Robust Adversarial Reinforcement Learning for Optimal Assembly Sequence Definition in a Cobot Workcell

Robust Adversarial Reinforcement Learning for Optimal Assembly Sequence Definition in a Cobot workcell

Alessandro ALESSIO[1], Khurshid ALIEV[1] and Dario ANTONELLI[1*]

[1] DIGEP, Politecnico di Torino, ITALY

* Corresponding author. Tel.: +39-011-090-7288; fax: +39-011-090-7299. E-mail address: dario.antonelli@polito.it

Abstract: The fourth industrial (I4.0) revolution encourages automatic online monitoring of all products to achieve zero-defect and high-quality production. In this scenario, collaborative robots, in which humans and robots share the same workspace, are a suitable solution that integrates the precision of a robot with the ability and flexibility of a human. To improve human-robot collaboration, human changeable choices or even non-significant mistakes should be allowed or corrected during work. This paper proposes a robust online optimization of the assembly sequence through Robust Adversarial Reinforcement Learning (RARL), where an artificial agent is deliberately trying to boycott the assembly completion. To demonstrate the applicability of robust human-robot collaborative assembly using adversarial RL, an environment composed of Markov Decision Process (MDP) like grid world is developed and a multi-agent RL approach is integrated. The results of the framework are promising: the robot observation on human activities has been successfully achieved thanks to a penalty-reward system adopted and the alternation of human to robot actions for the wrong terminal state is the one pursued by human, but due to robot blockage wrong actions, the right terminal state is followed by human, which is the same as the robot target.

Keywords: Smart Manufacturing, Machine Learning, Human Robot Collaboration, Industrial Assembly

1. Introduction

I4.0 is promoting companies trying to find new solutions to achieve high-quality products by integrating new enabling technologies into the sector. Integration of collaborative robots and machine learning (ML) into the assembly workspaces can improve and optimize assembly products and processes and reduce human errors during production. During assembly processes, human operators tend to do wrong

actions and human-robot reliability is important during collaboration with machines [1] [2]. However, machine learning tools can serve as a brain tool for the machines, especially for cobots to monitor human actions. Cobots are designed to interact with humans directly and physically inside a shared workspace [3]. According to some publications, human errors generate around 50% - 90% of quality problems in assembly manufacturing processes [4][5]. For this reason, the authors of [6] proposed an algorithm to assess individual memory structures and evaluation methods of human errors in different assembly tasks. Another research proposed by [7] is a method of analyzing human errors caused by quality defects on automobile engine assembly lines. A proposed method integrates cognitive reliability and error analysis methods and fault tree analyses. Machine learning method to detect human error and recovery in assembly is presented by the integration of supervision architecture at different levels of abstractions, functions, actions, and execution monitoring [8]. Above mentioned studies lack online autonomous monitoring of human actions during assembly processes. Thus, this paper presents a robust human-robot collaboration approach based on reinforcement learning (RL) that monitors human errors during collaborative job execution.

The paper is organized as follows: first human robot cooperative assembly formal definition through RL is presented Section 2, RL based assembly framework is described and explained in Section 3, the results of the RL based framework for cooperative assembly is discussed in Section 4 and conclusion is described in Section 5.

2. Human robot cooperative assembly definition through RL

Multi agent reinforcement learning (MARL) is an extension of single agent RL where multiple agents learn to maximize their individual cumulative rewards by collaborative interaction. Learning in single agent reinforcement learning is based on the Markov Decision Process (MDP), which is described by a 5-tuple $(S, A, P, r, \gamma, s_0)$ where, S is a finite set of states of the environment, composed of agent's all possible sensing information about the environment; A is a finite set of actions, including the agent's all possible actions; P is the state transition matrix $P_{SS'}^{a} = P[S_{t+1} = s' | S_t = s, A_t = a]$, R is the reward function $R_S^a = E[R_{t+1} | S_t = s, A_t = a]$; γ is the discount factor $\gamma \in [0,1]$ for the future rewards and s_0 is the initial state distribution.

