Planning for Human-Robot Collaboration using Markov Decision Processes

Khansa Rekik*†, Prof. Jörg Hoffmann†, Prof. Dr.-ing Rainer Müller*, Marcel Steinmetz†
and Dr.-ing Matthias Vette-Steinkamp*
*ZeMA -Zentrum für Mechatronik und Automatisierungstechnik gemeinntzige GmbH, Saarbrücken, Germany
†Saarland University, Saarbrücken, Germany

***Abstract*—In semi-automated industrial applications, interaction between humans and robots is essential. Such interactions require some level of mutual awareness and coordination. Precisely, while interacting with humans, robots need to be aware of their state and possible future actions in order to collaborate with them and help achieve their goal more efficiently. The focus of this work is the problem of planning for Human-Robot collaboration. First, the robot actions and their dependency on the human's activity are modeled as a Markov Decision Process (MDP). Second, the instances of the model are solved using an off-the-shelf planner. Through analysis of the solutions to the model, results highlight the influence of experimental parameters such as the size of the task and the horizon on the efficiency of the solver. Finally, the deployment of the MDP on a use-case assembly process scenario inspired from an aerospace manufacturing industry is discussed.**

***Index Terms*—Human-Robot collaboration, AI planning, Markov Decision Processes**

I. Introduction

Large scale industries are constantly seeking new technologies to increase overall efficiency in the production line. To that end, attempts are continuously being made to automate production processes. One such an attempt is introducing robots to automate repetitive and non-ergonomic tasks. However, introducing fully automated solutions can, in some cases, be inefficient compared to manual or semi-automated solutions. In such cases, one can use hybrid solutions involving both humans and robots. For instance, one can use Human-Robot Collaboration (HRC) based solutions. HRC solutions embody challenges present in division of responsibilities, action disambiguation etc [1].

Identifying the intended action by the human is important for the robot to disambiguate what the former is trying to achieve, hence to flexibly support him. However this disambiguation step can be significantly simplified in scenarios where the worker workflow is known to the robot, for instance, the ordering of steps in the workflow, the probabilities of transitioning from one activity to the other, etc.

In the context of establishing a human aware collaboration, the robot task can be viewed as "a Artificial Intelligence planning task"; it is the task of selecting a goal-leading course of actions based on observations and a model of the world and action behaviour [2], [3]. In the aforementioned scenarios' category the robot has to transition across states through submitting actions until reaching the common goal. A state is affected by multiple factors namely the observations about the human worker and model of the human's workflow, the model of the environment, as well as the model of its own actions. In other terms, based on the human work-flow as well as any environmental factor that affect the decision making process, the robot can infer which actions to pursue in order to achieve the shared task.

Such dynamic can be modeled with different planning frameworks (deterministic planning, stochastic planning etc.). The challenge is to find trade-offs between accurately modeling the problem and the planning complexity [4]. Planning, as presented in [2], [4], assumes certainty about the initial state and the actions' effects. This assumption is relaxed for probabilistic planning frameworks, such as *Markov Decision Processes (MDP) planning*, where transitions are probabilistic. According to [5], one can use MDPs for simulating a rational human's acting towards a certain intention. The result of solving these models would be integrated in the robot's model and used as a reference for the observations gathered by the robot. Similarly, the work-flow of both agents can be modeled by means of MDP [6].

As an initial case study, we consider the riveting process in the aircraft assembly for a use-case scenario as it is one important use case among many others. Due to its complexity, this process can be semi-automatized in a way where the human can perform the hammering while a robot counterholds from the opposite side. This scenario is modeled as an MDP, since MDPs represent a good compromise between accuracy and complexity. On one hand, MDPs allow to model stochastic workflows. on the other hand, they allow assuming that the human workflow state is known so that no reasoning about sensing and its implications is required. Note that the methods we use are generic, thus potentially reusable for more complex scenarios.

This work is divided into five main parts. First we introduce a use-case scenario from the aircraft industry. Second we give an overview about the MDP planning background. In the third part we present the modeling of the use-case process first as an MDP followed by results of the efficiency tests performed using an off-the-shelf solver. Finally, we discuss the deployment of the model to a real demonstration.

This research was supported by ZeMA -Zentrum für Mechatronik und Automatisierungstechnik gemeinntzige GmbH through the Robotix Academy project.

