Dario Antonelli
Associate Professor
Politecnico di Torino – Torino
DIGEP
Italy

Qingfei Zeng
PhD student
Tongji University – Shanghai
School of Mechnical and Engineering
China

Khurshid Aliev
Research Assistant
Politecnico di Torino – Torino
DISEG
Italy

Xuemei Liu
Professor
Tongji University – Shanghai
School of Mechnical and Engineering
China

Robust Assembly Sequence Generation in a Human-Robot Collaborative Workcell by Reinforcement Learning

Human-Robot Collaborative (HRC) workcells could enhance the inclusive employment of human workers regardless their force or skills. Collaborative robots not only substitute humans in dangerous and heavy tasks, but also make the related processes within the reach of all workers, overcoming lack of skills and physical limitations. To enable the full exploitation of collaborative robots traditional robot programming must be overcome. Reduction of robot programming time and worker cognitive effort during the job become compelling requirements to be satisfied. Reinforcement learning (RL) plays a core role to allow robot to adapt to a changing and unstructured environment and to human undependable execution of repetitive tasks. The paper focuses on the utilization of RL to allow a robust industrial assembly process in a HRC workcell. The result of the study is a method for the online generation of robot assembly task sequence that adapts to the unpredictable and inconstant behavior of the human co-workers. The method is presented with the help of a benchmark case study.

Keywords: *Reinforcement Learning, Machine Learning, Task-based Robot Programming, Human-Robot Collaboration, Markov Decision Process, Assembly.*

1. INTRODUCTION

Assembly process impacts both the quality of product and the process efficiency. The challenge is transferring concepts sedimented in mass production to small assembly through the introduction of innovative techno–logies in robot automation [1].

Human-Robot Collaborative workcells (HRC) emp–loy collaborative robots, light weight, highly flexible, easy to program and intrinsically safe. They can comply with small batch productions to meet the challenges of short-term production [2].

The ISO Technical Standard 15066 defines the safety requirements for collaborative industrial robot systems and the work environment. HRC has been clas–sified as:

- safety-rated monitored stop (temporal and spatial separation);
- hand-guiding (temporal separation);
- speed and separation monitoring (spatial separa–tion);
- power and force limiting (workspace sharing).

The last class represents the most complete and challenging kind of human-robot collaboration and it is the one addressed in present study (Figure 1).

Despite the advantages of HRC, its full exploitation in actual industrial workplaces is limited. What hinder the transition from manual assembly to HRC assembly, apart from safety concerns, is the cognitive difference between human and robot. Robot follows rigidly the assigned task sequence and cannot adjust its operations to the actions of the human counterpart. While the human adapts his way of working to the partner, the robot has no empathy, therefore can execute only the assembly task in the programmed order.

The rigidity of the robot in following the planned program hinders the collaboration with the humans that carry all the burden of cognitive effort in adapting to the automaton. Therefore, working with a collaborative robot induce a stress state considerably greater than working with other human teammates [3].

What is worst, robot has no fault tolerance, therefore if some assembly operation is incorrect it stops the process and cannot try to recover to the correct course of action.

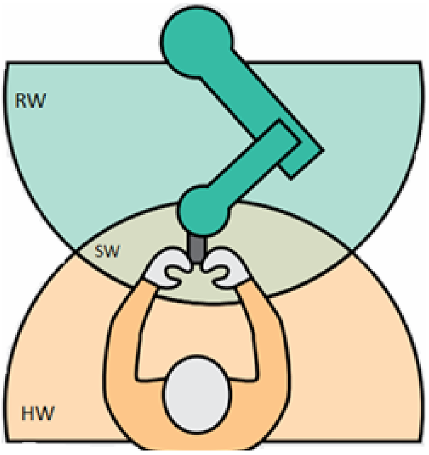


Figure 1. HRC workspace sharing: RW, HW and SW are respectively robot, human and shared workspace

Received: January 2021, Accepted: July 2021
Correspondence to: Prof. Dario Antonelli, Politecnico di Torino, Department of Management and Production Engineering, Corso Duca degli Abruzzi, 24 - 10129 Torino, Italy, E-mail: dario.antonelli@polito.it
doi:10.5937/fme2104851A
© Faculty of Mechanical Engineering, Belgrade. All rights reserved

Adding flexibility, dependability and even empathy to a collaborative robot is therefore an objective of research in HRC. Reinforcement Learning (RL) has been extensively adopted as an outstanding method to simplify and empower robot programming, making it concurrently more flexible and fault tolerant [4].

In present study, a model-free RL algorithm is used to guide the robot operations at high level, allowing the robot to adapt the assembly sequence if its human par–tner changes the order of the operations. The robot be–comes able to plan and follow a new assembly sequence if it doesn't prevent the completion of the job, perhaps at the price of a limited waste of time. Since in small productions time constraints are not as strict as in mass productions, provided that product quality and work safety be assured, a slowdown is preferrable to a protective stop. The method is explained and applied to a case study to show its effectiveness.

