POLITECNICO DI TORINO
Repository ISTITUZIONALE

Enhancing Trust in Collaborative Assembly Through Resilient Adversarial Reinforcement Learning

Original
Enhancing Trust in Collaborative Assembly Through Resilient Adversarial Reinforcement Learning / Antonelli, D., Aliev, K., Yang, B.. - In: APPLIED SCIENCES. - ISSN 2076-3417. - 16:7(2026). [10.3390/app16073244]

Availability:
This version is available at: 11583/3017033 since: 2026-10-26T15:45:30Z

Publisher:
Multidisciplinary Digital Publishing Institute (MDPI)

Published
DOI:10.3390/app16073244

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

19 November 2026

applied sciences

MDPI

Article

Enhancing Trust in Collaborative Assembly Through Resilient Adversarial Reinforcement Learning

Dario Antonelli [1,*], Khurshid Aliev [1] and Bo Yang [2]

[1] Department of Management and Production Engineering, Politecnico di Torino, 10129 Torino, Italy; khurshid.aliev@polito.it
[2] State Key Laboratory of Mechanical Transmission, Chongqing University, Chongqing 400044, China; yangbo61@cqu.edu.cn
* Correspondence: dario.antonelli@polito.it; Tel.: +39-011-090-7288

Featured Application

This work employs Adversarial Reinforcement Learning in the context of industrial collaborative assembly, thereby facilitating the robust adaptation of robots to human errors and unpredictable behaviors, ensuring the reliable completion of assembly tasks.

Abstract

Collaborative robots, or cobots, are designed to improve productivity and safety in industrial settings. However, effective Human–Robot Collaboration (HRC) relies heavily on the human operator's trust in the robotic partner. This study posits that trust is significantly enhanced by the robot's ability to adapt to unpredictable human behavior. To achieve this adaptability, we propose applying an Adversarial Reinforcement Learning (ARL) framework to the robot's activity planning. We model the assembly process as a Markov Decision Process (MDP) on a Directed Acyclic Graph (DAG). The robot learns an assembly policy using an on-policy algorithm while a simulated human agent, trained with the same algorithm, acts as an adversary that introduces disturbances and delays. We applied the proposed approach to a simple industrial case study and evaluated it on complex assembly sequences generated synthetically. Although the ARL-trained robot did not outperform conventional assembly optimization algorithms in terms of task completion time, it guaranteed robustness against human variability. This ensured task completion within a bounded timeframe regardless of human actions. By demonstrating consistent performance and adaptability in the face of uncertainty, the robot exhibits the Ability and Benevolence components of the ABI model of trust. This fosters a more resilient and trustworthy collaborative environment.

Keywords: human–robot collaboration; trust and safety in robotics; human-centered automation; Adversarial Reinforcement Learning; Automated Assembly Planning

Check for updates

Academic Editor: Rui Araújo

Received: 3 March 2026
Revised: 23 March 2026
Accepted: 25 March 2026
Published: 27 March 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

The manufacturing sector is currently undergoing a unique transformation characterized by the convergence of high-performance computing, advanced mechatronics, and autonomous decision-making systems. Central to this shift is Automated Assembly Planning (AAP), a critical research area that addresses the challenge of converting static digital product designs into executable manufacturing processes [1]. In the context of intensifying

labor shortages, escalating operational costs, and the demand for highly customized products, there is a need for highly flexible and intelligent assembly systems that can address these issues [2]. One way to achieve flexibility in automation is to deploy collaborative robots, also known as cobots, in the industrial workforce. Despite their potential, small and medium-sized enterprises (SMEs) often hesitate to fully adopt them [3,4]. Common barriers include high implementation costs, safety concerns, and the complexity of programming robots for collaborative environments [5].

To overcome these barriers, it is crucial to consider not only the technical specifications but also the dynamics of the team itself. Effective human collaboration relies on shared goals, open communication, defined roles, and, most importantly, trust [6,7]. Although task assignment and scheduling have been well researched, the psychological aspects of human–robot collaboration (HRC), specifically trust, have been less explored in the context of robot control strategies.

In organizational and collaborative settings, trust is commonly conceptualized using the ABI model proposed by Mayer et al. [8], which identifies three key antecedents of trustworthiness: ability, benevolence, and integrity.

- Ability refers to the skills and competencies that enable a party to function reliably within a specific domain. In HRC, this translates to the robot's capability to complete tasks correctly and safely.
- Benevolence is the extent to which a trustee is believed to want to do good for the trustor. For a robot, this can be interpreted as the capacity to adapt its actions to support human partners, even when they make errors or deviate from the plan.
- Integrity involves adhering to a set of principles acceptable to the trustor, which implies predictability and consistency in behavior.

These components form the basis for objectively analyzing trust in human–robot collaboration (HRC) situations [9]. In typical industrial scenarios, robots are programmed for rigidity and repetition, which conflicts with the flexibility required of human teams. Flexibility—the ability to adapt to changing circumstances—and resilience in the face of stress are characteristics of effective teamwork [10,11]. If a robot cannot cope with human variability (e.g., mistakes, fatigue, or creative problem-solving), then it fails to demonstrate ability and benevolence, thereby eroding trust.

This paper proposes that trust in HRC situations is directly impacted by a robot's ability to adapt to human behavior. The following research question (RQ) is addressed:

RQ1: *Are there solutions to the AAP problem that could ensure reliable task completion despite human unpredictability, thereby enhancing the trustworthiness of the system?*

Adversarial Reinforcement Learning (ARL) is used to train task planning in cobots. In this framework, the robot acts as the protagonist, aiming to complete the assembly, while the human is modeled as the "adversary" who introduces delays. This adversarial training compels the robot to develop conservative, resilient strategies that offset the effects of human variability, albeit with a slight increase in process times. By guaranteeing task completion within a reasonable time, regardless of human actions, the robot demonstrates high ability and functional benevolence, laying the foundation for a high-trust collaborative relationship.