II. Use-case scenario: Riveting process in aircraft assembly

The use-case scenario implemented in this work is the riveting process used for aircraft assembly. The body of an aircraft consists of different segments, which are adjusted with respect to each other employing handling systems and joined manually with rivets. Riveting requires two workers; one for counter holding and one for hammering. For ergonomical purposes and to improve production conditions and thus efficiency, we aim at contributing to the automatization of this process.

Previous work on this project, described in [7], replaced one of the workers with a robot to perform the counter-holder role during the riveting process. However, the remaining worker has to send orders to the robot of the actions to be performed through an interactive device. A more optimal solution would be like follows; the robot observes the worker, infers the current activity and plans actions to help reach the common goal.

We call *human workflow* the set of activities the worker can perform and the possible transitions between them. We restrict the human possible activities to three: {waiting, riveting, having a break}. Note that transitions are non-deterministic.

Once the robot has inferred the activity of the worker, it can decide on what actions to perform avoiding making the worker wait. The process is divided into three main parts. First the riveting points have to be scanned to determine their absolute positions with respect to the robot's coordinate system. Second, the riveting itself is performed i.e. the robot navigates between point and counter-holds while the human is hammering. Finally, points are inspected to verify their compliance with quality standards.

In the frame of this work, we assume that the scanning step has been accomplished, we focus on riveting and inspecting.

III. Background: MDP planning

We consider the formulation of [8] for a Markov Decision Process (MDP), it is represented as a tuple $\mathcal{P} = \langle S, A, T, R \rangle$ where:

- S stands for a set of state variables
- A represents a set of the agent's actions, in analogy with classical planning, one can extend this MDP formulation by adding effects $(eff(a))$ and preconditions $(pre(a))$ to actions
- T is a function of the transition probabilities $P_a(s'|s)$ for $a \in A$, $s', s \in S$, if one is in state s and performs and action a, one gets to a state s' with probability $P_a(s'|s)$,
- $\mathcal{R}$ is a reward function for executing an action a in state s

We define an MDP policy π as a function $\pi : S \rightarrow A$ that maps actions to MDP states, a the policy that maximizes the long-term expected reward is an optimal policy π^*. A reward can be discounted by means of a discount factor γ in $[0, 1]$. The role of the discount factor is to make earlier rewards advantageous. For instance, a reward n steps away is discounted by γ^n.

An MDP Horizon H is the number of actions the system will take during its life time [5]. It gives a foresight in several time steps in the future. Thus one can get a horizon-optimal policy i.e a policy that, for every initial state s_0, results in the maximal expected reward from times 0 to the size of the horizon. Note that MDPs assume the *Markov Property* [9], more explicitly, the effects of an action a taken in a state s_t do not depend on the prior history, they only depend on that state (s). In addition, in a markov decision process the dynamics of the environment are fully observable. In other terms, the state s' resulting from executing a is fully known by the system.

One can evaluate a policy thanks to the value function V. This function calculates the long-term expected reward of a policy π, it can be computed using the *Bellman equation*:

$$V_t^{\pi}(s) = R(s, \pi_t(s)) + \sum_{s' \in S} T(s, \pi_t(s), s') \cdot \gamma \cdot V_{t-1}^{\pi}(s') \quad (1)$$

Thus one can compute the optimal policy π^* like follows:

$$\pi^* = \arg\max_{a \in A} \left[R(s, \pi(s)) + \sum_{s' \in S} T(s, \pi_t(s), s') \cdot \gamma \cdot V^*(s') \right] \quad (2)$$

and:

$$V^*(s) = \max_{a \in A} \left[R(s, \pi_t(s)) + \sum_{s' \in S} T(s, \pi_t(s), s') \cdot \gamma \cdot V^*(s') \right] \quad (3)$$

Algorithms for solving MDPs: There are several ways to optimally solve MDPs namely Value-iteration and Policy-iteration algorithms. In value-iteration algorithms, one keeps improving the value function at each iteration until the value-function converges. Whereas in Policy-iteration algorithms one re-defines the policy at each step and computes the value according to the new policy until it converges. Another algorithm for planning under uncertainty is the UCT algorithm [10]. UCT is one of the representatives of Monte-Carlo Tree Search algorithms on which the planner PROST is based. According to [11], PROST implements techniques on top of the UCT skeleton to show its applicability to domain independent probabilistic planning and to adapt it to the stochastic planning context. One such a context is creating strongly connected search space. Moreover, unlike UCT, PROST detects reward locks which makes it more efficient in domains presenting such locks. Furthermore, PROST performs a Q-value initialization step which prevents initial random walks in the search space. The input language used by PROST is the Relational Dynamic Influence Diagram Language (RDDL) [12]. Conventionally, actions in MDPs do not explicitly have preconditions and effects. However, planning domain languages like PPDDL and RDDL specify these actions in a factored manner analogous to classical planning. Thus actions can have preconditions and effects. Winner of the IPPC 2011 and IPPC 2014 competitions, PROST is mainly efficient for MDP planning. To this end, we will use RDDL for the implementation of the MDP model of our use-case task and PROST for running the experiments.