2. STRUCTURING THE ASSEMBLY SEQUENCE

To assist the transition from manual to HRC assembly, [5] proposes to exploit a hierarchical model of the assembly job obtained by decomposing the assembly sequence. A slightly modified model is used here (Figure 2). Assembly job is decomposed in tasks, then in operations that eventually are split down in actions.

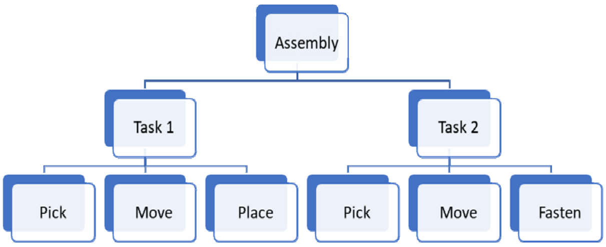


Figure 2. Hierarchical model of assembly job, tasks and operations

Assembly tasks conventionally identify the assembly of 2 or more parts, in a self-contained, comprehensive way, independent from other tasks. Individual operations are generic assembly actions and are used as a building block in different tasks. Operations must be further decomposed in specific robot actions, like open or close the gripper, move from a point to another, etc.

At the task level, the robot and the human are both involved. At the operation level, it is possible to have robot or human working in separate working areas (speed and separation monitoring) or together in operations that need their collaborative interaction in the shared workspace (power and force limiting). It must be stated that some tasks can be executed by the robot alone without human support. Fully automated tasks are not present in the experiment of section 5 as they don't present interaction issues, i.e., the robot can stick to the optimal assembly strategy defined at the beginning.

Having the Assembly Structure, it is possible to build a method to program the assembly robot adopting force and position control [6]. Note that also high-level robot programming is called task-level, but the term 'task' is used here with a different meaning.

Another outcome of the decomposition of the Assembly Structure is that now it is possible to associate an execution time to every action by using the predetermined time method systems (PTMS) [7].

It is important to remark that tasks, operations and actions are different concepts. Actions are equivalent to trajectory-level lines of code. Operations are task-level sequences of actions and constitute a set of primitives to be used by all similar tasks. Tasks are specific instances of the assembly diagram and their sequence is usually optimized respecting time and functional constraints. In industrial productions often tasks are further clustered in in macro-tasks.

In manual assembly every trained worker is able to execute all the assembly operations in a task and needs to know only the task sequence and the operation's parameters, e.g. position and number of welding points on a metal sheet. In small productions, when more than one worker is assigned to the task, the role of everyone is not defined and can be agreed among the components of the team. In HRC the role of human and robots are distinct: robot executes dangerous, heavy operations, human the ones that require dexterity or movements outside the robot work-area.

There are several methods to build an assembly task sequence. In present study the Hierarchical Task Analysis is adopted, a method developed in the context of ergonomic studies [8].

It has already been successfully extended to the task assignment inside a HRC team, using an expert guide [9] or a Machine Learning classification [10]. In the next section, starting from a given task assignment to both robots and humans, the task sequence planning problem is addressed.

3. FROM THE ASSEMBLY SEQUENCE TO THE MARKOV DECISION PROCESS

In literature, there are several solutions to the task planning problem, that is to find the optimal task sequence that minimize a cost objective, usually completion time, subject to a number of constraints. Authors of [11] proposed a simple simulated annealing (SA) algorithm to generate the optimal assembly sequence. Several capability variables were considered to obtain the best assembly sequence. Tseng and others [12] combined the factory information with the evaluation of assembly sequence scheme and achieved the best assembly sequence by genetic algorithm (GA). Authors [13] further improved the performance of the GA algorithm.

The above-mentioned algorithms work in a deterministic assembly workcell where the robot or even the human follow the planned task sequence. In manual assembly, a degree of uncertainty is present as it happens that an operator follows a different task sequence, either because he knows that it is equivalent to the one planned, or because of a small fault, often with negligible consequences on the completion time.

What in manual workcell would be just a source of variability in the cycle time, in an HRC workcell would cause the halt of the process as the robot could not be able to adapt to the change. To allow a greater degree of flexibility, the robot should be enabled to find an alternative assembly sequence with respect to the original optimal sequence. The new sequence can be equivalent or worse than the original one, but it would guarantee the achievement of the assembly goal [14-16].

To be able to adopt an alternative sequence, it is necessary to have a representation of the set of all the feasible assembly tasks. This set is formalized as a Markov Decision Process (MDP) and is used to train a RL algorithm that, for every considered task, will suggest a completion task sequence, hopefully the best one. The robot therefore will no more chose a predetermined task but will adapt its behaviour to the human partner.