The original contribution of this study lies in its alternative approach to the AAP problem. Instead of focusing solely on achieving the shortest completion time, the goal is to establish a high level of trust between the human operator and the robot. This objective is achieved by adopting an ARL framework during robotic planning training.

2. State of the Art in AAP

In a nutshell, the objective of AAP is to translate CAD product designs into executable robotic code. The computational core of AAP branches into two primary, interdependent subproblems: Assembly Sequence Planning (ASP) and Assembly Path Planning (APP). Both problems are computationally intensive [12]. APP generates a collision-free trajectory for each component from its initial kitting location to its final pose in the assembly, which is outside the scope of this paper. However, it is worth noting that modern APP research increasingly incorporates multi-criteria optimization. For instance, analyzing and minimizing robot energy consumption during trajectory planning is a growing trend in sustainable manufacturing (e.g., [13,14]).

ASP involves determining the optimal order in which components should be combined. Classic solution methods utilize interference matrices, precedence graphs, and sequence–relation matrices to define the geometric and technological constraints of a product [12]. The graphs are optimized using heuristic and meta-heuristic search algorithms. These include Genetic Algorithms [15], Particle Swarm Optimization [16], Ant Colony Optimization [17], and Cuckoo Search [18]. Recent advances have introduced the Q-Learning-based Genetic Algorithm, which enhances traditional Genetic Algorithms by incorporating Reinforcement Learning (RL) [19]. This hybrid approach enables the algorithm to escape local optima and handle complex parallel constraints with greater efficacy than standard optimization tools.

The infusion of Artificial Intelligence has transformed AAP from a purely geometric problem into a cognitive one. Neural networks are trained to find the optimal assembly algorithm. One approach is the k-means clustering algorithm [20]. Alternatively, RL enables agents to learn optimal strategies by interacting with a simulated environment and maximizing a cumulative reward [19]. In the context of ASP, the objective of the agent is to assemble or disassemble a product in the shortest amount of time while respecting all assembly constraints [21]. Recent research focuses on integrating RL with classical planning to account for the inner hierarchical nature of assembly task [22].

With Industry 5.0 and the growing prominence of SMEs, assembly planning has evolved from a narrow focus on time optimization to a broader commitment to improving the overall quality of the work environment [23]. Similarly, this study moves beyond optimizing standard planning metrics. Instead, it incorporates human variability into the decision-making framework. This enables the robot to demonstrate the adaptability and reliability necessary to foster trust. The desired result is that, if supported by the human partner, the robot follows an optimal assembly sequence; otherwise, it searches for the best obtainable sequence based on the human's choices. To achieve this, the RL solution to the AAP problem is extended to introduce the human factor as a contrasting agent in an ARL framework. While ARL has been successfully applied to continuous, low-level kinematic trajectory control to handle physical disturbances or sensor noise, its application to the high-level, discrete logic of Assembly Sequence Planning (ASP) remains largely unexplored. This study bridges that gap by applying adversarial training to the topological decision-making level, ensuring logical task completion rather than just physical collision avoidance.

3. Materials and Methods

3.1. Task Decomposition for Collaborative Assembly

To address the AAP problem, we adopt a hierarchical model of assembly [24]. The assembly job is decomposed into tasks and operations as described by Figure 1.

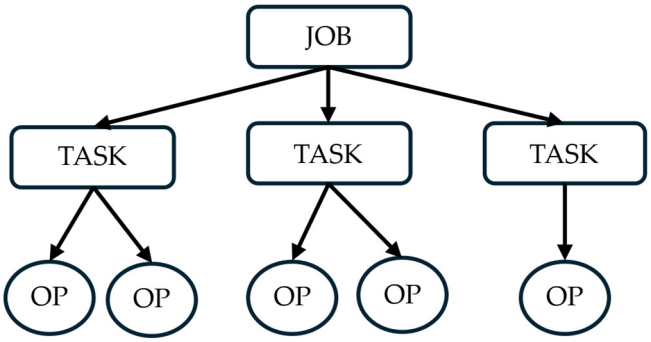


Figure 1. Hierarchical model of assembly job decomposed in tasks and related operations.

The assembly sequence can be formalized by defining a set of operations and applying specific constraints to generate an optimal Assembly Sequence Table (AST). According to the study of Gottipolu et al. [25], the process can be structured into task definitions, relational functions, and constraint integration.

Task Representation: Collaborative assembly work is represented as a set of tasks involving N assembly components. A task is defined by the tuple $\mathrm{AT_i}$:

$$AT_i = \langle P, R, O \rangle, \tag{1}$$

where $\mathrm{P} = \{\mathrm{P_1}, \mathrm{P_2}, \ldots, \mathrm{P_N}\}$ is the set of parts, $R = \{R_1, R_2, \ldots, R_M\}$ represents the relations between parts, and $O = \{O_1, O_2, \ldots, O_k\}$ is the ordered set of operations performed by the robot and/or human.

Operations are further defined as $\mathrm{O_{ij}} = \langle \mathrm{R_{ij}}, \mathrm{OC_{ij}} \rangle$ where R_{ij} is the connection between components and OC_{ij} represents optimization constraints for that operation. The relations between parts can be expressed as:

$$R_{ij} = \langle P_i, P_j, GC_{ij}, GT_{ij} \rangle. \tag{2}$$

The terms GC_{ij} determine if the touch contact function or the feasibility constraint are applied and GT_{ij} are general translational statements [26].

