A Research Agenda for Artificial Intelligence in the Field of Flexible Production Systems

Aljosha Köcher¹, René Heesch¹, Niklas Widulle¹, Anna Nordhausen¹, Julian Putzke¹, Alexander Windmann¹, Sven Vagt¹, Oliver Niggemann¹

¹ Institute of Automation
Helmut Schmidt University, Hamburg, Germany
Email: firstName.lastName@hsu-hh.de

January 7, 2022

Abstract

Production companies face problems when it comes to quickly adapting their production control to fluctuating demands or changing requirements. Control approaches aiming to encapsulate production functions in the sense of services have shown to be promising in order to increase flexibility of Cyber-Physical Production Systems. But an existing challenge of such approaches is finding production plans based on provided functionalities for a set of requirements, especially when there is no direct (i.e., syntactic) match between demanded and provided functions. In such cases it can become complicated to find those provided functions that can be arranged into a plan satisfying the demand. While there is a variety of different approaches to production planning, flexible production poses specific requirements that are not covered by existing research. In this contribution, we first capture these requirements for flexible production environments. Afterwards, an overview of current Artificial Intelligence approaches that can be utilized in order to overcome the aforementioned challenges is given. Approaches from both symbolic AI planning as well as approaches based on Machine Learning are discussed and eventually compared against the requirements. Based on this comparison, a research agenda is derived.

1 Introduction

Today’s companies operate in an environment characterized by more frequently changing customer demands leading to shorter product life cycles and smaller batch sizes. In order to perform successfully in this market, companies have to adjust their production quickly to changing requirements. This is often referred to as changeability or adaptability of productions [WEN+07]. Changeability covers both changes on a physical as well as on control level. While the first contains ways to add or remove machines and components, the second covers adaptations in process planning or production control [ElM07]. This paper deals with the second aspect and focuses on the shop-floor.
One class of highly changeable production systems are so-called Cyber-Physical Production Systems (CPPS). CPPS with their autonomous and cooperative behavior regarding all levels of production break with traditional, more hierarchical automation approaches \cite{Mon14}. But in order to exploit the potential of an increased changeability on control level, production planning needs to be automatized for use in CPPS. AI-approaches, which are concerned with finding a sequence of actions that lead from an initial to a goal state \cite{RN16}, have long been researched for production planning, but current planning systems do not satisfy the requirements of CPPS \cite{RFN18}.

![Figure 1: Step 0, the input models for AI algorithms for flexible production systems: process models, product models and optimization criteria.](image)

Figure 1 shows the input, i.e. the needed models, for such AI algorithms. Defining these models is step 0 of each AI-solution. First of all, resources (i.e. machines, production modules, etc.) must be grouped into classes. Within each class, machines fulfill similar tasks. Each class is formally described as a process step. Process steps are defined by their inputs (e.g. intermediate products, energy, information), by their outputs and by parameters. Parameters configure the machines, e.g. timings or robot speeds. Such process steps often describe capabilities of machines \cite{ESA+19}. Second part of the input are product descriptions. This includes raw materials, intermediate products and especially the final product. Last part of the input are optimization criteria, e.g. CO2 generation, resource consumption, price or throughput. Based on these inputs, a complete configuration of the production system should be computed automatically by the AI system.

Classically, AI algorithms for such problems first solve a planning problem (see figure 2). The given process steps are put in a sequence which leads from raw materials to the final product. This includes the computation of the features
Figure 2: Step 1: First algorithmic step of an AI solution—the planning step.

of all intermediate products, e.g. dimensions, chemical properties, strain/stress properties, and the computation of optimal parameters for the steps, e.g. timings, forces used by robots or transport speeds.

Next the scheduling problem is solved (see figure 3): Now each process step is mapped onto a resource, i.e. a concrete machine. Of course, this only works for mappings within one class of resources. Furthermore, the point in time is computed at which the individual resource starts—and finishes.

Since more and more data such as sensor data, ERP (Enterprise Resource Planning) / SCADA (Supervisory Control and Data Acquisition) and digital twins are available, a third step is gaining popularity: Machine learning is used to learn parts of the input models (see figure 1).

Currently, the modeling tasks and the planning / scheduling steps are done manually by experts—leading to high efforts, long downtimes and sub-optimal results. This leads to the main research question: Can AI-algorithms be used to improve or automatize these tasks?

The first contribution of this paper is the establishment of a common notion for the necessary solution steps and the models involved. The second contribution is an assessment of the state of the art in AI with regard to these solution steps. Finally, gaps are identified and a corresponding research agenda is derived.