Assembly process is a collection of a set of states (*S*), events (*V*) and relations (*R*). Here, *S* defines the individual tasks, *V* drives the progress of the assembly process from one step to another. *R* specifies the effect of a given event S_m on a given state V_t progressing the assembly process [17].

A state S_t has the Markov property if and only if respect (1):

$$P\left[S_{t+1} \middle| S_t\right] = P\left[S_{t+1} \middle| S_t, \ldots, S_1\right] \quad (1)$$

P is the state transition matrix, which can describe the transition probability of two states and reflect the uncertainty of the system. A MDP is a 5-tuple (S, A, P, R, γ), where: S is a finite set of states, A is a finite set of actions, P is the state transition matrix (2), R is the reward function (3) and γ is the discount rate.

$$P_{SS'}^{a} = P\left[S_{t+1} = s' \middle| S_t = s, A_t = a\right] \quad (2)$$

$$R_{SS'}^{a} = E\left[R_{t+1} \middle| S_t = s, A_t = a, S_{t+1} = s'\right] \quad (3)$$

For any MDP, there is an optimal policy π* better than or equal to other policies. In present study MDP represents the set of all feasible tasks and the optimal policy corresponds to the solution of the task planning problem.

4. Q-LEARNING OF THE ASSEMBLY SEQUENCE

RL is a behavioural decision-making method, which is widely used in the field of artificial intelligence. RL does the training by maximizing the value function of the rewards obtained in the crossed states. It constantly adjusts the agent's behaviour to gather the maximum cumulative reward. The action sequence is called policy and it is updated through a continuous interaction with the environment. Because the agent optimizes the policy by trial and error exploration, it can learn from unknown environment. Policy is the mapping relationship between state space *s* and agent action space *a* of the system. If the policy is stochastic, a probability distribution of action selection for agents is provided.

The task sequence in an assembly problem is defined and deterministic, therefore a model-free algorithm appears unnecessary. The basic assumption of present study is that the human worker could decide to execute an unplanned state. From the viewpoint of the robot it is the same as a stochastic policy, with a small, but not null, probability that some actions lead to unexpected states.

The state-value function for a policy *π* is used to estimate the cumulative future rewards (4).

$$V_\pi(\mathrm{s}) = \mathrm{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s\right] \quad (4)$$

The action-value (Q) function for a *π* is similarly used to estimate the cumulative future rewards (5).

$$Q_\pi(\mathrm{s,a}) = \mathrm{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s, A_t = a\right] \quad (5)$$

The optimal state-value function is (6):

$$V_*(s) = \max_\pi V_\pi(s) \quad (6)$$

The optimal policy π* is estimated by the optimal value function v_* as (7):

$$\pi_*(\mathrm{s}) = \arg\max_{a \in A} \sum_{s' \in S} P_{SS'}^{a}\left(R_{SS'}^{a} + \gamma V_*(s')\right) \quad (7)$$

Formula (7) requires the knowledge of the optimal state value function. When the system is model-free, the optimal policy cannot be obtained. When the action value function *Q*(*s,a*) is known, the optimal policy derives from (8):

$$\pi_*(\mathrm{s}) = \arg\max_{a \in A} Q_*(s,a) \quad (8)$$

Where *Q∗* is the optimal action value function (9):

$$Q_*(s,a) = \max_\pi Q_\pi(s,a) \quad (9)$$

Q-learning is a model-free RL family of algorithms that learns an approximator for the optimal action-value function based on the Bellman equation. Optimization is performed off-policy, therefore uses data collected at any point during training. The first proposal of a Q-learning algorithm is due to [18].

An initial estimate of Q value is updated iteratively in (10) with a learning rate *α*:

$$Q(s,a) = Q(s,a) + \alpha\left(r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right) \quad (10)$$

When Q-learning updates Q-value, it directly uses the $\max_{a'} Q(s',a')$, with a difference between behaviour and target policy (off-policy). The agent uses ε-greedy strategy to select the action. In this way it is possible to balance the exploration of the solution space with the exploitation of the best strategy.

5. EXPERIMENTAL SETUP

The experimental setup consists of UR3 robot equipped with an OnRobot 2-finger gripper and is shown in Fig 3. Several different and complementary human-robot interaction interfaces have been implemented and are present in the work area, ranging from Optirack motion capture in combination with a pointer equipped with markers, a gesture communication device based on Leap Motion hand tracking or a task selection menu accessed through a touch display. The strengths and weaknesses of the interfaces are outside the scope of present study and since now they will be generally referred as Human-Robot Interface (HRI).

The case study is composed of human operator, light weight collaborative robot and a desk corresponding to the shared workspace. On the table both robot and human worker can access all necessary components for assembly, such as base, flanges, bolts and nuts (Figure 3). The experiment is similar to the one described in [19], with the difference that in their work the focus was on the robust execution of operations despite the presence of disturbances. In present work the focus is moved to higher abstraction level, regarding the sequence of tasks and assuming the neat execution of the single operations.