Our proposed MARL system for optimal collaborative assembly can be expressed as a stochastic Markov game [9], where cobot (supervisor) agent engages to learn the optimal assembly sequence and we consider human (adversary) as a second agent who is involved to learn optimal path and have a tendency to do an error in the system. Thus MDP in this paper can be reformulated as a tuple: $(S, A_h, A_r, P, r, \gamma, s_0)$ where A_h actions of the human and A_r robot actions that can be performed. $P: S \times A_h \times A_r \times S \to R$ is the transition reward and $r: S \times A_h \times A_r \to R$ is the reward of both agents. If cobot is performing strategy μ and human

is performing wrong strategy v, the reward function is $r_{\mu,v} = E_{a^1\sim\mu(s),a^2\sim v(.|s)}[r(s,a^1,a^2)]$. In this case robot is maximizing the γ discounted reward while human is minimizing it.

A multi-agents cooperative assembly framework has been developed in the next section to demonstrate the robustness of the proposed human-robot cooperative assembly through MARL.

3. Reinforcement learning based Human robot cooperative assembly framework

The assembly task planning is a longtime field of study that has a first practical and successful solution in the AND/OR graph proposed by de Mello and Sanderson [10]. To produce the optimal assembly sequence, the authors of [11] proposed a simple simulated annealing method. To find the optimal assembly sequence, several capability factors were examined. The authors of [12] used a genetic algorithm (GA) to produce optimal assembly sequences by combining factory information with the evaluation of assembly sequence plans. The performance of the GA method was enhanced further by the authors [13]. The latest implementations of task planning algorithms, in human-robot, machine-to-machine collaborative/cooperative assembly applications can be found in the following researches [14][15]. Another key point of the human-robot cooperative/collaborative assembly applications is task assignment where authors [16] propose dynamic task classification and assignment approach for human and robot assembly in the collaborative work-cell.

The algorithms discussed above operate in a deterministic assembly work cell in which the robot or even a human follows the planned job sequence. A degree of uncertainty exists in manual assembly because an operator may follow the wrong or alternative job sequence, either because he knows it is equal to the one intended, or because of a minor fault, which frequently has minimal implications on the completion time.

To reduce human faults during cooperative assembly, this research proposes a framework that monitors multiple agents' actions to reach optimal paths using RL during cooperative assembly jobs.

Interactions between human, robot and environment take place as represented in Figure 1 in which four main parts are distinguished: the environment, the agents, the reinforcement learning algorithm and the trained neural networks.

The environment is constituted by the workspace where assembling operations are performed. For the reason that assembly sequences might vary depending on the job, performance, workload on the agents, and other factors, all potential combinations are examined, and each intermediate step in the process represents a different state. Only certain transitions are admissible from one state because they are sequenced according to assembly logic and the MDP structure provided in (2.3).

The agents also must develop the knowledge about the admissible actions for each state and then find the best assembling sequence to accomplish the task, for this reason they are trained following a specific reinforcement learning algorithm which is explained in detail in (2.4).

Agents trained independently can act according to different policies: in this case study, we used adversarial MARL directly compared to single agent approach. Since focus is on robot's behavior, this agent was trained to perform the best policy while human attempts to pursue a random objective, thus the robot must correct human's error during practice.

