MDE and Learning for flexible Planning and optimized Execution of Multi-Robot Choreographies

1st Eric Wete
4th Daniel Kudenko
5th Wolfgang Nejdl
Institute of Data Science
Leibniz University of Hanover
Hanover, Germany
eric.roslin.wete.poaka@stud.uni-hannover.de

2nd Joel Greenyer
Institute for Software Engineering
FHDW Hanover
Hanover, Germany

3rd Andreas Wortmann
ISW
University of Stuttgart
Stuttgart, Germany

***Abstract*—Multi-Robot systems in automotive are safety-critical systems that consist of collaborating-aware robots and components that interact with external components, the environment, or humans at run-time. This implies a significant complexity for the system engineer to design, model, validate the system, and optimize the cycle time, including considering unexpected events at run-time. This paper addresses this challenge by describing a model-driven engineering approach that formally designs the system under the consideration of uncertainties and at run-time optimizes the system actions using learning-based approaches. We implemented this approach in an industrial-inspired case study of a spot-welding multi-robot cell. Based on the system requirements, we generate valid system strategies that consider unexpected events such as robot interruptions and failures. Considering movement and interruption time models, we implemented a reinforcement learning method to optimize system actions at run-time. We show that via simulations and learning, our approach can be used to synthesize time-efficient schedules for robot task assignments that improve the overall cycle time.**

***Index Terms*—Motion planning, Task planning, Choreography planning, Learning-based scheduling under uncertainty**

I. Introduction

In multi-robot systems, scheduling, assigning robot tasks, and planning robot motions is challenging [1]. At run-time, human interventions for repairs due to malfunctions, manufacturing process changes, or time constraints can increase the cycle time and robot cell dead times. Considering strict commissioning time constraints, it is challenging to consider uncertainties in practice.

Model-Driven Engineering (MDE) techniques and synthesis tools offer the possibility for engineers to model reactive systems and deal with the high complexity of the system requirements [2]–[4]. Recently, SPECTRA, a specification language for reactive system design, has been proposed [4]. These approaches provide first-solution insights but must be completed since they do not consider time and stochastic constraints.

This paper describes an MDE approach that defines how we compute robot trajectories during system design and generate models for system requirements and execution. First, we leverage the manufacturing process description and requirements, including the virtual model of the robot cell to derive optimal robot trajectories and the formal requirement specification. In the digital twin (DT) process [5], even already existing real production cells can be used to produce the virtual production cell, and thus considered in our approach. We implement an iterative A* algorithm to quickly generate cost-optimal robot trajectories for each robot given its working range. Through trajectory simulations, joint trajectories leading to collisions are identified and formally described along with the system requirements in the formal specification model. Second, using reactive synthesis [6], [7], we build a correct-by-construction robotic system. This represents the reactive system implementation, called controller, that encodes valid strategies w.r.t. defined requirements and under the consideration of unexpected events. The controller is executable and can be integrated into a simulation loop. Third, the system execution is optimized at run-time using reinforcement learning (RL) approaches. The RL-agent learns from simulations or experience and optimizes the decision-making system.

This paper is organized as follows. In Sect. II, we define the terminologies, foundation concepts used in this paper, and describe related works. Sect. III describes our methodology. We evaluate our approach in Sect. IV, and conclude in Sect. V.

II. Background & Related Work

A. Background

1) Choreography Planning: A robot *choreography* is defined by the joint execution of robot trajectories inside a robot cell during a manufacturing process cycle [8]. We define *choreography planning* as the task of defining the execution timeline of each robot trajectory in the robot cell.

2) Reactive Synthesis: The process of building a correct-by-construction reactive system from its temporal logic specification is called *reactive synthesis* [9].

The *linear temporal logic* defines modal temporal logic using time-modalities to specify a reactive system behavior [10]. LTL is defined over a set of atomic

propositions, logical operators $\neg$, $\vee$, and temporal operators $\mathbf{X}$ (next), $\mathbf{U}$ (until) [7]. The LTL syntax is defined as follows:

$$\varphi := p \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathbf{X}\,\varphi \mid \varphi\,\mathbf{U}\,\varphi \mid$$

where p is an atomic proposition.