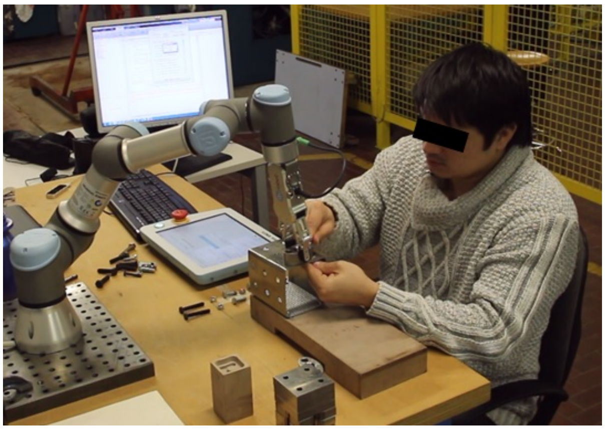
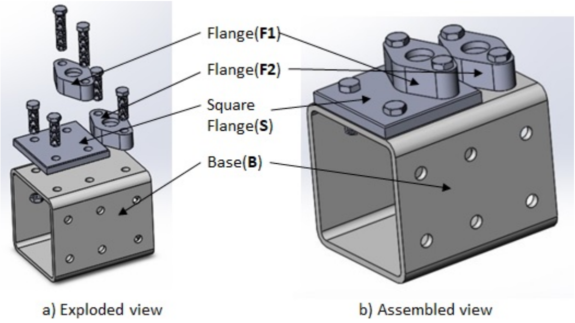

[redacted]

Figure 3. Experimental setup

The case study is made of two flanges mounted on a base. One flange is a sub-assemble of a square and an oval flange. In Figure 4 the components are presented in exploded and in assembled view. Figure 5 shows the assembly diagram. The two flanges named F1 and F2 in figure are identical. Possible positions for the flanges on the base are 1, 2, 3, corresponding to the 3 rows of holes, starting from the left edge.



a) Exploded view

b) Assembled view

Figure 4. Case study in a) exploded and b) assembled view

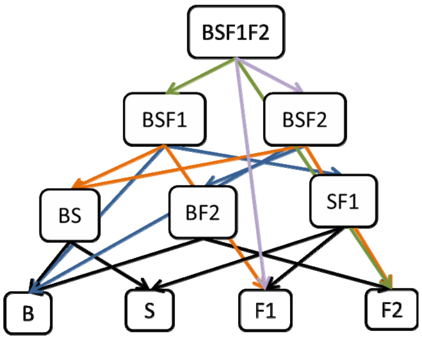


Figure 5. Assembly tree of the case study

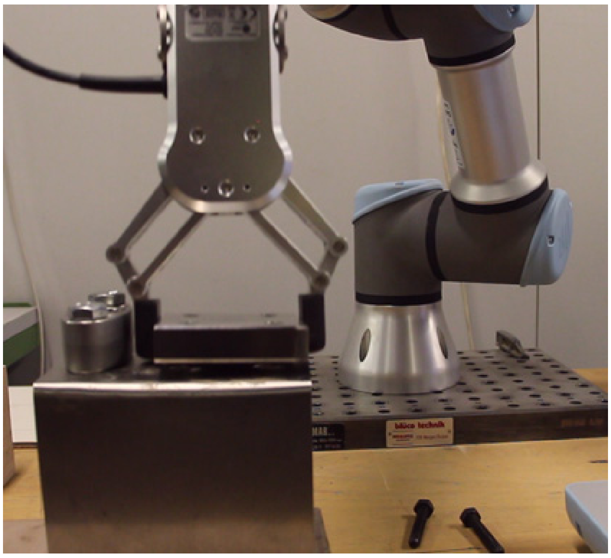


Figure 6. Robot mounts the square flange on the base by executing a task-oriented program after receiving the indication of the target position by the operator

The case study, despite its simplicity, allows to structure the assembly sequence by introducing sub-assembled groups. The symmetry of the base allows more configurations that are equivalent, in view of the final objective. This remark is apparent to humans but not to robots. Therefore, the human can start putting flanges starting from the right or from the left, indifferently. In addition to this fact, human has two choices for the position of flange F2, on the first or the second row of holes on the square. Only one position is correct as the other is a mistake, although it doesn't violate any assembly constraint.

The collaborative mechanism has been devised as follows: the robot picks a flange, move it to a position indicated by the operator and hold it while the human fasten the flange to the base by screwing the hexagonal bolts. The robot actions have been programmed on a Universal Robots UR3 as task-oriented programs: the target position of the flange is indicated by the human through HRI during the execution of the job (see Figure 6). A snapshot of a collaborative situation is taken in Figure 7, during the flange joining.