Constraints are integrated using Boolean operators on the relational functions to validate the AST. They distinguish in absolute constraints and optimization constraints.

1. Absolute Constraints (feasibility, precedence):
   a. Feasibility Constraints (RTC): validate contact existence. The "OR" operator is applied to the feasibility constraint TC.

$$\mathrm{RTC} = \mathrm{TC_1} \vee TC_2 \vee \mathrm{TC_3} \vee \mathrm{TC_4} \vee \mathrm{TC_5} \vee \mathrm{TC_6} \tag{3}$$

   b. Precedence Constraints (RTT): checks for collision-free paths. The "AND" operator is applied to columns of the precedence constraint TT truth table to find the Boolean product $\mathrm{TT_i}$, before to sum them:

$$TT_i = T_i^{B,F1} \wedge T_i^{S,F1} \tag{4}$$

$$RTT = TT_1 \vee TT_2 \vee TT_3 \vee TT_4 \vee TT_5 \vee TT_6 \tag{5}$$

2. Optimization Constraints (topological, functional, and stability):
   a. Topological Constraint: Ensures the application of precedence rules.
   b. Functional Constraint: Ensures the task is feasible for the robot gripper.
   c. Stability Constraint: Ensures parts remain stable during the assembly.

This representation of assembly tasks can be converted into an MDP graph by making assembly-specific assumptions [27]. The conversion in MDP is presented in Section 4 for the chosen case study.

3.2. Synthetic Generation of Assembly Sequences

To generalize the problem, we can assume that the assembly sequence is represented by a graph displaying the assembly tasks defined in (1). The level of detail of the graph, i.e., the precise definition of an elementary operation, is left to the applicants' discretion. The assembly process is expressed as a Directed Acyclic Graph (DAG), where the nodes represent assembly operations (states), and the edges correspond to feasibility or precedence constraints between components. Feasibility constraints are applied by removing edges corresponding to unfeasible sequences.

A DAG is a pair $G = (V, E)$, where V is a finite, non-empty set of elements called vertices (or nodes), $E \subseteq \{(u, v) \in V \times V \mid u \neq v\}$ is a set of ordered pairs of distinct vertices called directed edges (or arcs) [28]. An edge $(\mathrm{u}, \mathrm{v}) \in \mathrm{E}$ represents a directed connection from u to v. In a DAG, for any vertex v, there is not a non-empty directed path starting at v and ending at v.

In the specific application to assembly sequences, the DAG implies a partial ordering of tasks. There exists a linear ordering of vertices such that for every directed edge $(\mathrm{u}, \mathrm{v}) \in \mathrm{E}$, vertex u comes before v in the ordering. This represents a valid sequence of assembly operations ensuring that the process contains no cycles, preventing loops or deadlocks during execution.

Note that the choice of DAG is not imposed by the problem, but rather, it is a deliberate design choice aimed at applying ARL to the system represented by the graph. In fact, the DAG does not directly correspond to the assembly graph generated by incorporating all feasible assembly sequences. A DAG must respect additional ordering constraints that can only be obtained by duplicating every node that could be accessed in multiple orders. An automatic procedure can convert a standard assembly graph into a DAG assembly sequence. Figure 2 illustrates this process. If both sequences $A \rightarrow B \rightarrow C$ and $A \rightarrow C \rightarrow B$ are admissible, the non-ordered assembly task representation could be the one of Figure 2a. Both directions are admissible for moving through tasks B and C. However, if we force a sequence, either task B or task C must be executed in one step, and the other must be executed in the next time step. Therefore, the DAG representation becomes that of Figure 2b.

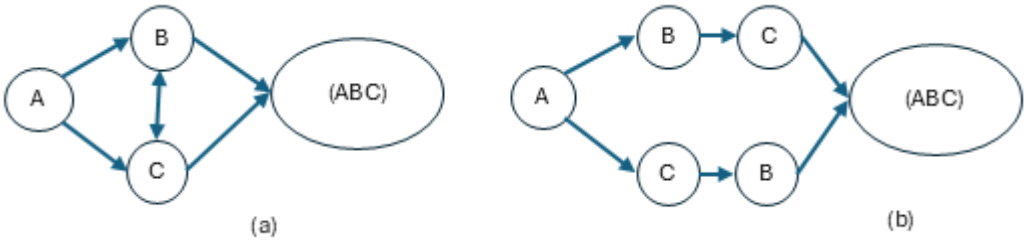


Figure 2. Assembly graph with (**a**) ordinary representation and (**b**) DAG representation.

To evaluate assembly processes of arbitrary complexity, synthetic data generation was employed. A straightforward procedure was formulated to generate random DAGs for the purpose of evaluating the proposed ARL algorithm across a wide spectrum of assembly sequences. The random edge selection process incorporates a specific heuristic: early nodes are assigned a higher likelihood of performing more actions, reflecting the greater number of alternatives typically available at the start of an assembly. Edges are drawn preferentially

towards adjacent layers to reduce the likelihood of creating funnel-like structures. Figure 3 provides an example of a randomly generated DAG.