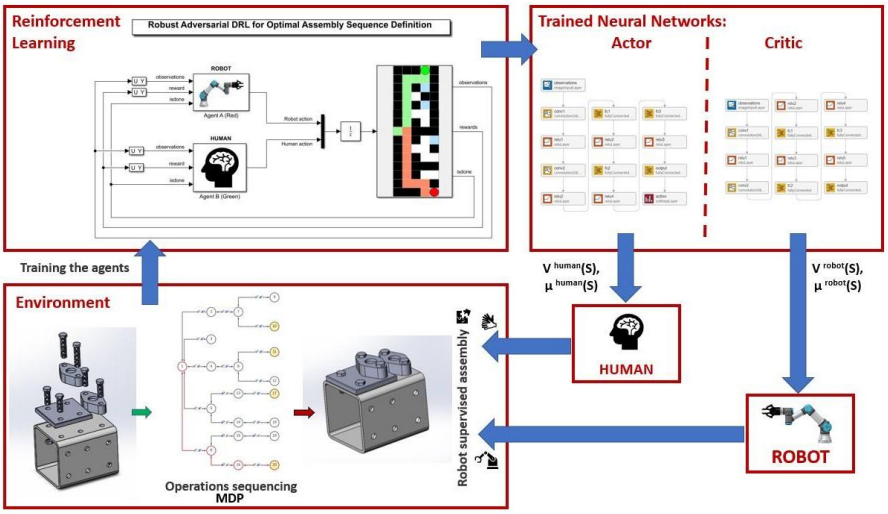

Figure 1. Reinforcement learning based framework for cooperative assembly

3.1 Adversarial RL for cooperative assembly

In the adversarial environment assembly process, at every timestamp t both agents (robot and human) observe the state s_t and take actions $a_t^r \sim \mu(s_t)$ and $a_t^h \sim v(s_t)$. The state transitions $s_{t+1} = P(s_t, a_t^r, a_t^h)$ and a reward $r_t = r(s_t, a_t^r, a_t^h)$ is obtained from the environment. In the human robot assembly process robot gets a reward $r_t^r = r_t$ while a human is adversary receives a reward $r_t^h = r_t$. Thus, each step of the assembly MDP can be represented as $(s_t, a_t^r, a_t^h, r_t^r, r_t^h, s_{t+1})$. In the assembly robot protagonist is attempting to optimize the following reward function,

$$R^1 = E_{s_0 \sim p, a^1 \sim \mu(s), a^2 \sim v(s)} \left[\sum_{t=0}^{T-1} r^1(s, a^1, a^2) \right]$$

because policies μ and v are learnable elements, $R^1 \equiv R^1(\mu, v)$. Likewise, the human seeks to do 'wrong' actions and maximize its own negative reward: $R^2 \equiv R^2(\mu, v) = -R^1(\mu, v)$. In our example, the assembly path is optimized first using a robot agent, and then with the involvement of a second human agent. In this case, the human operator's objective is to pursue a terminal condition which is not necessary the same of the robot, for this reason if it would be any mismatch between the agents' assembly sequences, the robot would correct the human driving him performing the right action.

3.2 Environment (MDP and GridWorld)

The physical environment is the workspace where the assembly is executed but to train the agents also a virtual one is needed. As the same as the real world with constrains and feasible actions, the agents can perform only one operation at time and can only move to certain states: they are allowed to advance to the next state, to regress to a previous step or to wait without doing nothing. This kind of behavior has been simulated by means of a grid world which has the same structure of the MDP schema represented in Figure 2. In the MDP chart arrows indicate the admissible transitions, blue labels indicate whether that action should be done by the human (a^h) or the robot (a^r), yellow highlighted states are terminals and red-colored path is the one that must be learnt by agents thanks to RL algorithm.

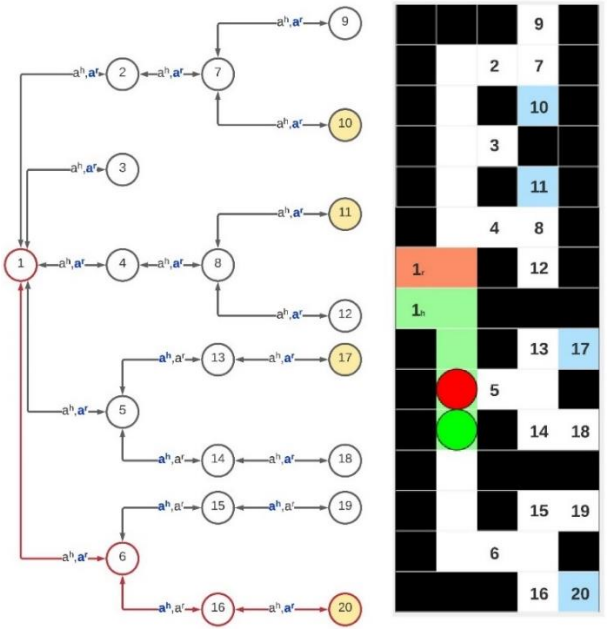

Figure 2. Human robot collaborative assembly MDP structure and grid world frame during training

Matlab software was used to create the grid world, which shows the sequence of potential assembly steps. Black cells indicate constraints that force agents to follow only approved trajectories by limiting their movement. The same designation was used to indicate states in MDP and grid world, and the terminals' background is distinguished with a light blue tone. Agents in the grid are represented by circles, with red indicating the robot and green indicating the human; each path covered is marked with the color of the corresponding agent.