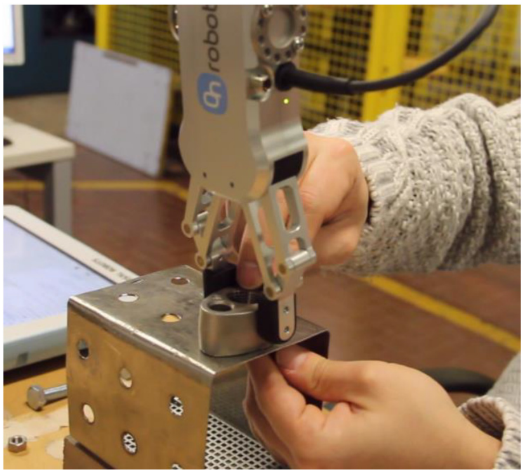


Figure 7. Collaboration during flange joining to the base

MDP states in present research correspond to the assembly tasks. The feasible assembly tasks can be automatically generated using the assembly sequence generation formalism [20].

The formalism is composed of a set of operations and of topological, functional and stability constraints. Constraints are represented as matrices with assembly pairs in the rows and contact and translation functions in the column. In [21] the method has been already applied to this case study. The generation of tasks and corresponding MDP states can be automated, removing from the list of tasks the unfeasible ones by applying in sequence the constraints. The list of Table 1 contains the resulting feasible assembly tasks.

Table 1. State list: feasible assembly tasks

State	Task	Slot n.	Action
1	B	-	Ass F, ass S
2	B U F1	1	Ass F, ass S, disass
3	B U F2	2	Ass F, ass S, disass
4	B U F3	3	Ass F, ass S, disass
5	B U S12	1,2	Ass F, ass S, disass
6	B U S23	2,3	Ass F, ass S, disass
7	B+F1 U S23	2,3	Ass F, ass S, disass
8	B+F3 U S12	1,2	Ass F, ass S, disass
9	B+F1+(S23 U F02)	2	Wrong state
10	B+F1+(S23 U F03)	3	Terminal
11	B+F3 + (S12 U F01)	1	Terminal
12	B+F3 + (S12 U F02)	2	Wrong state
13	B+ (S12 U F01)	1	Ass F, ass S, disass
14	B+ (S12 U F02)	2	Ass F, ass S, disass
15	B+ (S23 U F02)	2	Ass F, ass S, disass
16	B+ (S23 U F03)	3	Ass F, ass S, disass
17	B+ (S12 + F01) U F3	3	Terminal
18	B+ (S12 + F02) U F3	3	Wrong state
19	B+ (S23 + F02) U F1	1	Wrong state
20	B+ (S23 + F03) U F1	1	Terminal

The state is represented as a tuple made by the task, the slot position where the two parts are assembled and the actions that can be executed from this state. The possible actions are: assemble F, assemble S, disassemble. Some tasks are terminal, corresponding to the completion of the assembly, while some tasks are dead ends: feasible assembly but with a part in a wrong position, due to human mistake.

To understand the logic behind task generation it is important to remind that every value of a Markov state is determined only by the preceding state and action. Therefore, every Markov state must keep memory of the whole assembly sequence so far. As an example, states S11 and S17 are different, despite representing the same assembled configuration because their assembly sequence was different.

The structure of MDP is best understood by the graph of Figure 8. For sake of simplicity rewards and actions have been omitted. Actual rewards have been put equals to -1 for every couple (action, state), except for the terminal states that are bestowed a +2 reward.

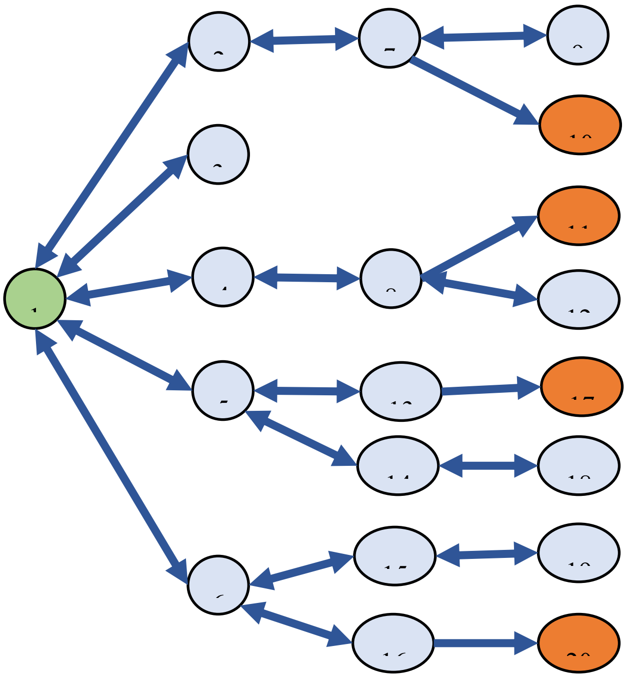


Figure 8. MDP with initial state in green and terminal states in red

Actions are either assemble oval or squared flange if the arrow points to the right. If the arrow points to the left, disassembly action is selected. Sometime the same action could lead to different solutions. The 'assemble flange' from S7 leads to S10 corresponding to the cor–rect mount in slot 3 and to S9 corresponding to a wrong position (slot 2). In this case a small statistical pro–bability has been introduced in the transition function to take into account the possibility of a human mistake. From S9 the only allowed action is disassembly.