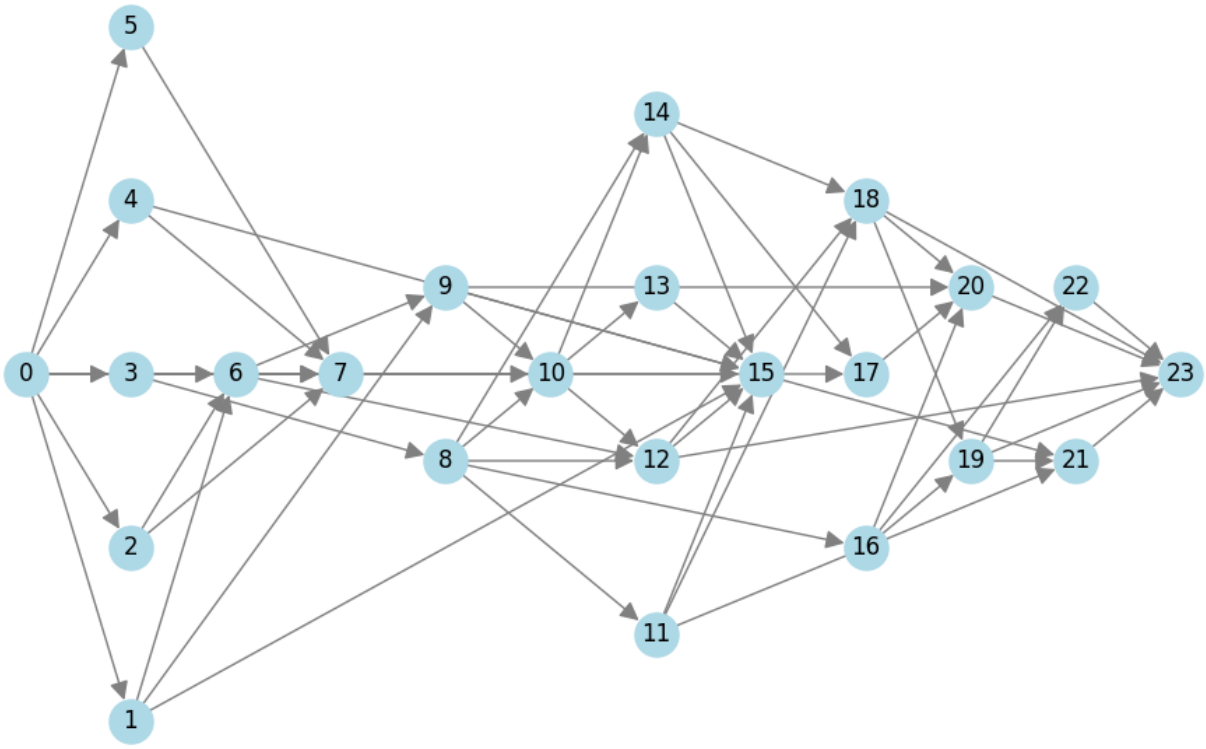


Figure 3. Random generated DAG for testing purposes with ten inner layers, ten nodes maximum per layer, and four actions maximum per node.

3.3. Reinforcement Learning Models

To give the robot with the decision-making autonomy necessary for flexible collaboration, RL is employed. RL is a machine learning paradigm in which an agent learns to make decisions by performing actions in an environment and receiving feedback in the form of numerical rewards or penalties [29]. The objective of the agent is to discover a policy π, a mapping from perceived states to actions, that maximize the cumulative reward over time.

The collaborative assembly process is modeled as a Markov Decision Process (MDP). An MDP is formally defined as a tuple $(\mathrm{S}, \mathrm{A}, \mathrm{P}, \mathrm{R}, \gamma)$, where [30]:

- S is the set of all possible states in the environment (e.g., the status of the assembly);
- A is the set of valid actions the agent can take (e.g., picking a part, fastening a bolt);
- P represents the state transition probability, describing the likelihood of moving to a new state S' given the current state S and action A;
- R is the reward function, providing a scalar feedback signal $\mathrm{R}(\mathrm{S}, \mathrm{A})$ received after transitioning from state S via action A;
- $\gamma \in [0, 1]$ is the discount factor, which determines the importance of future rewards compared to immediate ones.

The core of the RL algorithm involves estimating the value function, specifically the Q-value, which is denoted as $\mathrm{Q}(\mathrm{S}, \mathrm{A})$. The Q-value represents the expected cumulative reward that an agent can achieve by starting from state S, taking action A, and following a specific policy thereafter. These values are updated iteratively until they converge to the optimal policy.

On-policy algorithms, such as SARSA and PPO, learn the value of the current policy, meaning they improve the specific strategy the agent is currently using, whereas off-policy algorithms (like Q-learning) learn the value of the optimal policy independently of the

agent's current actions, allowing them to learn from data generated by other strategies. In this work, we use Proximal Policy Optimization (PPO), a state-of-the-art on-policy gradient algorithm [31]. Unlike value-based methods which estimate the value function to derive a policy, policy gradient methods optimize the policy $\pi_\theta(a|s)$ directly. PPO is designed to ensure stable and reliable updates by preventing the policy from changing too drastically in a single step. The core of PPO is the clipped surrogate objective function:

$$L^{CLIP}(\theta) = \hat{E}_t\left[min\left(r_t(\theta)\,\hat{A}_t,\ clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\,\hat{A}_t\right)\right], \tag{6}$$

where $r_t(\theta)$ is the probability ratio between the new and old policies, $\hat{A}_t$ is the estimated advantage, and ϵ is a hyperparameter defining the clipping range. The clipping mechanism ensures that the update remains within a "trust region," thereby improving training stability.

The estimated advantage is mathematically defined as the difference between the specific action-value and the state-value:

$$\hat{A}_t = Q(s_t, a_t) - V(s_t), \tag{7}$$

where $Q(s_t, a_t)$ (Action Value) is the expected cumulative reward for taking action a_t in state s_t and $V(s_t)$ (Baseline) is the average expected cumulative reward for being in state s_t, regardless of the specific action taken.

If $\hat{A}_t$ is positive, then the action performed yielded a better outcome than expected and the PPO algorithm will update the policy to make this action more likely in the future. If it is negative the action performed yielded a worse-than-average outcome and the algorithm will update the policy to make this action less likely. By using the advantage function rather than just the raw reward, the algorithm reduces variance and learns more stably, focusing on the relative quality of actions.