The remainder of this paper is structured as follows. In section 2 requirements to AI solution for CPPS are presented. Section 3.1 outlines AI-approaches for the modeling of inputs (see figure 1). The subsequent sections 3.2, 3.3 and 3.4 contain an overview of AI algorithms for the three main AI steps described above: (i) classical planning techniques (see figure 2), (ii) scheduling algorithms (see figure 3) as well as (iii) approaches employing machine learning (see figure 4). These approaches are compared with the requirements and discussed in section 4. A research agenda is derived. A summary and outlook on future research directions conclude this paper.

Figure 3: Step 2: Second algorithmic step of an AI solution—the scheduling step.
2 Requirements

We identified seven requirements an AI-approach needs to meet for CPPS.

(R1) The input models must be able to capture necessary machine capabilities. This includes raw materials and products: Here all relevant features must be covered.

(R2) The results must be optimized according to different, contradicting cost functions such as resource consumption, CO2 generation, price or throughput.

(R3) The complex dependencies between inputs, outputs and process parameters must be captured. These are continuous variables with non-linear functional dependencies. E.g. a chemical process creates specific products (e.g. viscosity) depending on the process parameters (e.g. temperature over time).

Furthermore, time can be an important process parameter set during the planning, e.g. heating a product for a given time depending on the material. 

(R4) Another challenge a planner has to handle are recurring calls of functionalities, i.e. loops in the control flow. For example in a painting process, it might be necessary to paint an area more than once to reach a certain layer thickness. A single call of the functionalities might not process the product sufficiently and a planner needs to be able to account for that.

(R5) The fifth challenge we identified concerns the explainability of AI planning. This includes the transparency of a planner’s decisions as well as the trust in its functionality. It is necessary that production workers and managers trust a novel planning approach in the same way they trust current production planning systems. Thus the decisions of the planner have to be explainable and planning errors have to be recognizable and correctable.

(R6) Implementing a planning solution requires knowledge and effort by an automation engineer. The amount of implementation effort required is dependent on the chosen solution. Therefore a further requirement is to keep this effort low to maintain economic viability.

(R7) Existing data should be used to learn parts of the necessary models. E.g. cost functions or the dependencies between inputs, output and parameters of process steps are promising candidates.
3 AI Methods

3.1 Semantic Models

Semantic models such as ontologies allow for an abstract modeling of entities, their properties and their dependencies. Furthermore, they support reasoning, i.e. the automatic computation of so-far unknown facts.

New AI-based automation approach leverages on these advantages: For a high degree of changeability on control level, production control approaches based on capabilities and skills have been proven to be more advantageous compared to classical approaches [DW19].

AI approaches use semantic models for formal descriptions of resources, their capabilities and skills. While capabilities are understood as abstract process descriptions in these approaches, skills are seen as executable functions, which can be invoked, e.g., via OPC UA. In [KHvdSF20] we presented a capability and skill model in the form of an ontology and we discussed the benefits of having an integrated semantic model of capabilities and skills. E.g. in [ESA+19], the authors described their initial ideas to use a classical planning approach in a capability-based production.

3.2 Symbolic Planning Approaches

AI planning aims to find a solution to a given problem in form of a sequence of actions leading from the initial to the goal state. Traditional AI planning methods are based on symbolic approaches such as search and logical reasoning. This section focuses on the representation of a planning problem in the Planning Domain Definition Language or as a satisfiability problem which are the basis of most planning algorithms in this field. First, the state of the art is presented before analysing whether the approaches meet the defined requirements.

3.2.1 Planning Approaches based on PDDL

The Planning Domain Definition Language (PDDL) was introduced in 1998 for the International Planning Competition (IPC) to express the physics of a domain [AHK+98]. The language follows the declarative, classical paradigm. In PDDL, a planning problem is described in two separate files, a domain and a problem file, this helps to use the same domain descriptions for different problems. A domain file contains definitions of actions whereas the materials in their initial state and the required goal state are defined in the problem file [ASN14]. Actions are defined based on the STRIPS-style (Stanford Research Institute Problem Solver [FN71]) [AHK+98]. The definition of an action contains a set of optionally typed parameters, the necessary preconditions and its effects. Preconditions and effects are expressed as predicate statements on provided parameters. Depending on the supported extensions and versions of PDDL these statements can be combined [WVN+19]. As PDDL was initiated to support classical planning, different extensions and versions of PDDL have been developed to increase the expressive power towards applying planning technology to realistic problems. With PDDL 2.1 it became possible to consider time using so-called durative actions as well as numeric properties. A qualitative model of time was added later with PDDL 3.0 [ASN14]. PDDL 3.1 is the latest PDDL...
As PDDL with all its versions and extensions is just a language to describe the planning problem and the domain, different planners were developed in course of the IPC to solve these problems using PDDL as their input language. These planners are based on different solving strategies, such as search algorithms and logic approaches or a mixture of both.