The optimal sequence was found in [21] using a standard optimization method. It corresponds to the policy {S1, S2, S7, S10}.

Rewards should reflect to the task execution time with a minus sign. Longer the assembly time, less the value. By assigning the same reward -1, to all the tasks we made the system much more adaptable. Now there is a complete equivalence among the optimal sequence and other alternative sequences, namely {S1,S4,S8,S11} {S1,S5,S13,S17} {S1,S6,S16,S20}. These sequences are not completely equivalent as some of them require a slight amount of additional time to complete.

Anyway, this is knowledge from experience that purposely was not provided to the machine. The learning phase starts without any clue of which could be the best sequence.

6. RESULTS

The learning phase was conducted by privileging exploration for a long time in order to build alternative strategies for every possible solution. This result has been obtained by using the ε–greedy strategy with an initial value of ε equal to 0.9 and a decay factor equal to 0.005 after each episodes.

Therefore, the graph of reward (Figure 9) displays a purposely long number of episodes before convergence, even if the overall optimal policy could have been found way faster with a greedy strategy.

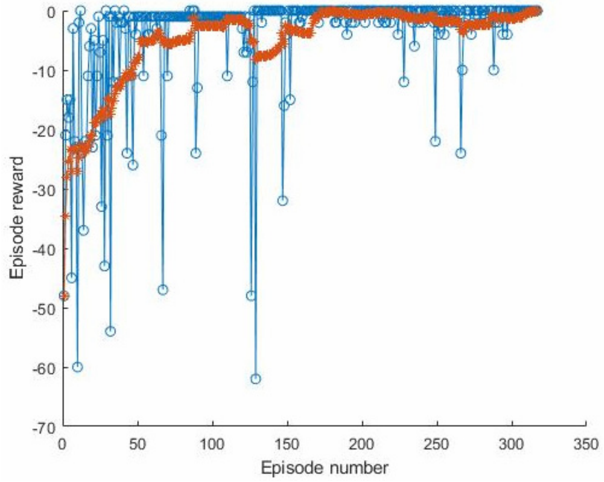


Figure 9. Rewards vs. training episodes. Expected return for every episode (light grey). 20 episodes-averaged reward (dark grey) is considered for terminating the training sequence

The optimal assembly sequence was found correctly as the states {S1, S2, S7, S10} with corresponding actions {assF, assS, assF}.

It is necessary to remark that in this simple case study, every optimization method would have found the same result with smaller computational effort. Thus, the objective of the work was to find all the alternative suboptimal sequences, to face unpredicted human interventions.

To test the robustness of the RL training, simulations were executed by having sometime the operator opting for states different from the ones belonging to the optimal sequence. This corresponds to an unexpected action by the human. If the chosen state belonged to an equivalent 'good' sequence (e.g. S5), the RL algorithm was able to accept the proposed alternative sequence. If the chosen state was a wrong one (e.g. S14 or S12), the algorithm proposes to disassemble the last part in such a way to come back to the 'good' path.

The robot initially trys to run the optimal sequence {S1, S2, S7, S10}. Due to the relatively small number of states, after every correct task, all the feasible alternative tasks were tested, to see the reaction of the RL trained agent.

Results are shown in Table 2. It is apparent that the agent makes always a 'minimum effort' move. If the human forced a state belonging to an acceptable task sequence, though not the optimal one, the RL agent switches on the new task sequence. It is equivalent to say that the robot tries to adapt its way of work to the human partner. On the contrary, if the human forces a state belonging to a wrong sequence, the robot disassemble last part (undo the task) and from this point proposes the nearest task sequence that allows to complete the assembly.

It is noteworthy that, with this approach, the robot never halts during the collaborative work, even if the human makes a 'small' mistake, as mounting the flange in a wrong slot.