A *computation* defines a sequence of truth propositions, i.e., $\rho = \rho_0\rho_1 \cdots \in 2^{AP}$, where ρ_i consists of atomic propositions that hold at time instant i.

A LTL formula φ is *satisfiable* if a computation ρ can be found so that $\rho \models \varphi$, i.e., φ is true at ρ_0.

GR(1) defines a subset of LTL having an efficient synthesis algorithm [6], [7]. A GR(1) specification is made up of (1) initial assumptions and guarantees that define initial states, (2) safety assumptions and guarantees considering the current state along with the next state, and (3) justice assumptions and guarantees that define assertions that keep infinitely often [11].

SPECTRA is a specification language for reactive systems [4]. This specification language implements the GR(1) synthesis algorithm.

3) Reinforcement Learning: Q-learning: Q-learning [12] is a model-free RL approach that does not require an environment transition model to compute an optimal policy. The RL-agent through a trial-and-error principle interacts with its environment and thereby learns how to efficiently take actions to reach its goal, e.g., a terminal state, and improves its expected accumulated reward. More interestingly, Q-learning suits our problem as it can deal with probabilistic transition-based systems.

We denote $\mathcal{S}$ as the set of states and $\mathcal{A}$ as the set of actions. Given a state $s \in \mathcal{S}$, we define $\mathcal{A}(s)$ as the subset of $\mathcal{A}$ that consists of possible actions in state s. The Q-learning algorithm applies the temporal difference learning approach based on the Bellman-equation [13], as described below.

$$Q(s,a) = (1-\alpha)\,Q(s,a) + \alpha\left(r + \gamma \max_{a' \in \mathcal{A}(s')} Q(s',a')\right) \quad (1)$$

where:

- $Q(s,a)$ is the Q-value of action $a \in \mathcal{A}(s)$ in $s \in S$
- r is the reward obtained by applying the action a
- $\alpha \in [0,1]$ is the learning rate
- γ defines the discount factor
- s' is the new state reached by applying action a.

B. Related Work

To address the large variances of robot execution time, the research work in [14] proposed an approach that provides a set of trajectories along with execution time constraints for each task [15]. At run time, and according to the time constraints, trajectories are selected, and tasks are (re)scheduled based on the uncertainty of robot movement time [16]. These approaches do not consider task dependencies or unexpected events as we do.

In DT area, the research work in [17] integrated process planning and scheduling with service-based production systems using a Deep-Q-network. The environment behavior is encoded using a Markov Decision Process model that defines action rewards, and the RL-agent model is updated according to stochastic environment events. Instead of learning correct actions, our approach leverages the GR(1) controller that provides valid, requirement-compliant actions for the RL-agent. Then, the RL-agent learns a policy to select cost-efficient actions at run time.

The paper [18] proposed an approach to learn the action costs and correlation between actions based on previous action executions. Action correlations identify actions that impact the execution time of other actions. The task planner learns to select the least correlated joint action to optimize the joint plan execution time. The approach does not cover unexpected behavior of the environment.

III. METHODOLOGY

Our development process to synthesize and execute a controller-based robot cell is illustrated in Fig. 1. We describe the process in four steps as follows.

A. Requirements Formalization

Given a robot cell description and the related informal requirements, the robotics expert models the robot cell using a robot cell simulation tool and CAD models, such as ABB RobotStudio[1]. The requirements include the robot description and the manufacturing process that describes each task and its dependencies. The robot expert designs the tasks and the robots, including their home position, the workpiece, the part positioner, and other devices and objects (robot control unit case, fences). A near-to-real robot cell is required for the next step. If the real robot cell to be designed already exists, a DT process [5] can be applied to build a virtual robot cell model.

We developed a DSL, namely RCSML [19], for the formal specification of robot cell requirements. Thus, robotics experts can construct reactive robotic systems without prior knowledge of GR(1)-based specifications. In fact, the requirement model (RCSML) and the reactive specification (SPECTRA) are derived from the robot cell virtual model. Our case study models can be found in [20]. The end user does not need to manually change the specifications. If the GR(1) specification is not realizable, the end user must amend the robot cell virtual model and eventually the requirements. The models will be automatically updated accordingly. Moreover, RCSML is an easy human-readable model that offers a high abstraction level for multi-robot cell systems. At the early design stage, and even without a simulation tool, the end users can check the informal requirement feasibility or verify some system properties beforehand.