IV. The proposed Model for the riveting process

In this section, the MDP model of the aforementioned riveting process is presented. The human's work-flow is embedded in the robot's model. More precisely, the activity of the human is represented as a state variable that evolves in a probabilistic fashion. The state space is composed of the robot's state variables S_R, the human's state variables S_R and the state variables relative to the riveting points S_{rp} :

$$S = S_R \times S_H \times S_{rp}$$

The robot variables contain the robot position, which is a number between one and the total number of riveting points, and weather it has stopped i.e the task has been finished. Moreover, the rivets variables contain the state of the riveting point: scanned, riveted or inspected. Also if a point has been inspected or riveted in the last time step. The human variables contain the human position and current activity: waiting, riveting or having-a-break.

Analogically to classical planning, in this model actions have preconditions and effects. The possible actions are move next or previous, counter-hold, inspect and stop. Those actions are parametrized by the considered riveting point. Preconditions restrict the applicability of the actions, they can depend on the human state variables. For instance, counter-holding is only possible if the human is waiting in the same position as the robot, this point should be scanned and not previously riveted, also the robot should not be "stopped". Once this action succeeds the state of the point switches from scanned to riveted with a certain probability.

The transition function $P_a(s'|s)$ represents the uncertainty of the output of the action performed by the robot, as well as the uncertainty entailed by the change of the activity of the human.

$$Pr_a(s'|s) = Pr(s_{t+1} = s'|s_t = s, a_t = a) \tag{4}$$

In what follows, we divide a state s into its three different components s_h, s_R and s_{rp} for respectively states variables relatives to the human, the robot and the riveting points. The transitions relative to the position of the robot are not probabilistic as they depend only on the success of the move actions which are chosen to be deterministic. They are independent of the human and the rivets' states. Note that this does not apply for cases where a move-next is performed at the last point or if a move-previous is performed at the first point.

The transitions of the state variable relative to the riveting points are probabilistic. Their transition probabilities depend on the success probabilities of either the counter-hold action and the transition probabilities of the human state or on the success probability of the inspect action.

Analogically the states relative to the human activity evolve in a probabilistic scheme. The model of the human work-flow can be seen as a sub-model of the MDP of the robot where transitions are also probabilistic. Given that the human is more likely to transition from *"watiting"* to *"riveting"* and vice-versa, it is less likely that he takes a break very often and thus the probability of *"having a break"* is lower. Note that transitioning from a *"having a break"* to *"riveting"* is not possible. In other terms, being in break the human can only go to a waiting state. The model of the human, although independent of the robot actions, influences the evolution of the rest of the state variables as well as the applicability of some actions e.g counter-hold.

The reward function is defined in a way that boosts the human workers comfort and penalizes encumbering her plan execution. As described in the following function a positive reward is assigned each time a new point, that has not been riveted previously, is riveted or a point, that has not been inspected, is inspected. A negative reward is however assigned if the robot keeps the human worker waiting or if none of the aforementioned conditions are satisfied.

V. Experiments and preliminary results

The MDP model is implemented using RDDL. The flexibility of the language and the broad range of modeling possibilities it offers allows capture the real world setting of the riveting task. However, this flexibility is constrained by the solver that is used to run models encoded in RDDL i.e. PROST. As the latter does not support all possibilities offered by RDDL in terms of modeling, hence, assumptions and simplifications need to be made while modeling the task.

One such a simplification is downgrading the models from their factored form to a ground for in which each factorized variable with a parameter x is transformed to set a variable for which each variable is an instantiation of x.

The results, calculated considering an action time limit of 0.5 s and a total time limit of 60 min, and showed in Fig.1, indicate that, PROST can time out during its heavy parsing phase. To that end, the parsing method has been modified. This has resulted in a significant improvement in performance as shown in the figure.