According to the IPC, the participating planners do not need to support all facets of the language [RN17]. We picked out some examples according to the previously defined requirements. The planner presented by [BCC12] is able to consider temporal as well as non-temporal preferences and to search for an optimal solution. In order to minimize the computing effort, most of the planners based on a search algorithm do not consider an action again if it is entailed in the plan already. Thus, these planners will not be able to handle recurring calls of one capability (i.e. loops, R4).

To the best of the authors’ knowledge there are no successfully tested planners, which are able to determine process parameters in course of the planning process especially when it comes to more complex real world problems (R2). As existing planners could not handle the complexity of manufacturing planning problems, [RN17] presents an approach to use automated planning in real industrial environments based on PDDL. The resulting CPPS planner should be able to react to new situations in dynamic environments (R7), but has not been validated so far.

The explainability (R5) of the planners depends on the underlying algorithm. Most planners use algorithms whose choices at each decision point are deterministic and repeatable [MDD17].

However, PDDL is not expressive enough to represent the complexity of real-world applications. Thus all previous approaches to use existing PDDL-based planners for such problems failed [RN17].

3.2.2 Planning Approaches Based on Satisfiability

The planning approaches introduced in the previous subsection are based on deduction meaning that a plan is deduced from information given as initial and goal states as well as possible actions that depend on certain preconditions and lead to certain postconditions [KS92]. The problem of Satisfiability is different in that it aims to find a valid model (i.e., a set of values) that satisfies a given formula. In the case of boolean satisfiability problems (SAT), these formulas are given in propositional logic. The basis for applying satisfiability for planning problems was laid by Kautz et al. in 1992 [KS92].

SAT is a well studied problem with many efficient solvers available. Describing a planning problem using just boolean constraints is difficult however. Satisfiability Modulo Theories is an extension of the boolean SAT problem in which parts of the propositional formula can be expressed using a variety of different so-called theories such as integers, arrays or real numbers [BT18]. This makes it much more suited towards industrial applications. A large number of well established solvers for SMT exist (e.g., [dMB08], [CGSS13]). These are most often used for applications in software verification [BDW18] and automated theorem proving. While interfacing with these solvers is usually done through APIs, a standardized language for SMT exists as a part of the smt-lib specification [BdMR+11]. Some modern solvers include tools for optimization. It is clear that being able to additionally search for an optimal solution within
the space of all valid solutions is beneficial for our application. This extension of SMT is being referred to as optimization modulo theory (OMT) [LGÁT20].

Central to using satisfiability as an approach to planning is how to formulate the problem in a suitable way. An initial formalism for a limited scenario is given in [KS92], encoding entities as binary variables, actions as predicates and time as integers. The advantage of this approach is that adding additional constraints to the solution - not only to initial and goal state - is rather straightforward. Leofante et al. have a more recent approach using OMT and are focusing on production planning. Here the state space is encoded using real valued variables. The possible movements are kept in a transition matrix and the goal is a set of state variables. Together this is a bounded symbolic reachability problem. In order to find an optimal solution it is combined with a reward function specific to the given task. Leofante et al. also provided an implementation to use PDDL to describe a problem and then subsequently solve it using OMT [LGÁT20]. An overview of different modeling strategies for industrial manufacturing using OMT was developed by Roselli et al. with a focus on job shop scheduling [RBA19].

It can be stated that problems modeled in smt-lib to be solved using modern SMT-solvers cover most of the requirements from section 2. Variables for costs or quality can be easily modeled and taken as optimization criteria - given that the problems are of linear nature as most planners do not support OMT with nonlinear functions (R2).

Just like cost or quality, time can also be modeled, e.g., using real values (R2). These values can be ordered to express predecessor and successor relationships.

Just like the two previous requirements, allowing recurring capability invocations (R4) is a problem of modeling, too.

Regarding (R5), Explainable AI Planning is a term that was introduced by [MDD17] and is gaining a lot of attention. In case of SMT / OMT, various decisions of a planner have the potential to be explained for a human user. For example, in order to explain why no plan could be found, so-called unsatisfiable cores can be used [LÁN19].

3.3 Scheduling

Scheduling a well-established field in AI and production respectively [CZHL19]. Several publications cover shop-floor applications, [AI20] gives a good overview. Typical constraints in the field of production have also been modeled. Different algorithms have been evaluated [EPA00,SMV15] and approaches can also differentiate between automated and non-automated processes [AMS14]. Also different optimization criteria have been evaluated [LHN16].

3.4 Machine Learning

Machine Learning (ML) algorithms use data in order to automatically fit a function that solves a given task. In the context of planning, ML can be used to directly learn an optimal plan, to learn useful heuristics for solving a planning problem or can be used to learn functional dependencies between process steps.