B. Trajectories Computation

Instead of the classical A* algorithm [21], we implemented an iterative A* search algorithm that applies A* over iterations and progressively decreases the step size until a path is found. The step is initialized with the distance

[1]https://new.abb.com/products/robotics/de/robotstudio/robotstudio-desktop

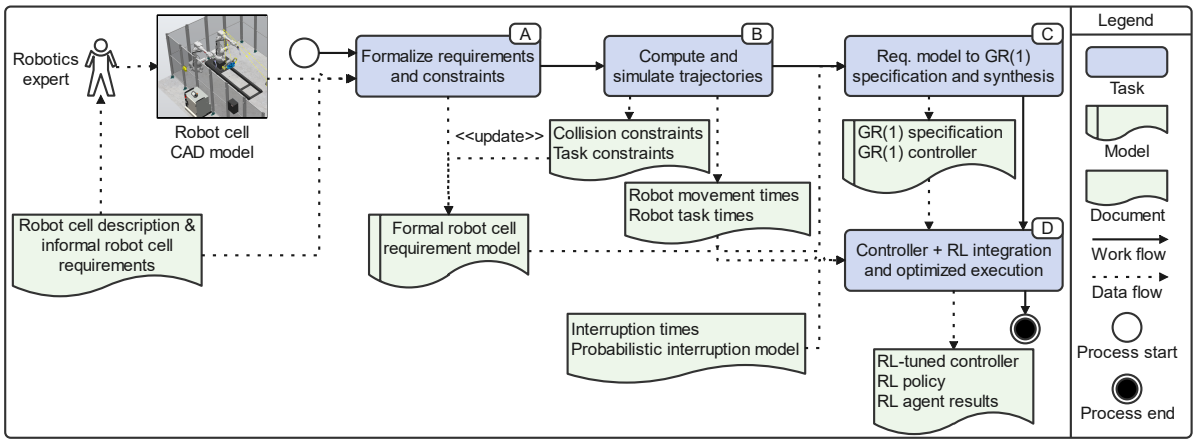

Fig. 1. The methodology development process

from the start to the final node. The step is used to find the successors of a given point in 3D space. Each iteration applies the A* algorithm to each segment with a collision. The first iteration has one segment that consists of the start and end points. At the end of an iteration, if we find a collision-free, cost-efficient path for all segments, the iteration ends. Otherwise, the next iteration applies A* on all segments with collision and a step size divided by two. For collision avoidance, we define the minimum distance between the robot and obstacles as an input (> 0). At the end of the iterations, the solution path is constructed using the paths found during the iterations.

The execution of trajectories and tasks permits us to capture the time models of the robot movements and tasks, identify potential collisions that can occur by simultaneous executions of all pairs of different robot trajectories. The identified collision risks are formally defined as *collision constraints* in the robot cell requirement model.

C. Robotic System Reactive Specification

We build a reactive controller for the robotic system. Indeed, based on the formal robot cell requirement model, we derive the GR(1) specification using SPECTRA [4] that describes the system behavior, constraints, and environment behavior. The system behavior states how the system can react or which actions can be selected to respond to the environment behavior with respect to the system requirements and constraints. The environment behavior includes uncontrollable events that may occur during the robot cell operation, e.g., robot interruptions or repairs. We leverage SPECTRA analysis and synthesis to check if the SPECTRA specification is realizable, and generate an executable GR(1) controller that implements the system strategy.

D. Integrated Controller Execution

We integrate the executable controller with an RL technique, namely Q-learning, to optimize the system strategy at run-time. Indeed, the controller is enriched with the robot movement and task time models, along with the interruption time and probabilistic interruption models. These data permit through episode iterations to evaluate the system actions provided by the GR(1) controller. Given an environment state, the RL-agent learns the optimal policy to select among the controller actions the most promising action.

IV. Results and Discussion

We experimented on a PC with Windows 10 on a x64 CPU Intel(R) Core(TM) i7-11850H, with RAM 32.0 GB.