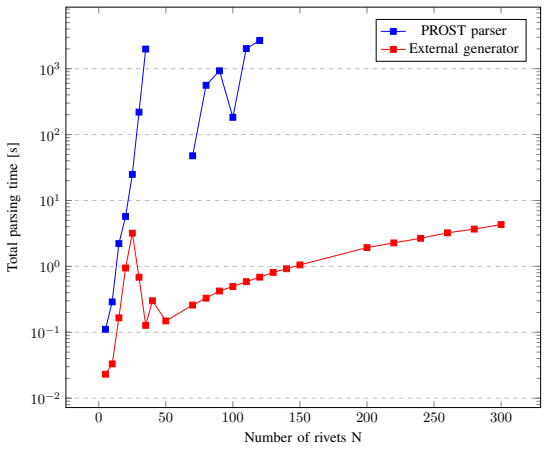

Fig. 1. Comparison of the parsing time between the internal parser of PROST and an external parser given a horizon size H equal to the size of the MDP model N.

The blue curve, representing the parsing time of PROST's conventional parser, shows a variation in the time needed for

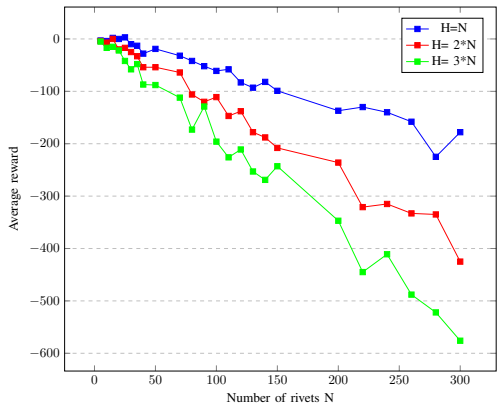

Fig. 2. Change in the average reward for different MDP model sizes N in accordance to change in the horizon size H

parsing with respect to the size of the task i.e. the number of rivets. The whole experiment frequently times out during the parsing namely for tasks with sizes between 40 and 60, and above 120.

As the reward functions is parameter that drives the MDP model, we evaluate the performance of the use-case models solved by PROST in terms of average rewards. In Fig.2, we show the reward values for different horizon sizes with respect to the number of rivets. The rewards have negative values as each action that does not result in a new riveted or inspected point is penalized. In this analysis, it was shown that the solver can solve models with sizes up to 300 riveting points within the time limits imposed. Furthermore, the larger the horizon is, the lower the reward values are for a fixed number of rivets. This is due to the accumulated uncertainty. Note that, having a lower reward for a larger horizon does not presume that a restricted horizon is better. The horizon is not only the foresight but also the number or actions to be performed. More precisely, for a 5 rivets task, if $H = N$ then the process would stop after submitting 5 actions, whereas with $H = 3 * N$ the process would stop after 15 actions. As a result, reward values are lower for larger horizons for there is room for submitting more actions, accumulating more uncertainties and getting more penalized.

VI. Deployment to the use-case

The aforementioned model is deployed to a demonstrator of the use-case scenario. The demonstrator consists of the a Universal Robot 10, a lifting unit enabling the transportation of the robot and the sections to be riveted. The section of the demonstrator contains thousands of rivets, thus having a model with hundreds of rivets it's practical for that it helps limit the number of runs launches.

Using a Markov decision process in a real application requires assuming that sensors use for observations are noise free. For the riveting process scenario sensors are used to update the state variables as actions are being executed. Theoretically each state variable should have a way to be validated through the feedback of the sensors. As discussed previously, there are variables relative to the robot, to the human and to the riveting points. In order to observe those variables three external sensors are used in addition to the robot's internal sensors. Detecting the position of the robot is performed by means of the robot's internal encoder. The external sensors are a safety mat for detecting the presence of the human in the workplace, a force-torque sensor to detect if the human is waiting or riveting and also if the rivet has been successfully riveted and a laser scanner to validate the inspection.

Furthermore, bringing the models to practical use requires replacing the RDDLSim server with the server of the real demonstrator. The latter handles the sensor data as well as the deployment of the actions by the robot.