PPO was selected over off-policy, continuous-action algorithms such as Soft Actor–Critic (SAC) because the problem formulation relies on a discrete action space (selecting outgoing edges on a DAG) rather than continuous joint torque manipulation. Since the environment operates purely at the high-level logic of task sequencing, the state space is defined solely by the assembly nodes; physical variables such as sensor noise, human skeleton tracking, and kinematic variability are intentionally abstracted away.

3.4. Adversarial Reinforcement Learning Application

An important extension of RL methods occurs when it is necessary to synchronize and optimize a collaborative action strategy between different agents that may be multiple robots or robots and human operators [32]. This study stems from the observation that humans and robots act very differently and that it is not possible to force humans to faithfully execute a certain plan, especially when the chosen strategy does not offer clear advantages. Therefore, robots cannot rely on humans adhering to the plan. Instead, the robot must implement robust strategies that allow tasks to be executed even in the presence of significant deviations from the plan.

To address the unpredictability of human behavior, we introduce an ARL framework. In this setting, the robot and human are modeled as competing agents in a pseudo-game, as in [33].

The Robot's Goal: Complete the assembly (reach the final node of the DAG) as quickly as possible, minimizing the cumulative negative reward.

The Human's Goal: Delay the process, effectively maximizing the path length to test the robot's resilience.

The interaction takes place on a DAG where the agents take turns selecting actions. Unlike standard zero-sum games where one player wins and the other loses even before the end of the assigned tasks, here the assembly process always terminates; the competition is over the cost (time/steps) to reach the end. The human is modeled as an optimal "adversary" strictly as a mathematical proxy during the training phase. We do not assume workers are malicious; rather, training against a theoretical worst-case adversary forces the robot to learn conservative, fail-safe strategies. By finding the absolute upper bound for task completion against an optimal opponent, the resulting robotic policy is robust enough to handle any clumsy, forgetful, or creative human deviations in practical application.

The example in Figure 4 shows the risk of adopting the optimal task sequence in a multi-agent environment. The graph represents all the feasible assembly sequences. The sequence (1,2,9) is the fastest path, three steps long, but if robot choses node 2, the human from node 2 could chose the path (1,2,6,7,8,9), six steps, that is, the longest path in the graph. Conversely if the robot chooses node 3, whichever action the human adopts, the paths will be shorter than the worst case and bounded to a maximum length of 4. Thus, a policy that includes node 3 can be considered robust, even if it is not optimal.

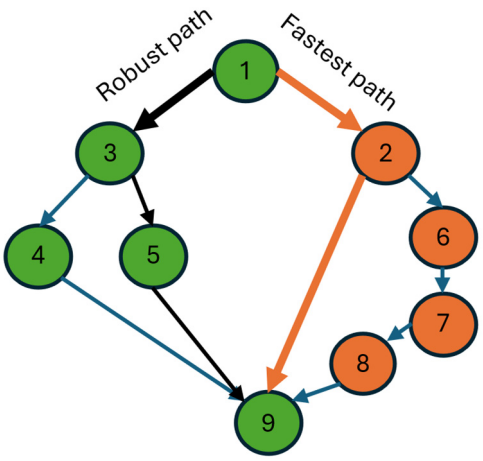


Figure 4. Example of robust against fast path. Orange is the fastest path, green is a robust path.

On-Policy Learning in Adversarial Settings: In our adversarial framework, two agents (robot and human) compete against each other. PPO is an on-policy algorithm that requires data generated by the current policy to perform updates. In a multi-agent adversarial environment, the environment is usually non-stationary because the opponents are learning simultaneously. To address this, we use an alternating training scheme. Fixing the opponent's policy while training the active agent makes the environment temporarily stationary. This allows the active agent to collect valid on-policy trajectories against the opponent's current strategy, effectively optimizing a response to the game's current "goal."

3.4.1. Problem Formulation

The State Space is composed by the nodes of the DAG representing assembly states. The Action Space is the set of outgoing edges (transitions) to the next node. The Rewards are defined in a way that:

- The robot receives a penalty for every time step to encourage speed. Upon reaching the final node, it receives a sparse completion reward.
- The human receives a positive reward for every step the game continues, incentivizing the prolongation of the task.

The human is modeled as an optimal adversary strictly as a mathematical proxy to establish the absolute worst-case boundaries of the assembly task. By forcing the robot

to optimize against an opponent actively maximizing delay, the resulting policy becomes inherently robust enough to accommodate the awkward, forgetful, or creative deviations of a real-world operator without encountering logical deadlocks.

3.4.2. ARL Algorithm

The training process replaces standard simultaneous updates with an iterative, alternating approach. The agents take turns making moves in the environment. During the training phase, we alternate between optimizing the robot agent and the human agent. When the robot is training, the human agent acts as a fixed part of the environment (using its latest policy), and vice versa. This continues until the maximum rewards for both agents stabilize. The resulting robot policy proposes a robust policy to execute the assembly in a workplace where the humans are allowed to deviate from the optimal assembly sequence.