In the following subsections, ML algorithms are categorized by the way they receive feedback if they perform a task correctly, presenting the state of the art.
and analyzing if the algorithms meet the defined requirements.

3.4.1 Supervised Learning

Supervised Machine Learning uses domain knowledge in form of labeled datasets to train a model. The weights of the model are adjusted iteratively regarding prior knowledge. The goal for planning tasks is to find the shortest path from an initial state to a goal state in a specific domain. Machine learning for planning problems is a long studied field, but we focus only on the most current approaches for knowledge representation. For further research we refer to [JdLF+12].

The STRIPS-HGN [STT20] and the ASNet [TTTX20] model both rely on a STRIPS description of the planning domain. The STRIPS-HGN learns domain independent heuristics on the delete relaxation of the problem (negative effects are not considered) by encoding the input graph, recursively apply message-passing and decode the graph representation to extract the heuristic value. In training the weights are learned with the Mean-Squared-Error loss against the perfect heuristic h*. ASNet transform the feature representation of the current state into a policy by processing the input through alternating layers of action and proposition modules. Due to a domain specific connectivity between the layers, the learned weights of the policy network can be shared for different problems in the domain. ASNet is learned via imitation learning, so that the policy mimics the optimal solution.

Other approaches use (Gated) Recurrent Graph Neural Networks for solving shortest path problems, such as the GGS-NN [LTBZ] or the Graph2Seq [XWW+]. These models discard the STRIPS representation and are applicable for more general tasks. The GGS-NN model consist of sequentially operating Gated Graph Neural Networks (GG-NN) modules, which each consists of a propagation module to create node representation and an output module. At each timestep, the GG-NN module outputs the node prediction for the current step and the node annotations for the next step until the goal is reached. The Graph2Seq model maps an input graph to an output sequence. The graph encoder generates a graph embedding, which is decoded sequentially by a Recurrent Neural Network like structure with a special node attention. For the training phase different graph-path examples are created.

Multiple models employ attention mechanisms based on the work of Vaswani et al. [VSP+], and Velickovic et al. [VCC+] to differentiate the importance of every neighbor in the propagation process.

For the above mentioned models it is possible to optimize the planning process regarding costs or other criteria (R2). Often these models do not search for the shortest path, but the path with the lowest cost.

Graph Neural Networks are often not interpretable by humans, hence, (R5) could lead to problems regarding the trust in the functionality. The effort (R6) to build these models might be an issue too, since the models need prior domain knowledge to work and the algorithms are not trivial to implement. Nevertheless the result of generalized domain independent solutions might compensate the effort. The evaluation of the requirements is summarized in Table 1.
3.4.2 Unsupervised Learning

Labeling a dataset is a labor-intensive process, which is why real-world datasets in production usually do not contain labels. Thus, there is a broad trend in ML to find methods to exploit knowledge in a dataset in an unsupervised manner. Most of the work on ML for planning still relies on some form of labeling. As the research community is only beginning to test the potential of unsupervised methods for planning, the approaches are not ready for CPPS yet. However, there are initial works in the domain of generalized planning that have recently addressed the dependence on labels and prior knowledge about the underlying system.

For example, planning instances can be clustered according to their similarity, i.e. whether their solutions share a common structure with an unsupervised method [JSA17]. The clusters can then be used to assign planning instances to a fitting generalized plan.

Rather than using ML to learn an optimal plan directly, it can be used to learn general heuristics that simplify the problem and thus help to solve the problem more efficiently. By automatically labeling states on whether they are alive or unsolvable, a mixed integer linear program can be developed [FCGP19]. As the method automatically labels the states by brute force, it only works on problems with a small number of reachable states. Building on that work, a qualitative numerical planning problem can be learned, which can then be solved using an SAT planner more efficiently [GBH21].

A key challenge is to detect states that are unsolvable. By automatically labeling states on a few small instances of the planning domain with an exhaustive exploration, a heuristic can be learned that helps detecting similar states in more complex planning domains [SFS21].

Most of the requirements formulated in Section 3 cannot be fulfilled. This does not mean that unsupervised methods are fundamentally inapplicable to planning for production—on the contrary, conventional methods can benefit from the use of less supervision, as the effort to deploy a planning solution decreases. However, current methods only work for small planning problems, thus substantial research effort is required to develop production-ready unsupervised ML models.

3.4.3 Reinforcement Learning

Reinforcement Learning (RL) is regarded as the third category of machine learning. The goal is for an agent to learn the optimal policy to maximize the rewards it receives [KL19].

With respect to the planning of cyber-physical production systems, only partial or no methodological approaches to modeling them through RL are currently known. Nevertheless, RL offers a potential to solve complex and dynamic decision problems as an alternative to mathematical approaches [KKT+21].