The training was performed as a multiple simulation sequence during which the PPO agent's neural networks of actor and critic were updated in weights and biases.

An observation is taken at each time step by taking a photograph of the current simulation. Four channels are supplied as input to the agents for a single observation: The first channel is for obstacles, which defines the grid world's structure; the second is the "self-channel", which defines the agent's path; the third is the "other-agent-channel", which describes the path covered by the other agent; and the fourth is for terminal states. Agents explored the grid world looking for terminal states according to environment's constraints. Rewards were assigned differently basing on the cells reached and regarding to the agents, Table 1 shows the rewards in detail for each agent.

Table 1 Rewards and penalties for each agent

Action	**Robot Reward**	**Human Reward**
Illegal action (obstacles, out of grid world)	-10	-10
Idle	-10	-10
Move to already explored cell	+0.5	-0.5
Admissible action	-1	-1
Collision with another agent	+1	-1
Terminal state 10 (row 3, col 4)	+1	+5
Terminal state 11 (row 5, col 4)	+1	+1
Terminal state 17 (row 9, col 5)	+1	+1
Terminal state 20 (row 15, col 5)	+5	+1

The penalty to each admissible action was assigned to get the target faster, to make the robot forces the human to follow the desired path each collision between them was taken into account to return respectively a penalty for the human and a reward to the robot, rewards for terminal states were assigned differently to the agents to drive them through different paths and perform the adversarial RL. Since in this case study the adversarial MARL approach was used with focus on robot's behavior, to its final reward value a 10% of human's reward value was subtracted. The reason of doing this for the robot but not for human lies in the fact that assembling task is not the same of standing up to the adversary in a game for which a classical adversarial approach is needed for both agents: in assembly tasks human and robot have to reach a common goal but in real case it can happen that human makes mistakes respect the predetermined sequence; in those cases, for a robust

design, the robot should adjust the next actions according to the best strategy. Adversarial behavior is realized thanks to the human that "unconsciously" (with his actions which differs from the one of the robots) reduces the robot's reward. This behavior pushes the robot to hamper human's actions when they are wrong. If both agents were totally adversarial the assembly wouldn't be possible.

In single agent approach rewards assignment doesn't change except for the penalty inflicted to robot in relation to human's reward which is obviously absent.

3.3. Actor-critic agents' network.

Proximal Policy optimization (PPO) agent is an online, model free, policy gradient reinforcement learning method [10][11]. This algorithm alternates sampling data from interaction with the environment and optimizing surrogate objective function: PPO agent estimates the probability to take each action in a specific state and acts with respect to probability distribution; the current policy is implemented for a determined number of epochs and then both actor and critic are updated using a minibatch. Using PPO agents either the observations or the actions can be both discrete and continuous.

Policy and value function are estimated thanks to two function approximators: actor $\mu(S)$ and critic $V(S)$. The actor takes the observations S and returns the probabilities of taking each action in that state. The critic, from observations S returns expectation of discounted long-term reward. At the end of the training the optimal policy is stored in the actor.

Policy gradient methods estimate the weights of the policy using the gradient ascent algorithm. According with Schulman et al. (2017) the loss policy for PPO agent is:

$$L_t^{CLIP+VF+S}(\theta) = \hat{E}_t[L_t^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S[\pi_\theta](s_t)\,]$$

where c_1, c_2 are coefficients, $L_t^{CLIP}(\theta)$ is the clipped policy gradient objective ("surrogate"), S is the entropy bonus to promote the exploration of the agent and $L_t^{VF}(\theta)$ is the squared-error loss $\left(V_\theta(s_t) - V_t^{targ}\right)^2$.