For the tests run on the robot, it is possible to use an input model describing a task with a maximum size of 240 riveting points. At first, PROST is called once, its input is the initial state of the demonstrator and a horizon size of $3 \times N$ for a model of N points. Once an action is submitted, it is sent to the server. An internal clock of the server waits for six seconds which is the time needed for the longest action to be performed. After this period the server reads the new state s' and sends it to the solver. The stopping criterion is the end of the task which is reached once the state variable R-stopped becomes true. However, as the horizon values considered for testing are relatively small, it is unlikely to finish the task within the horizon.

VII. Discussion

In the simulated context, it was shown that a large horizon allows a higher number of actions to be submitted. However, with large horizons, the short term reward is not as maximized as it can be with a restricted horizon. In other words, the size of the horizon influences the decision making in terms of prioritizing actions that give a higher accumulated reward over several decisions instead of maximizing short term rewards. It is important to mention that a different level of flexibility is gained using planning tools in comparison to rule-based or fixed controllers namely the possibility of looking several steps ahead to gather information useful for making the decision.

In the real context, several test have been performed from which one can only detect salient problems, therefore no actual evaluation has been made so far. A further investigation is considered for future work. Nevertheless, some preliminary conclusions and remarks can be done.

On one hand, through the deployment of the model to the real scenario, multiple flows in translating the behavior established in the simulated environment has been detected. For instance, assuming that sensors are noise free has resulted in problems in detecting the current activity of the human (waiting vs. riveting). More precisely, testing the sensor off load shows already a significant variation in the force values returned although no force is applied. This makes it challenging to opt for a specific threshold.

Moreover, using the external parser alleviated the pre-computational but limits the horizon size which makes it almost not possible to finish the whole process.

On the other hand, In comparison with the semi-automated version of the process where the human needs to use an interactive device and guide the robot, our version is flexible. More precisely, one can adapt the time of the counter-holding based on the pace of the worker (beginner or expert).

VIII. Conclusion

This work tackles Human-Robot Collaboration from an MDP planning perspective for a simple industrial scenario. This application represents a step towards deploying non-deterministic planning tools in real life settings. In our case the human model is known to the robot which facilitates deciding on what actions to perform. Whereas if the human has a wider range of activities that are not necessarily explicitly cited in the robot's MDP, the decision making would be more challenging.

In future work, we intend to expand our results to account for more industrial like settings i.e. more uncertainty about the human activity and environmental state.

Acknowledgment

We thank our colleagues from Loria Nancy in France who provided insight and expertise that greatly assisted the research.

References

[1] B. Hayes and B. Scassellati, "Challenges in shared-environment human-robot collaboration," *learning*, vol. 8, no. 9, 2013.

[2] J. Hoffmann, "Everything you always wanted to know about planning," in *Annual Conference on Artificial Intelligence*. Springer, 2011, pp. 1–13.

[3] J. A. Hendler, A. Tate, and M. Drummond, "AI planning: Systems and techniques," *AI magazine*, vol. 11, no. 2, p. 61, 1990.

[4] M. Ghallab, D. Nau, and P. Traverso, *Automated Planning: theory and practice*. Elsevier, 2004.

[5] A.-B. Karami, "Modèles décisionnels d'interaction homme-robot," Ph.D. dissertation, Université de Caen, 2011.

[6] B. Bakker, Z. Zivkovic, and B. Krose, "Hierarchical dynamic programming for robot path planning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2005, pp. 2756–2761.

[7] R. Mueller, M. Vette-Steinkamp, A. Kanso, and T. Masiak, "Collaboration in a hybrid team of human and robot for improving working conditions in an aircraft riveting process," SAE Technical Paper, Tech. Rep., 2019.

[8] M. Ramírez and H. Geffner, "Goal recognition over pomdps: Inferring the intention of a pomdp agent," in *Twenty-Second International Joint Conference on Artificial Intelligence IJCAI*, 2011.

[9] L. D. Pitt, "A markov property for gaussian processes with a multi-dimensional parameter," *Archive for Rational Mechanics and Analysis*, vol. 43, no. 5, pp. 367–391, 1971.

[10] L. Kocsis and C. Szepesvári, "Bandit based monte-carlo planning," in *European conference on machine learning*. Springer, 2006, pp. 282–293.

[11] T. Keller and P. Eyerich, "Prost: Probabilistic planning based on uct." in *International Conference on Automated Planning and Scheduling ICAPS*, 2012.

[12] S. Sanner, "Relational dynamic influence diagram language (rddl): Language description," *Unpublished ms. Australian National University*, p. 32, 2010.