The procedure is formalized in the pseudo-code of Figure 5. Human policy is discarded at the end because it was only used to train the robot to operate under the worst practical conditions. If a human is truly collaborative, however, the robot can follow the fastest policy. However, it is not guaranteed that this policy will be the globally optimal one. It could be suboptimal, which is acceptable for practical applications.

```
Initialize Robot Policy π_R and Human Policy π_H with random weights
Initialize Environment E with DAG structure

Repeat until convergence:
    # --- Phase 1: Train Robot ---
    Freeze Human Policy π_H
    Set Opponent in E to π_H
    For k steps:
        Collect trajectories τ_R by playing π_R against fixed π_H
        Calculate advantages and rewards for Robot
        Update π_R using PPO objective L^CLIP to max Robot reward

    # --- Phase 2: Train Human ---
    Freeze Robot Policy π_R
    Set Opponent in E to π_R
    For k steps:
        Collect trajectories τ_H by playing π_H against fixed π_R
        Calculate advantages and rewards for Human
        Update π_H using PPO objective L^CLIP to max Human reward

    # --- Check Convergence ---
    Evaluate π_R and π_H
    If mean rewards stabilize:
        Break

Return Optimal Robot Policy π*_R
```

Figure 5. Pseudo-code of ARL alternating PPO optimization between robot and human agent.

The pseudocode was used to prompt the Large Language Model (LLM) Google Gemini 3.1 Pro to generate the executable training program. Implementation and training were conducted using a Python-based machine learning stack. PPO reinforcement learning

algorithms were deployed using the Stable-Baselines3 library, which uses PyTorch as its deep learning backend. To ensure seamless integration with the learning agents, the custom collaborative assembly environment was built according to the Gymnasium API standard. NumPy was used for efficient numerical computations and matrix operations, and Cloudpickle handled the serialization and deserialization of the models and environment states (Table 1).

Table 1. Software and library specifications.

Software/Library	Version/Status
Python	3.13.11
Stable-Baselines3	2.7.1
PyTorch	2.9.1
NumPy	2.4.1
Cloudpickle	3.1.2
Gymnasium	1.2.3

Here is the comprehensive list of the training parameters used for the ARL model.

Environment and Graph (structure of the DAG and agents' interaction rules)

Graph Structure: Random Directed Acyclic Graph.
Action Space: Discrete.
Observation Space: Discrete (Node ID).
Turn Structure: Alternating turns (Robot $\leftrightarrow$ Human).

Reward Function for the Robot

Step Penalty -1 (per turn, to encourage speed).
Completion Reward: $+n$ (upon reaching the final node).

Reward Function Human

Step Reward: +1 (per turn, to encourage delay).

Training Loop Parameters

Training Scheme: Alternating (Iterative).
Timesteps per Cycle: 2048 steps (One full PPO buffer collection).
Max Cycles: 100 (Safety termination limit).
Convergence Patience: Five checks.
Convergence Threshold: 0.5 (Maximum difference in mean reward over the last five checks).

PPO Algorithm Hyperparameters (Stable Baselines3 Defaults)

Policy Architecture: MlpPolicy (Multi-Layer Perceptron).
Learning Rate: 3×10^{-4}.
n_steps (Buffer Size): 2048.
Batch Size: 64.
n_epochs: 10.
Gamma (Discount Factor): 0.99.
GAE Lambda: 0.95.
Clip Range: 0.2.
Entropy Coefficient: 0.0.

4. Results

4.1. Definition of the Performance Metrics

To validate the performance of the proposed ARL algorithm, a combination of approaches is used, including both quantitative efficiency metrics and robustness indicators [34].

- Efficiency metrics focus on the baseline performance of the system, primarily the Task Completion Time (TCT), which measures the total number of steps required to traverse the assembly DAG from start to finish. This is the absolute minimum number of steps if the human cooperated perfectly (or if the robot controlled both turns). While ARL is not expected to outperform purely optimal planning in ideal conditions, TCT serves as a benchmark to ensure the resilient policy remains within acceptable productivity limits.
- Robustness metrics, which directly address the ability and benevolence components of trust, are critical for demonstrating resilience. Key indicators include:
    a. Worst-Case Path Length (WCPL): the number of steps to complete the task when the robot contrasts the optimal policy of the human that is trying to delay the process.
    b. Resilience Ratio (RR): comparing WCPL against the distribution of all possible path lengths. A high percentile ranking confirms the robot's ability to mitigate human variability. To calculate it the script runs 1000 simulations of the robot (optimal) vs. human (random). The ratio calculates the percentage of random trials that finished within the time bound established by the adversarial case. A 100% ratio confirms the robot has effectively learned a robust upper bound.

By evaluating these indicators, we can confirm that the robot does not merely optimize for speed but provides a reliable, bounded, and supportive collaboration framework.

4.2. Definition of a Case Study

For ease of exposition, we present a laboratory case study extracted from a longer industrial process. This case study involves assembling a turbomolecular vacuum pump (Figure 6) in the collaborative robotics laboratory at Politecnico di Torino. This process uses a collaborative cell with two UR3e cobots (R1 and R2) and a human operator (H) working together in a shared workspace to increase efficiency. The components are identified by specific acronyms in the bill of materials (BOM): the main body (BD) serves as the base for the plastic case (PC) and bottom cap (BC). The assembly is then mounted onto an envelope (EV) and completed with a foreline flange (FOR). Various screws are used for fastening: VT1 (M3×8), VT2 (hex M4×10), VT3 (hex M5×20), and VT4 (hex M3×8). This process is ideal for collaborative execution by humans and cobots.

The human operator (H) acts as the supervisor of the robotic cell and handles tasks that require dexterity, decision, and flexibility. The operator controls the timing and flow of the assembly, ensuring the cobots only move when manual tasks are completed. H is responsible for changing the bits on the robotic screwdriver and positioning screws into holes and also performs final manual screwing operations that the robot cannot execute due to part geometry.

The Universal Robot UR3e installed on the right side of the workbench is equipped with an automatic screwdriver (R1). It is exclusively dedicated to screwing tasks (Figure 7). The UR3e cobot installed on the left side is equipped with a two-jaw parallel gripper (R2). It handles the movement of components and tools across the workspace (Figure 8), lifts and holds components steady during manual assembly phases.