A. Iterative A*

Our iterative A* approach finds the optimal path, if it exists, with the fewest via-points on the path from the initial to the goal node. The number of via-points is inversely proportional to the step size in the A* search. This implies fewer computations in the search algorithm (i.e., iterations over successor nodes), and thus the time and space complexity of the algorithm is improved compared with the classical A*. Moreover, the classical A* can fail to find a solution path may be because the step size is not well chosen (or too high). The user then may select a lower step that will be applied globally on the search. Our approach systematically identifies the segment where the solution cannot be found and applies a local search with a decreased step.

B. Optimized Controller Evaluation

For our case study [20], we executed 5000 cycles (number of produced parts) with some interruption probabilities observed in our car body shop. We optimized the SPECTRA controller execution with Monte Carlo Tree Search (MCTS) and Q-learning, and compared each optimized controller with the controller provided by SPECTRA. Tab. I shows the experiment results. The columns `Alg.`, `p` denotes the controller algorithm used and the interruption probability of robot task assignment. Column `Imp.` shows the cycle time

improvement in percentage compared with the SPECTRA algorithm. The cycle time-related columns `Avg.`, `Min.`, `Max`, and `Med.` show the average, minimum, maximum, and median values. The results show that the Q-learning-based controller provides the best performance in each experiment. We can say that the RL-agent can efficiently plan and execute task sequences and react to unexpected events of the environment by optimally readapting the task schedule.

TABLE I
CYCLE TIME RESULTS WITH 5000 CYCLES

Alg.	p	Avg.	Imp. (%)	Min.	Max.	Med.
Spectra		33.00		19.47	63.05	32.55
MCTS	0	31.07	5.8	17.01	59.81	30.70
QL		24.55	25.6	16.19	48.01	23.69
Spectra		35.56		19.88	177.40	32.76
MCTS	0.005	33.37	6.2	17.01	172.28	30.95
QL		21.87	38.5	16.03	170.78	16.03
Spectra		38.30		19.64	174.94	33.23
MCTS	0.010	35.75	6.7	17.67	169.04	31.20
QL		24.88	35.0	16.52	159.38	16.52

V. CONCLUSION

This paper presented an MDE approach that leverages RL-techniques for the flexible planning and optimized execution of multi-robot choreographies. Our approach supports robotics experts in correct-by-constructing robot cell systems using GR(1) specifications and RL for system execution optimization.

We consider extending our approach to additional, more expressive RCSML concepts, e.g., dynamic or conditional task dependencies. We would like to highlight the requirement inconsistencies in the RCSML editor and the robot cell simulator environment.

REFERENCES

[1] B. Gerkey and M. Mataric, "Multi-robot task allocation: analyzing the complexity and optimality of key architectures," in *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, vol. 3, Sep. 2003, pp. 3862–3868 vol.3.

[2] J. Greenyer, L. Chazette, D. Gritzner, and E. Wete, "A Scenario-Based MDE Process for Dynamic Topology Collaborative Reactive Systems - Early Virtual Prototyping of Car-to-X System Specifications," in *Joint Proceedings of the Workshops at Modellierung 2018 co-located with Modellierung 2018, Braunschweig, Germany, February 21, 2018.*, 2018, pp. 111–120. [Online]. Available: http://ceur-ws.org/Vol-2060/mekes8.pdf

[3] S. Maoz and J. O. Ringert, "On the software engineering challenges of applying reactive synthesis to robotics," in *Proceedings of the 1st International Workshop on Robotics Software Engineering, RoSE@ICSE 2018, Gothenburg, Sweden, May 28, 2018*, F. Ciccozzi, D. D. Ruscio, I. Malavolta, P. Pelliccione, and A. Wortmann, Eds. ACM, 2018, pp. 17–22. [Online]. Available: https://doi.org/10.1145/3196558.3196561

[4] ——, "Spectra: a specification language for reactive systems," *Software and Systems Modeling*, Apr 2021. [Online]. Available: https://doi.org/10.1007/s10270-021-00868-z