The training algorithm, after initialization of both actor $\mu(S)$ and critic $V(S)$ with random parameter values θ_μ, θ_V respectively, is in this way executed:

1. N experiences are generated by following the current policy: $S_{ts}, A_{ts}, R_{ts}, S_{ts+1}, A_{ts+1}, R_{ts+1}, \dots S_{ts+N}, A_{ts+N}, R_{ts+N}$ where S are the states, A are the actions and R the Rewards; N corresponds to a terminal state or at maximum to the Experience Horizon value.
2. For each episode step compute the return and advantage function.
3. Learn from mini-batches of experience over K epochs:
    a. Sample random mini-batch data set from the current set of experience;
    b. Update the critic parameters by minimizing the loss L_{critic} across all sampled mini-batch data;

c. Update the actor parameters by minimizing the loss L_{actor} across all sampled mini-batch data and additional entropy loss is added to this term, which encourages policy exploration.

Steps are repeated until the training episode reaches a terminal state.

4. Results and discussions

In this example the grid world environment was used to visualize the training results. The grid world was structured to look like the MDP graph for a better interpretation. Since they have the same structure, one could expect to have a terminal condition for each training episode corresponding to each terminal state but in this way, in multi-agent scenario, no convergence was achieved. For this reason, during multi-agent training only the desired end condition was set as terminal among all the possibilities.

The robot surveillance on human activities has been successfully achieved thanks to the penalty-reward system adopted and a step-by-step alternation of human to robot actions. Terminal state 10 is the one pursued by the human during the very first training episodes; on the contrary, robot's target is terminal state 20. Opposite direction between these terminal states have been exploited to cause agents collision and allow them to understand how to react on these occurrences to maximize their own rewards.

Assuming that the robot's aim is right because it leads to a correctly assembled item, and since the robot's actions are free of decision-making autonomy compared to humans', the robot blocks human's incorrect movements. Because the single episode does not terminate if agents are not in the planned final position or the single episode reaches the maximum number of steps, even if the human reaches a wrong terminal state without being blocked by the robot, the human's return value is heavily affected due to the numerous collisions between agents.

Figure 3 depicts the agents' learning progress: red marks and lines denote robot behavior, whereas green marks and lines denote human behavior. The early episodes are required for each agent to explore, as seen in the graph; nevertheless, rewards are low in value since they do numerous illegal actions that result in high penalties. Agents begin to understand the path after a first phase of random actions, and a second phase of training is visible in the plot by the first "horizontal" trend: agents begin to recognize the goodness of terminal states that are regularly achieved, but human and robot objectives are still different. Collisions appear to be considered in the latter stages of training, just before the convergence asymptote: there are remarkable spikes in rewards values that alternate between the agents. Given all the above-mentioned rewards and penalties, maximizing the reward value for each agent causes the human to change its objective and not deviate from the robot's optimal path in order to avoid more penalties. Training came to an end when the agents reached the terminal designated condition in the shortest time possible.

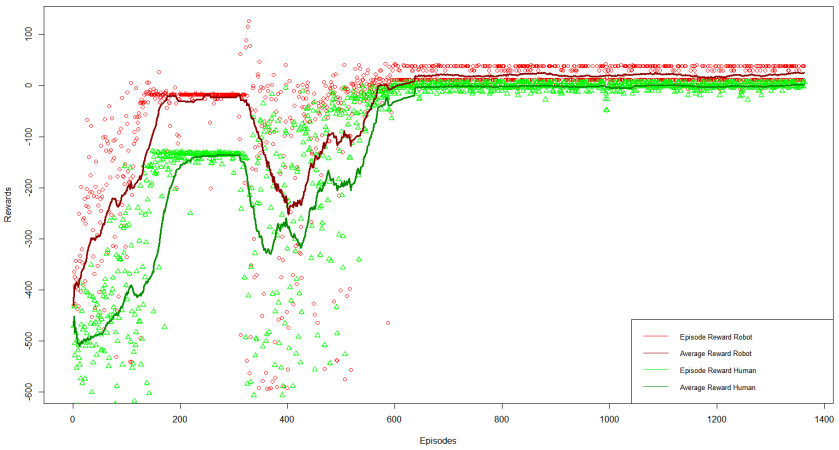

Figure 3. Training process diagram: dots represent single episode reward for each agent, lines represent corresponding average reward computed on a set of 50 episodes.