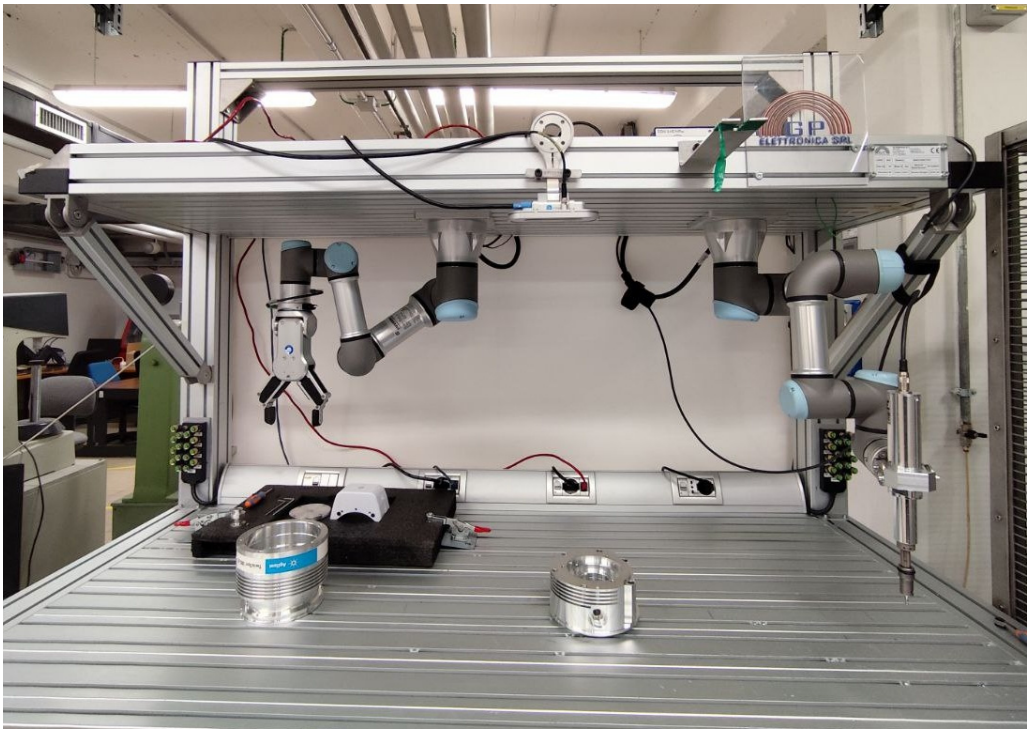


Figure 6. Case study for the collaborative assembly process: a turbomolecular vacuum pump. The workstation is equipped with ceiling-mounted robotic arms: the left is equipped with a gripper, the right with a screwdriver.

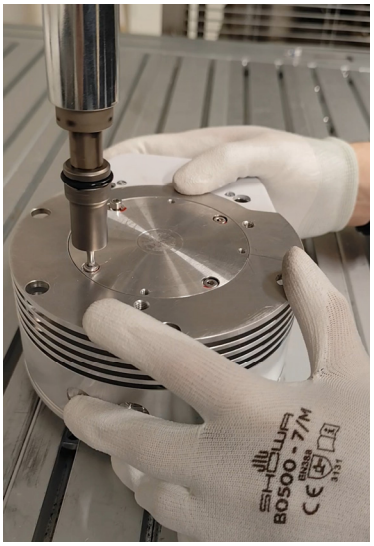


Figure 7. Collaborative operation: R1 is screwing while H holds the part in position.

The assembly sequence (Table 2) consists of five main tasks and several sub-tasks. In task one, R1 screws the plastic case onto the main body. In task two, the bottom case is positioned on the body and then screwed into place by R1. R2 handles the parts to be assembled in both tasks. In task three, R2 picks up and places tools and parts. This task is fully automated without human collaboration. In task four, H1 manually screws the plastic case onto the body while R2 lifts and supports it. In task five, R2 positions the assembled body onto the envelope. The human operator (H1) then manually screws them together to complete the process.

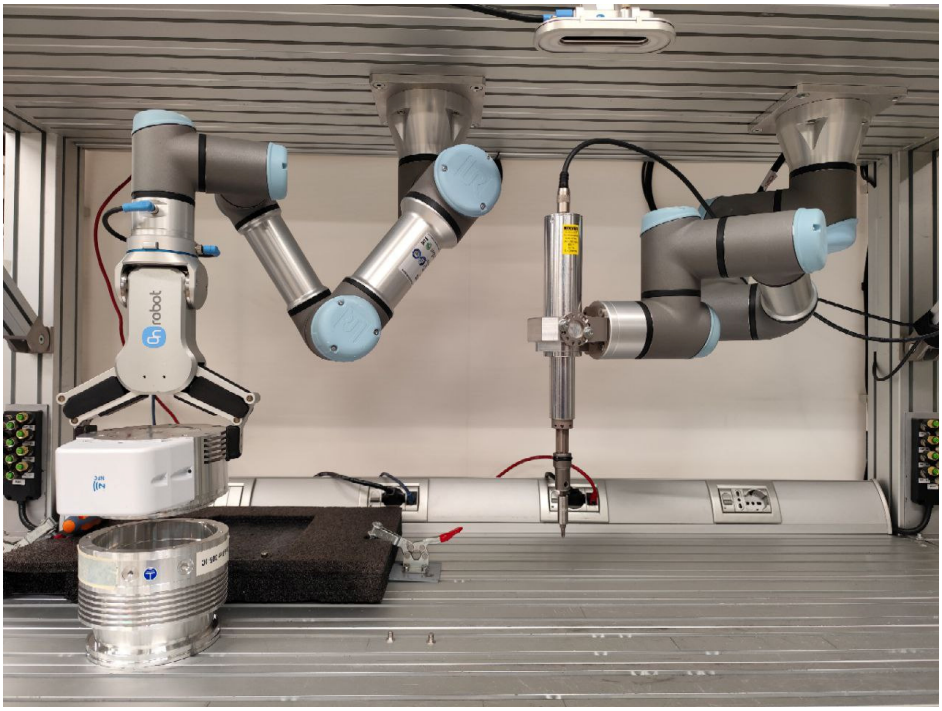


Figure 8. Assembly operation executed only by the robot without human assistance.