[5] P. Aivaliotis, Z. Arkouli, K. Georgoulias, and S. Makris, "Methodology for enabling dynamic digital twins and virtual model evolution in industrial robotics - a predictive maintenance application," *International Journal of Computer Integrated Manufacturing*, vol. 0, no. 0, pp. 1–19, 2023. [Online]. Available: https://doi.org/10.1080/0951192X.2022.2162591

[6] R. Bloem, B. Jobstmann, N. Piterman, A. Pnueli, and Y. Sa'ar, "Synthesis of Reactive(1) designs," *Journal of Computer and System Sciences*, vol. 78, no. 3, pp. 911 – 938, 2012, in Commemoration of Amir Pnueli. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0022000011000869

[7] N. Piterman, A. Pnueli, and Y. Sa'ar, "Synthesis of Reactive(1) Designs," in *Verification, Model Checking, and Abstract Interpretation*, E. A. Emerson and K. S. Namjoshi, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 364–380.

[8] M. Apostolos, M. Littman, S. Lane, D. Handelman, and J. Gelfand, "Robot choreography: An artistic-scientific connection," *Computers & Mathematics with Applications*, vol. 32, no. 1, pp. 1–4, 1996. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0898122196000818

[9] A. Pnueli and R. Rosner, "On the Synthesis of a Reactive Module," in *Proceedings of the 16th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, ser. POPL '89. New York, NY, USA: Association for Computing Machinery, 1989, p. 179–190. [Online]. Available: https://doi.org/10.1145/75277.75293

[10] A. Pnueli, "The Temporal Logic of Programs," in *18th Annual Symposium on Foundations of Computer Science, Providence, Rhode Island, USA, 31 October - 1 November 1977*. IEEE Computer Society, 1977, pp. 46–57. [Online]. Available: https://doi.org/10.1109/SFCS.1977.32

[11] S. Maoz and R. Shalom, "Inherent Vacuity for GR(1) Specifications," in *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ser. ESEC/FSE 2020. New York, NY, USA: Association for Computing Machinery, 2020, p. 99–110. [Online]. Available: https://doi.org/10.1145/3368089.3409669

[12] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3, pp. 279–292, May 1992. [Online]. Available: https://doi.org/10.1007/BF00992698

[13] E. Barron and H. Ishii, "The Bellman equation for minimizing the maximum cost," *Nonlinear Analysis: Theory, Methods & Applications*, vol. 13, no. 9, pp. 1067–1090, 1989.

[14] S. Pellegrinelli, A. Orlandini, N. Pedrocchi, A. Umbrico, and T. Tolio, "Motion planning and scheduling for human and industrial-robot collaboration," *CIRP Annals*, vol. 66, no. 1, pp. 1–4, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0007850617300951

[15] S. Pellegrinelli, F. L. Moro, N. Pedrocchi, L. Molinari Tosatti, and T. Tolio, "A probabilistic approach to workspace sharing for human–robot cooperation in assembly tasks," *CIRP Annals*, vol. 65, no. 1, pp. 57–60, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S000785061630035X

[16] M. Cialdea Mayer, A. Orlandini, and A. Umbrico, "Planning and execution with flexible timelines: a formal account," *Acta Informatica*, vol. 53, no. 6, pp. 649–680, Oct 2016. [Online]. Available: https://doi.org/10.1007/s00236-015-0252-z

[17] Z. Müller-Zhang and T. Kuhn, "A Digital Twin-based Approach Performing Integrated Process Planning and Scheduling for Service-based Production," in *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*, Sep. 2022, pp. 1–8.

[18] S. Sandrini, M. Faroni, and N. Pedrocchi, "Learning Action Duration and Synergy in Task Planning for Human-Robot Collaboration," in *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*, Sep. 2022, pp. 1–6.

[19] E. Wete, J. Greenyer, D. Kudenko, W. Nejdl, O. Flegel, and D. Eisner, "A Tool for the Automation of Efficient Multi-Robot Choreography Planning and Execution," in *Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*, ser. MODELS '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 37–41. [Online]. Available: https://doi.org/10.1145/3550356.3559090

[20] *MDE and Learning for flexible Planning and optimized Execution of Multi-Robot Choreographies*. Zenodo, Jun. 2023. [Online]. Available: https://doi.org/10.5281/zenodo.8040848

[21] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, July 1968.