializations.

5 Conclusion

This paper presents the key improvements implemented by the GRIPS team in order to effectively address the challenges posed by the RCLL. These enhancements concern both the high-level decision-making component and the mid-level executive layer of the software stack utilized by the team. In terms of the high-level decision-making, a novel architecture has been developed, which incorporates a temporal action-planner, a goal reasoner, and an action dispatcher. This architecture effectively coordinates the fleet of robots and assigns and schedules appropriate actions for execution. Furthermore, these actions are subsequently transformed into behavior trees within the mid-level layer. These behavior trees incorporate additional diagnosis and monitoring techniques that are seamlessly integrated into the overall structure. In addition, a new gripper allowed for faster grasping and placing operations. The incorporation of these

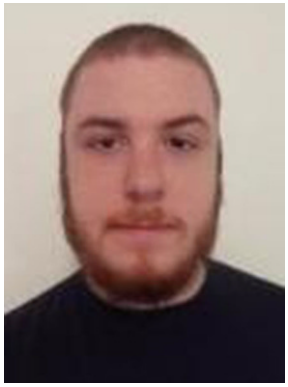
improvements has resulted in enhanced performance within the RCLL competition, as demonstrated by the team's success in winning two competitions following the adoption of the new architecture.

Acknowledgement Team GRIPS is grateful to its sponsors Knapp AG, ANEXIA Internetdienstleistungs GmbH, Magna Steyr GmbH and AccuPower GmbH.

Funding Open access funding provided by Graz University of Technology.

References

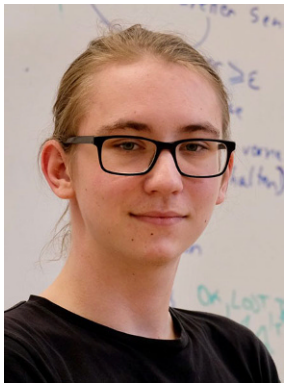
1. Castillo L, Fdez-Olivares J, González-Muñoz A (2002) A temporal constraint network based temporal planner
2. Colledanchise M, Ögren P (2018) Behavior trees in robotics and AI: An introduction. CRC Press, Boca Raton, Florida, USA
3. Eyerich P, Mattmüller R, Röger G (2009) Using the context-enhanced additive heuristic for temporal and numeric planning
4. Fox M, Long D (2003) PDDL2. 1: An extension to PDDL for expressing temporal planning domains. J Artif Intell Res 20:61–124
5. Gebser M, Kaminski R, Kaufmann B, Schaub T (2014) Clingo= asp+ control. https://doi.org/10.48550/arXiv.1405.3694 (arXiv preprint arXiv:14053694)
6. Hofmann T, Limpert N, Mataré V, Ferrein A, Lakemeyer G (2019) Winning the robocup logistics league with fast navigation, precise manipulation, and robust goal reasoning. In: Chalup S, Niemueller T, Suthakorn J, Williams MA (eds) RoboCup 2019: Robot World Cup XXIII. Springer, Cham, pp 504–516
7. Ingrand F, Chatila R, Alami R, Robert F (1996) Prs: a high level supervision and control language for autonomous mobile robots. Proceedings of IEEE International Conference on Robotics and Automation, vol 1, pp 43–49
8. Kohout P, De Bortoli M, Ludwiger J, Ulz T, Steinbauer G (2020) A multi-robot architecture for the RoboCup Logistics League. e&i Elektrotechnik Informationstechnik 137:291–296
9. Steinbauer G, Niemueller T, Karras U (2018) The RoboCup Logistics League - A Testbed for Novel Concepts in Flexible Production. Robotic Assembly – Recent Advancements and Opportunities for Challenging R&D: Workshop IEEE International Conference on Automation Science and Engineering
10. Ulz T, Ludwiger J, Steinbauer G (2019) A Robust and Flexible System Architecture for Facing the RoboCup Logistics League Challenge. In: Holz D, Genter K, Saad M, von Stryk O (eds) RoboCup 2018: Robot World Cup XXII. Springer, Cham, pp 488–499
11. Van Den Briel M, Sanchez R, Do M, Kambhampati S (2004) Effective approaches for partial satisfaction (over-subscription) planning. Proceedings of the National Conference on Artificial Intelligence (AAAI), pp 562–569



Marco De Bortoli, is a Ph.D. student at the Graz University of Technology. His field of study is planning and scheduling. He graduated in Computer Science from the University of Udine, specializing in logic programming and planning.

Peter Kohout, is a Master's student of Computer Science at the Graz University of Technology and leader of the GRIPS team in the RoboCup Logistics League.

Dominik Lampel, is a Master's student at the Graz University of Technology. He is currently finishing his Master's thesis in the area of robotics and software development and has been part of the RoboCup team GRIPS since 2022. His Master's thesis comprises a simulation that will be used in the RoboCup in the years to come.



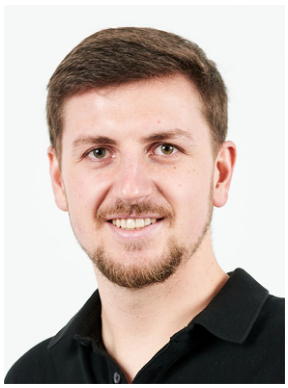
Leo Fürbaß, is an Information and Computer Engineering Master's student. He joined the RoboCup GRIPS team at the end of 2018, during the third semester of his Bachelor's studies. At the moment, he is mainly working on tasks regarding, for example, control and low-level execution.

Anna Masiero, is currently doing a Master's degree in Electrical Engineering at the Graz University of Technology. Before that, she attended a school for higher technical education in her hometown Bolzano, Italy. She joined the RoboCup GRIPS team in 2020 and has mainly worked on electrical or hardware-related issues focusing on the gripping system.

Stefan Moser, is a Master's student of Information and Computer Engineering at the Graz University of Technology and a member of the GRIPS team in the RoboCup Logistics League.

Martin Nagele, is a student of Information and Computer Engineering at the Graz University of Technology. He is currently pursuing a Master's degree with a focus on robotics and machine learning.

Tobias Frick, is currently studying Information and Computer Engineering at the Graz University of Technology specializing in robotics and computational intelligence. Previously, he attended the school for higher technical education in Rankweil with a focus on computer engineering. Alongside his studies, he is working as a software developer.



Lukas Knoflach, graduated in Information and Computer Engineering from the Graz University of Technology. He is a former member of the GRIPS team in the RoboCup Logistics League.

Gerald Steinbauer-Wagner, is an Associate Professor at the Institute for Software Technology at the Graz University of Technology. His main research interests are autonomous mobile robots, model-based diagnosis, reasoning, planning, and the RoboCup. He is particularly interested in intelligent robust control of autonomous mobile robots.