Table 2. Assembly sequences described by the list of states, corresponding to assembly tasks. Every row in the table corresponds to the activation of a non-expected task (the modified states in underlined bold)

Assembly sequences proposed by RL agent						
Decision Steps	1	2	3	4	5	6
Optimal Sequence	S1	S2	S7	S10		
Human -> S2	S1	S2	S7	S10		
Human -> S3	S1	S3	S1	S2	S7	S10
Human –> S4	S1	S4	S1	S2	S7	S10
Human –> S5	S1	S5	S13	S17		
Human -> S6	S1	S6	S16	S20		
Human -> S9	S1	S2	S7	S9	S7	S10
Human -> S4&S8	S1	S4	S8	S4	S1,2,7,10	
Human->S5&S14	S1	S5	S14	S5	S13	S17

Unfortunately, there is also some inefficient choices by the RL agent. In some observations, the operator forces the execution of task4, the robot proposes the sequence {S1, S2, S7, S10}, but the human again forces S8, leading to both good and wrong terminal states. This state is equivalent to S7 but was not explored during training. Therefore, the agent is not able to propose the termination in S11 but insists in returning to the base sequence by disassembling all the work done.

This is due to uncomplete exploration of the solution state. In this simple case study, it would have been possible to further fine tune the exploration parameters in order to correct even this mistake. Although, in an industrial assembly, the number of parts to be assembled in a workcell could be far greater than this case and full exploration of the solution space would be computationally unfeasible.

This means that in an industrial assembly, if the human operator will deviate too much from the sequence proposed by the robot (completion-time optimal), it is possible that the robot won't be able to cope with the changes and will refuse to follow its human partner.

To be fair, even in manual production, if the operators make a too big mistake, or many small ones, during the job, the only viable solution is to stop everything and restart the job from the beginning.

7. CONCLUSIONS

The paper addresses the problem of improving HRC through the introduction of a further degree of flexibility in the robot directives. In the case of a collaborative assembly job executed by human and robot in a shared workspace, flexibility at operation level means trying to obtain robust movements, insensitive to disturbances and, at a task management level, trying to adapt the task sequence to the actual situation, at the price of giving up optimality.

While RL has been extensively exploited to improve robot flexibility at the trajectory level, it has not been considered until now as a way for optimizing the task sequence in presence of disturbances caused by unpredictable human behaviour. In this paper RL allows the collaborative robot to reach its goal by updating online the assembly tasks to meet the human counterpart choices. RL not only finds the best assembly sequence, but also explores the solutions space for viable alternatives. The robot becomes able to choose not only the best solution, but also an acceptable one by adapting to human moves.

The scientific contribution of present study is to highlight the importance of exploiting RL in robotic, not only for the optimization of continuous trajectory control, but also to support decision strategies in order to reduce the cognitive effort on the human worker.

Despite the overall good performances of the method, results have also shown that it is difficult to match the flexibility of human thinking and excessive changes of program could hinder the collaborative work.

Next research step will be using RL algorithms devised for the solution of adversarial games. The human operator, instead of a collaborator, will be considered as an opponent and the goal of the RL agent will be to complete the assembly job whichever interference the human will carry forward. This is a more correct strategy to maximize robustness of robot operations.

Another essential research step will be extending present method to a full complexity assembly derived by an industrial case study. MDP will consistently increase the number of states and the challenge will be to guarantee an adequate exploration of the solution state.

ACKNOWLEDGMENT

This work is supported by International Exchange Program for Graduate Students, Tongji University (No. 201902042).

REFERENCES

[1] Pereira, A. C., Dinis-Carvalho, J., Alves, A. C., & Arezes, P.: How Industry 4.0 can enhance lean practices. FME Transactions, 47(4), 810-822, 2019.

[2] Liu, Z., Liu, Q., Xu, W., et al.: Human-Robot Collaborative Manufacturing using Cooperative Game: Framework and Implementation, in: *51th CIRP Conference on manufacturing systems*, 72: 87-92, 2018.

[3] Hinds, P. J., Roberts, T. L., and Jones, H.: Whose job is it anyway? A study of human-robot interaction in a collaborative task. Human–Computer Interaction - J, 19(1-2), 151-181, 2004.

[4] Kober, J., Bagnell, J. A., and Peters, J.: Reinforcement learning in robotics: A survey. The International Journal of Robotics Research - J, 32(11), 1238-1274, 2013.

[5] Mateus, J., E.C., Aghezzaf, E.H., Claeys, D., Limère, V., & Cottyn, J.: Method for transition from manual assembly to human-robot collaborative assembly, in: *IFAC-PapersOnLine*, 51(11), 405-410, 2018.

[6] Nagai, Tatsuichiro, Shigeto Aramaki, and Isao Nagasawa. "Representation and programming for a robotic assembly task using an assembly structure." 7th IEEE international conference on computer and information technology, 2007.

[7] Ore, F., Hanson, L., Wiktorsson, M., and Eriksson, Y.: Automation constraints in human industrial robot collaborative workstation design, in: *Swedish Production Symposium SPS 2016*, 25 Oct 2016, Lund, Sweden, 2016.