In particular, (i) applications with a limited scope (in terms of the number of states and actions), (ii) responsive real-time decision systems, (iii) complex environments that can hardly be described in detail, and (iv) abundant or easily generated training data, are useful features. [KL19]

Kuhnle et al., due to robustness, use a Trust Region Policy Optimization (TRPO) agent to study two production problems in a system of eight machines:
Table 1: Comparison of different approaches to planning in skill-based production

|                | R1 | R2 | R3 | R4 | R5 | R6 | R7 |
|----------------|----|----|----|----|----|----|----|
| **Step 0: Semantics** | ●  | ●  | ○  | ○  | ●  | ○  | ○  |
| **Step 1: Planning**      | ●  | ●  | ○  | ○  | ●  | ●  | ○  |
| **Step 2: Scheduling**     | ●  | ●  | ●  | ●  | ●  | ●  | ○  |
| **Step 3: Machine Learning** | ○  | ○  | ○  | ○  | ○  | ○  | ●  |

order sequencing and route planning. By giving dense reward after each complete iteration of state, action, and reward, good results in detecting valid and invalid actions can be achieved through a lower training cost. [KKT+21]

A different approach is taken by Eysenbach et al, who use the so-called Search on the Replay Buffer (SoRB) method to learn a goal-conditioned policy and afterwards apply graph search to plan high-dimensional tasks over longer time horizons. The approach was tested using a complex visual navigation task. Using automatic wayfinding by observations in the replay buffer and subsequent graph search, a reasonable sequence of waypoints can then be planned. [ESL19]

Rivlin et al. show that planning strategies involving graph neural networks can be learned that are generalizable. No heuristics or existing solutions are considered here. By creating a state-goal graph, the learning process is improved and general applicability in longer instances is enabled. [RHK20]

Due to the "trial and error" function of RL and the linkage with different neural networks, there is a high flexibility with respect to the mentioned requirements (R2, R7). The flexibility arises with the number of learned states and possible actions of the agent. The more simulation processes, which are used to execute as many process runs as possible in a time-saving manner, have been run, the more flexible an agent can react to changes. Since RL algorithms are created automatically and often in a kind of "black box", meeting R5 could be a challenge. Compliance with R7 depends on the means of generalization of the RL algorithm. Since an RL agent learns different ways to successfully complete a process by going through many simulation processes, it is also possible to handle loop calls (R4). Considerations of time (R3) and cost (R1) may need to be further defined in the restrictions for the agent.

4 Discussion and Research Agenda

In this section, the suitability of existing AI-algorithm for CPPS is evaluated. For this, for each of the four AI-solution steps from section 1 the suitability of the existing AI-algorithms from section 3 is evaluated with regards to the requirements presented in section 2. Table 1 shows an overview of this evaluation. Later, gaps a identified and a corresponding research agenda is derived.
4.1 Step 0: Input Models

In the following, we evaluate whether existing AI-algorithms from section 3 can be used to improve–or automatize–the creation of the necessary (input) models (see also figure 1).

R1: Semantical models and ontologies are well suited for this task (see section 3.1). Especially capability and skill model are helpful to capture required CPPS characteristics.

R2: Optimization goals can be modeled as ontologies. But important features such as pareto optimality, the weighting between contradicting goals can not be modeled well. Furthermore no standard exist how these models can be used later by planning and scheduling algorithms.

R3: Currently mainly symbolic (i.e. cardinal variables) dependencies can be used. Numerical, functional dependencies are not covered so-far. Especially process parameters are often ignored.

R4: This requirement is not relevant for this step

R5: Semantical models and ontologies have a high degree of self-explainability.

R6: The creation and maintenance of such models is a considerable effort. Modern tools can help to ease this task.

R7: Learning such models is still in its infancy.

4.2 Step 1: Planning

In the following, we evaluate whether existing AI-algorithms from section 3 can be used to improve–or automatize–the the planning step (see also figure 2).

R1: Planning algorithms in general are well suited to model production steps.

R2: Planning algorithms can optimize the result according to arbitrary goals—often with the price of increasing runtimes.

R3: While there are some extension to planning algorithm with regard to continuous inputs / outputs, they are still mainly used for pure symbolic representations. Especially process parameters are hard to integrate.

R4: Planning algorithms have problems creating such loops.

R5: The result itself can be interpreted easily but planning algorithms have problems explaining why a specific result has been chosen.

R6: Such algorithms need no support by the user.

R7: Planning algorithms are not well suited to integrate data—they stem from classical symbolic AI methods.

4.3 Step 2: Scheduling

In the following, we evaluate whether existing AI-algorithms from section 3 can be used to improve–or automatize–the scheduling step (see also figure 3).

R1: Scheduling algorithms can capture all relevant machine characteristics.