5. Conclusions

Flexibility and adaptability in tasks execution are unrivalled characteristic proper of human nature, anyway this property can bring the human to execute actions in different way respect the scheduled order. In collaborative work-cells is then necessary to perform robust programming to allow the robot to manage every situation to avoid production slowdowns and stops. In this case study was demonstrated that robot can successfully force human to follow the assembling sequence thanks to the proper implementation of MARL.
The use of a grid world environment to simulate agents' state transitions is the approach's limit; in fact, respect the real scenario in which one agent's activity changes the state of both, in this example agents have their own states, so they can't be in the same state at the same time.
This work may be improved further modifying the training environment and applying the created algorithm to a real-world assembly process, as well as testing it with human errors.

6. References

1. Park, K. S. (2014). Human reliability: Analysis, prediction, and prevention of human errors. Elsevier.

2. Aliev, Khurshid, and Dario Antonelli. (2021). Proposal of a Monitoring System for Collaborative Robots to Predict Outages and to Assess Reliability Factors Exploiting Machine Learning. Applied Sciences 11.4, 1621.
3. Galin, Rinat, and Roman Meshcheryakov. (2019). Review on human–robot interaction during collaboration in a shared workspace. International Conference on Interactive Collaborative Robotics. Springer, Cham.
4. Liu, G. Z., Zhang, Y. J., Li, Z., Ying, Y., & Cai, Z. X. (2007). Human errors analysis in substation operation based on CREAM. Electric Power, 40(5), 85-89.
5. Di Pasquale, V., Miranda, S., Neumann, W. P., & Setayesh, A. (2018). Human reliability in manual assembly systems: a Systematic Literature Review. Ifac-Papersonline, 51(11), 675-680.
6. Strenge, B., & Schack, T. (2021). Empirical relationships between algorithmic SDA-M-based memory assessments and human errors in manual assembly tasks. Scientific Reports, 11(1), 1-12.
7. Le, Y., Qiang, S., & Liangfa, S. (2012, October). A novel method of analyzing quality defects due to human errors in engine assembly line. In 2012 International Conference on Information Management, Innovation Management and Industrial Engineering (Vol. 3, pp. 154-157). IEEE.
8. Lopes, L. S., & Camarinha-Matos, L. M. (1995). A machine learning approach to error detection and recovery in assembly. In Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots (Vol. 3, pp. 197-203). IEEE.
9. Perolat, J., Scherrer, B., Piot, B., & Pietquin, O. (2015). Approximate dynamic programming for two-player zero-sum markov games. In International Conference on Machine Learning (pp. 1321-1329). PMLR.
10. De Mello, L. H., & Sanderson, A. C. (1990). AND/OR graph representation of assembly plans. IEEE Transactions on robotics and automation, 6(2), 188-199.
11. Milner, J. M., Graves, S.C. and Whitney, D.E. (1994). Using simulated annealing to select least-cost assembly sequences, in: IEEE International Conference on Robotics & Automation, IEEE, 2058-2063.
12. Tseng, Y. J., Chen, J.Y. and Huang, F.Y. (2010). 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.
13. Lu, C., Yang, Z. (2016). 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.
14. Aliev, K., Antonelli, D., & Bruno, G. (2019). Task-based programming and sequence planning for human-robot collaborative assembly. IFAC-PapersOnLine, 52(13), 1638-1643.
15. Aliev, K., & Antonelli, D. (2019, May). Analysis of cooperative industrial task execution by mobile and manipulator robots. In International Scientific-Technical Conference MANUFACTURING (pp. 248-260). Springer, Cham.
16. Bruno, G., & Antonelli, D. (2018). Dynamic task classification and assignment for the management of human-robot collaborative teams in workcells. The International Journal of Advanced Manufacturing Technology, 98(9), 2415-2427.
17. Schulman, John, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. “Proximal Policy Optimization Algorithms.” ArXiv:1707.06347 [Cs], July 19, 2017.
18. John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, Pieter Abbeel. “Trust Region Policy Optimization.” ArXiv:1502.05477 [cs.LG], February 19, 2015