[8] Stanton, N.A.: Hierarchical task analysis: Developments, applications, and extensions. Applied ergonomics - J, 37(1), 55-79, 2006.

[9] Tan, J. T. C., Duan, F., Zhang, Y., Watanabe, K., Kato, R., and Arai, T.: Human-robot collaboration in cellular manufacturing: Design and development, in: *IEEE/RSJ International Conference on Intelligent Robots and Systems,* pp. 29-34, IEEE, 2009.

[10] Bruno, G., Antonelli, D.: Dynamic task classification and assignment for the management of human-robot collaborative teams in workcells. The International Journal of Advanced Manufacturing Technology - J, 98(9-12), 2415-2427, 2018.

[11] Milner, J. M., Graves, S.C. and Whitney, D.E.: Using simulated annealing to select least-cost assembly sequences*,* in: *IEEE International Conference on Robotics & Automation,* IEEE, 2058-2063, 1994.

[12] Tseng, Y. J., Chen, J.Y. and Huang, F.Y.: A multi-plant assembly sequence planning model with integrated assembly sequence planning and plant assignment using GA, International Journal of Advanced Manufacturing Technology - J, 48(1-4), pp. 333-345, 2010.

[13] Lu, C., Yang, Z.: Integrated assembly sequence planning and assembly line balancing with ant colony optimization approach, International Journal of Advanced Manufacturing Technology - J, 83(1-4), 243-256, 2016.

[14] Antonelli, D., Astanin, S., Bruno, G.: Applicability of Human-Robot Collaboration to Small Batch Production, in: *Working Conference on Virtual Enterprises,* Springer, Cham, 24-32, 2016.

[15] Assembly Task Using an Assembly Structure, *7th IEEE International Conference on Computer and Information Technology (CIT 2007)*, Aizu-Wakamatsu, Fukushima, pp. 909-914, doi: 10.1109/CIT.2007.173, 2007.

[16] Stanton, N.A.: Hierarchical task analysis: Developments, applications, and extensions. Applied ergonomics - J, 37(1), 55-79, 2006.

[17] Akkaladevi, S., Plasch, M., Pichler, A., Rinner, B.: Human Robot Collaboration to Reach a Common Goal in an Assembly Process, in: *ECAI The Eight Starting Artificial Intelligence research symposium,* The Hague, Netherlands, 2016.

[18] Watkins, C., Dayan, P.: Q-learning. Machine Learning - J, 8(3-4):279-292, 1992.

[19] Akkaladevi, S. C., Plasch, M., Pichler, A., and Ikeda, M.: Towards Reinforcement based Learning of an Assembly Process for Human Robot Collaboration, in: *Procedia Manufacturing*, 38, 1491-1498, 2019.

[20] Gottipolu, R.B. and Ghosh, K.: A simplified and efficient representation for evaluation and selection of assembly sequences. Computers in Industry - J, 50(3), pp.251-264, 2003.

[21] Antonelli, D., & Bruno, G.: Dynamic distribution of assembly tasks in a collaborative workcell of humans and robots. FME Transactions, 47(4), 723-730, 2019

[22] Mandić, P., Lazarević, M.: An Application Exam–ple of Webots in Solving Control Tasks of Robotic System, FME Transactions, 41, 153-162, 2013.

РОБУСТНА ГЕНЕРАЦИЈА СЕКВЕНЦИ МОНТАЖЕ У ЧОВЕК-РОБОТ КОЛАБО–РАТИВНОЈ РАДНОЈ ЋЕЛИЈИ ПОМОЋУ УЧЕЊА ПОЈАЧАЊЕМ (*REINFORCEMENT LEARNING*)

Д. Антонели, К. Зенг, Х. Алиев, К. Лиу

Човек-робот колаборативне (ЧРК) производне ће–лије могле би повећати инклузивно запошљавање људске радне снаге без обзира на њихову снагу или вештине. Колаборативни роботи не само да замењују човека у опасним и тешким задацима, већ и чине све повезане процесе приступачним свим радницима, превазилазећи недостатак вештина и физичка ограничења.

Да би се омогућило потпуно искоришћавање колаборативних робота, потребно је превазићи традиционално програмирање робота. Смањење времена програмирања робота и когнитивног напора радника током посла постају „јаки“ захтеви које треба задовољити. Учење појачањем (РЛ) игра кључну улогу која омогућава роботу да се прилагоди променљивом и нест–руктурираном окружењу и људском независном извршавању понављајућих задатака. Овај рад се фокусира на употребу РЛ -а како би се омогућио робустан процес индустријске монтаже у ЧРК производној ћелији. Резултат студије је метода за *on-line* генерисање секвенце монтажних пеоцеса робота која се прилагођава непредвидивом и непостојаном понашању људских ко-радника. Метода је представљена уз помоћ референтне студије случаја.