R2: Scheduling algorithms work with arbitrary optimization goals.

R3: Since on the scheduling level no continuous interdependence are used anymore (solved by the planner), scheduling algorithm can fulfill this requirement.

R4: Since on the scheduling level no loops are used anymore (solved by the planner), scheduling algorithm can fulfill this requirement.

R5: The results of scheduling algorithms can be understood rather easily.

R6: Such algorithms need no support by the user.
Scheduling algorithms normally do not need to integrate data. Sometimes the runtime of process steps are learned—which can be done in a straightforward fashion, e.g. by regression.

### 4.4 Step 3: Machine Learning

In the following, we evaluate whether existing AI-algorithms from section 3 can be used to improve—or automatize—the application of machine learning (ML) methods for the learning of models (see also figure 3).

- **R1:** So-far, no modeling of machine capabilities suited for ML exists. Nor can capability models be learned.
- **R2:** Such cost functions can be learned. In order to be used for optimization, they must be extrapolatable, i.e. they must predict cost for operation points not covered by the data. This is a challenge for most ML algorithms.
- **R3:** ML algorithms can be used well to learn such functional dependencies.
- **R4:** Learning or even detecting such loop is rather difficult for ML algorithms.
- **R5:** ML algorithms often suffer from a lack of explainability.
- **R6:** Choosing and configuring ML algorithms is challenging. But after this step, ML algorithms run rather automatically.
- **R7:** ML algorithms are made for handling data.

From these analyses, a research agenda can be developed:

**Research Question 1:** Ontologies and functional dependencies

In production, many aspects can be modeled symbolically, e.g. if a process step glues two objects, this create a third object (symbol) from two given objects (symbols). But in reality many aspects are sub-symbolic, i.e. must be modeled via functional dependencies between numerical values. E.g. the dependency between material characteristics, forces needed to drill a hole and resulting object.

While symbolic relations can be modeled rather easily, functional dependencies between continuous variables are not well covered. A solution could be an integration with ML methods. I.e. so-far we miss an integration between semantic models and ML.

**Research Question 2:** Production-enabled planning algorithms

Planning is also neither integrated with current CPPS models (e.g. capability approaches) nor with ML. Current capability models are not created with automatic planning in the mind. E.g. they often do not capture the already mentioned functional dependencies or optimization goals. What is missing are suitable algorithms. Approaches such as PDDL show rather the limitations than are a working solution.

**Research Question 3:** Explainability

The acceptance of AI-solutions depends on the explainability of the results. So-far only partial solutions, e.g. for some ML-methods or for scheduling algorithms exist.

What is needed is an explainability generation covering the whole solution chain starting with the modeling step and ending at the ML step.

**Research Question 4:** Loops

Any solution will comprise loops, e.g. repetitive process steps or re-working of suboptimal products. While some first algorithmic approaches have been developed, e.g. for planning and ML, so-far no solution on a production level exists.
5 Summary & Outlook

In this contribution, different approaches to planning in a production environment based on AI-methods were presented. In order to discuss these approaches, requirements that arise from such production systems, were defined. The approaches covered can be categorized into two main categories: While the first one contains approaches of symbolic AI such as semantic models, PDDL and SMT, the second category consists of sub-symbolic AI approaches from the field of machine learning.

While we tried to give a concise summary of the approaches and their compliance with regard to the requirements (see Tab. 1), there are certainly limitations when it comes to discussing a whole category such as PDDL as there exists a variety of individual approaches using PDDL which might fulfill (or neglect) specific requirements. Therefore, Tab. 1 can only give a coarse overview of a category which might not reflect the particularities of every individual approach of that category. In addition, we have only been able to conduct a theoretical discussion because many of the approaches do not publish their algorithms and there are no uniform benchmarks for planning within CPPS.

Overall, we see that most, if not all, ML-based approaches are in their infancy and are being tested in simplified scenarios that cannot be compared to realistic planning problems. Symbolic methods have a higher degree of maturity due to their longer research history. Nevertheless, we could not find an approach that is suitable for modern production approaches based on modular functions. To overcome these limitations, we defined a research agenda.

References

[AHK+98] Constructions Aeronautiques, Adele Howe, Craig Knoblock, ISI Drew McDermott, Ashwin Ram, Manuela Veloso, Daniel Weld, David Wilkins SRI, Anthony Barrett, Dave Christianson, et al. PDDL| The Planning Domain Definition Language. Technical report, Technical report, 1998.

[AI20] Muhammad Akbar and Takashi Irohara. Metaheuristics for the multi-task simultaneous supervision dual resource-constrained scheduling problem. Engineering Applications of Artificial Intelligence, 96:104004, 2020.

[AMS14] Alessandro Agnetis, Gianluca Murgia, and Simone Sbrilli. A job shop scheduling problem with human operators in handicraft production. International Journal of Production Research, 52(13):3820–3831, 2014.

[ASN14] Anas Anis, Wilhelm Schäfer, and Oliver Niggenmann. A comparison of modeling approaches for planning in cyber physical production systems. In Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA), pages 1–8. IEEE, 2014.

[BCC12] J Benton, Amanda Coles, and Andrew Coles. Temporal planning with preferences and time-dependent continuous costs. In
Clark Barrett, Leonardo de Moura, Silvio Ranise, Aaron Stump, and Cesare Tinelli. The SMT-LIB Initiative and the Rise of SMT. In Sharon Barner, Ian Harris, Daniel Kroening, and Orna Raz, editors, *Hardware and Software: Verification and Testing*, Lecture Notes in Computer Science, pages 3–3. Springer, 2011.

Dirk Beyer, Matthias Dangl, and Philipp Wendler. A unifying view on SMT-based software verification. *Journal of Automated Reasoning*, 60(3):299–335, 2018.

Jürgen Beyerer, Alexander Maier, and Oliver Niggemann. *Machine Learning for Cyber Physical Systems*, volume 13. Springer Berlin Heidelberg, Berlin, Heidelberg, 2021.

Clark Barrett and Cesare Tinelli. Satisfiability Modulo Theories. In Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, and Roderick Bloem, editors, *Handbook of Model Checking*, pages 305–343. Springer International Publishing, Cham, 2018.

Alessandro Cimatti, Alberto Griggio, Bastiaan Joost Schaafsma, and Roberto Sebastiani. The MathSAT5 SMT solver. In Nir Piterman and Scott A. Smolka, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, Lecture Notes in Computer Science, pages 93–107. Springer, 2013.

Zhengcai Cao, Lijie Zhou, Biao Hu, and Chengran Lin. An adaptive scheduling algorithm for dynamic jobs for dealing with the flexible job shop scheduling problem. *Business & Information Systems Engineering*, 61(3):299–309, 2019.

Leonardo de Moura and Nikolaj Bjørner. Z3: An efficient SMT solver. In C. R. Ramakrishnan and Jakob Rehof, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, Lecture Notes in Computer Science, pages 337–340. Springer, 2008.

Kirill Dorofeev and Monika Wenger. Evaluating skill-based control architecture for flexible automation systems. In *Proceedings, 2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1077–1084, Piscataway, NJ, 2019. IEEE.

Hoda A. ElMaraghy. Reconfigurable process plans for responsive manufacturing systems. In Pedro Filipe Cunha and Paul G. Maropoulos, editors, *Digital enterprise technology*, pages 35–44. Springer, New York, 2007.

Hoda ElMaraghy, Vishvas Patel, and Imed Ben Abdallah. Scheduling of manufacturing systems under dual-resource constraints using genetic algorithms. *Journal of Manufacturing Systems*, 19(3):186–201, 2000.
[ESA+19] Kathrin Evers, Jan R. Seyler, Vincent Aravantinos, Levi Lucio, and Anees Mehdi. Roadmap to Skill Based Systems Engineering. In Proceedings, 2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), pages 1093–1100, Piscataway, NJ, 2019. IEEE.

[ESL19] Benjamin Eysenbach, Ruslan Salakhutdinov, and Sergey Levine. Search on the replay buffer: Bridging planning and reinforcement learning. CoRR, 2019.

[FCGP19] Guillem Francès, Augusto B. Corrêa, Cedric Geissmann, and Florian Pommerening. Generalized Potential Heuristics for Classical Planning. In Sarit Kraus, editor, Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI-19), pages 5554–5561, California, 2019. International Joint Conferences on Artificial Intelligence.

[FN71] Richard E. Fikes and Nils J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. Artificial Intelligence, 2(3):189–208, 1971.

[GBH21] Guillem Francès, Blai Bonet, and Hector Geffner. Learning general planning policies from small examples without supervision. Proceedings of the AAAI Conference on Artificial Intelligence, 35(13):11801–11808, 2021.

[JdLF+12] Sergio Jiménez, Tomás de La Rosa, Susana Fernández, Fernando Fernández, and Daniel Borrajo. A review of machine learning for automated planning. The Knowledge Engineering Review, 27(4):433–467, 2012.

[JSA17] Javier Segovia-Aguas, Sergio Jiménez, and Anders Jonsson. Unsupervised classification of planning instances. Proceedings of the International Conference on Automated Planning and Scheduling, 27:452–460, 2017.

[KHVdSF20] Aljosha Köcher, Constantin Hildebrandt, Luis Miguel Vieira da Silva, and Alexander Fay. A formal capability and skill model for use in plug and produce scenarios. In 2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), volume 1, pages 1663–1670, 2020. ISSN: 1946-0759.

[KKT+21] Andreas Kühnle, Jan-Philipp Kaiser, Felix Theiß, Nicole Stricker, and Gisela Lanza. Designing an adaptive production control system using reinforcement learning. Journal of Intelligent Manufacturing, 32(3):855–876, 2021.

[KL19] Andreas Kühnle and Gisela Lanza. Application of reinforcement learning in production planning and control of cyber physical production systems. In Jürgen Beyerer, Christian Kühnert, and Oliver Niggemann, editors, Machine Learning for Cyber Physical Systems, pages 123–132, Berlin, Heidelberg, 2019. Springer Berlin Heidelberg.
[KS92] Henry Kautz and Bart Selman. Planning as Satisfiability. In Proceedings of the 10th European Conference on Artificial Intelligence (ECAI 92), 1992.

[LÁN+19] Francesco Leofante, Erika Ábrahám, Tim Niemueller, Gerhard Lakemeyer, and Armando Tacchella. Integrated Synthesis and Execution of Optimal Plans for Multi-Robot Systems in Logistics. Information Systems Frontiers, 21(1):87–107, 2019.

[LGÁT20] Francesco Leofante, Enrico Giunchiglia, Erika Abrahám, and Armando Tacchella. Optimal planning modulo theories. In Christian Bessiere, editor, Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020, pages 4128–4134. ijcai.org, 2020.

[LHN16] Jingyao Li, Yuan Huang, and Xinwei Niu. A branch population genetic algorithm for dual-resource constrained job shop scheduling problem. Computers & Industrial Engineering, 102:113–131, 2016.

[LTBZ] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks.

[MDD17] Maria Fox, Derek Long, and Daniele Magazzeni. Explainable Planning. In Proceedings of IJCAI-17 Workshop on Explainable Planning. 2017.

[Mon14] László Monostori. Cyber-physical Production Systems: Roots, Expectations and R&D Challenges. Procedia CIRP, 17:9–13, 2014.

[RBA19] Sabino Francesco Roselli, Kristofer Bengtsson, and Knut Akesson. SMT solvers for flexible job-shop scheduling problems: A computational analysis. In 2019 IEEE 15th International Conference on Automation Science and Engineering (CASE), pages 673–678, 2019. ISSN: 2161-8089.

[RFN18] Antje Rogalla, Alexander Fay, and Oliver Niggemann. Improved domain modeling for realistic automated planning and scheduling in discrete manufacturing. In 2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA), volume 1, pages 464–471, 2018. ISSN: 1946-0759.

[RHK20] Or Rivlin, Tamir Hazan, and Erez Karpas. Generalized planning with deep reinforcement learning. CoRR, abs/2005.02305, 2020.

[RN16] Stuart Russell and Peter Norvig. Artificial intelligence: A modern approach. Pearson, Boston, third edition, 2016.

[RN17] Antje Rogalla and Oliver Niggemann. Automated process planning for cyber-physical production systems. In 2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), pages 1–8, 2017.
[SFS21] Simon Ståhlberg, Guillem Francès, and Jendrik Seipp. Learning generalized unsolvability heuristics for classical planning. *Twenty-Ninth International Joint Conference on Artificial Intelligence*, 4:4175–4181, 2021.

[SMV15] María R. Sierra, Carlos Mencía, and Ramiro Varela. New schedule generation schemes for the job-shop problem with operators. *Journal of Intelligent Manufacturing*, 26:511–525, 2015.

[STT20] William Shen, Felipe Trevizan, and Sylvie Thiébaux. Learning domain-independent planning heuristics with hypergraph networks. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 30, pages 574–584, 2020.

[TTTX20] Sam Toyer, Felipe Trevizan, Sylvie Thiébaux, and Lexing Xie. Asnets: Deep learning for generalised planning. *Journal of Artificial Intelligence Research*, 68:1–68, 2020.

[VCC+] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks.

[VSP+] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need.

[WEN+07] H-P Wiendahl, Hoda A Elnaraghy, Peter Nyhuis, Michael F Zäh, H-H Wiendahl, Neil Duffie, and Michael Brücke. Changeable manufacturing-classification, design and operation. *CIRP annals*, 56(2):783–809, 2007.

[WVN+19] Bernhard Wally, Jiří Vyskočil, Petr Novák, Christian Huemer, Radek Šindelář, Petr Kadera, Alexandra Mazak, and Manuel Wimmer. Production planning with iec 62264 and pddl. In *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*, volume 1, pages 492–499, 2019.

[XWW+] Kun Xu, Lingfei Wu, Zhiguo Wang, Yansong Feng, Michael Witbrock, and Vadim Sheinin. Graph2seq: Graph to sequence learning with attention-based neural networks.