Table 2. Assembly sequence for the turbomolecular vacuum pump.

Task	Operation	Description	Assigned
1.1	Input confirmation	Confirm start of R2 movement	H
1.2	Bit change	Replace hex bit with cross bit on R1	H
1.3	Component move	Move PC to the work area	R2
1.4	PC Positioning	Place PC and 2× VT1 screws on BD	H
1.5	Component move	Move BC to the work area	R2
1.6	Screwing	Screw VT1 (0.5 Nm) while holding BD	R1
2.1	Bit change	Replace cross bit with hex bit on R1	H
2.2	BC Positioning	Place BC on BD	H
2.3	Component move	Position 4×x VT2 screws on BC	R2
2.4	Screw placement	Insert VT2 screws into holes	H
2.5	Screwing	Screw VT2 (1.5 Nm) while holding BD	R1
3.1	Tool move	Move screwdriver to work area	R2
3.2	Tool move	Move hex keys to work area	R2
3.3	Component move	Move FOR to work area	R2
4.1	Input confirmation	Move BD position	H
4.2	Support	Lift and hold BD	R2
4.3	Manual screwing	Position and screw 2×x VT1 (0.5 Nm)	H
5.1	Input confirmation	Confirm BD movement	H
5.2	BD Positioning	Place BD on EV	R2
5.3	Manual screwing	Position and screw 6×x VT3 (3 Nm)	H
5.4	Final assembly	Position and screw FOR and VT4 (1.5 Nm)	H

The assembly plan is shown in the DAG in Figure 9. For simplicity, tasks three, four, and five are combined into one final node. In this case study, the human operator is expected to replace the hex bit with a cross bit on the screwdriver on R1. The expected assembly sequence is therefore the bottom path on the graph with increasing task numbers. However, it is likely that H forgets to change the bit, leaving the hex bit mounted, and positions BC. This would be the same as executing tasks 21 and 22 instead of 12. In conventional manufacturing, this would result in an exception stop, causing the operator to lose time

understanding the cause of the stop. With the ARL strategy, however, the robot finds an alternative path on the graph and executes the process accordingly.

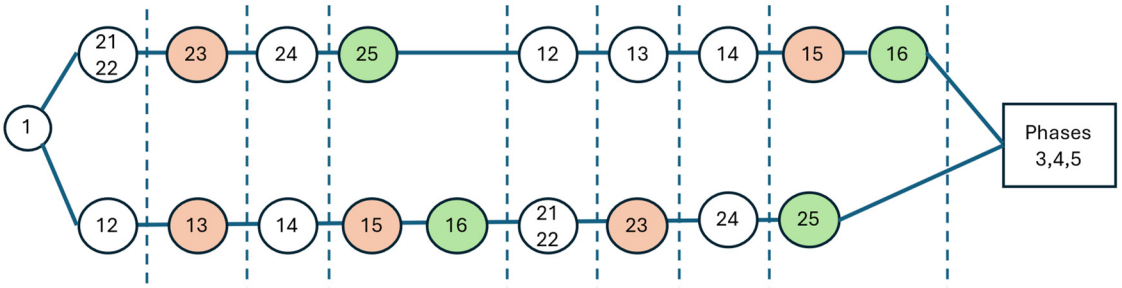


Figure 9. DAG for the vacuum pump assembly. White is used for operations assigned to H, orange to R2 and green to R1. Dash lines separate robot turns from human ones. The assembly sequence has been simplified as phases three, four, and five were not detailed.

The strategy proves effective, but the use of ARL in this simple example is clearly an overuse of computational force. While this laboratory scenario effectively serves as a baseline proof-of-concept, it is relatively simple and does not fully capture the combinatorial explosion of tasks found in complex industrial environments. To critically demonstrate the actual benefit of the proposed ARL method—especially when compared to classical planning baselines that optimize purely for the Theoretical Shortest Path—it is necessary to test ARL on more complex, synthetically generated work situations.

4.3. Execution of the Experiment on Synthetic Data

The experimental plan utilizing synthetic assembly sequences was specifically designed to explore conditions that are both favorable and unfavorable for the collaborative robot. Conditions are considered favorable when the human operators face few choices, meaning they cannot deviate significantly from the established, optimal path. Conversely, unfavorable conditions occur when the graph features many layers, allowing certain path choices to drastically extend the time required to conclude the assembly. The main performance metrics resulting from the experimental runs are summarized in Table 3.

Table 3. Performance metrics on synthetic data for graphs with different layers and max number of nodes per layer. The metrics considered are: TCT, WCPL, RR.

Layers/Max Nodes	TCT	WCPL	Max Random Length	Avg Random Length	RR
4/10	1	4	5	3.86	87.90%
10/4	2	7	8	4.57	94.20%
10/10	2	9	11	8.18	93.40%
15/10	5	10	12	8.18	93.30%
20/10	4	11	18	10.75	71%
20/20	3	14	19	12.28	85%

Since the graphs are generated randomly, the specific metric values will change with each new run of the experiment. To account for this variability, an additional experiment was conducted focusing specifically on the 20/10 configuration (20 layers with a maximum of 10 nodes per layer), where 10 distinct graphs were generated to evaluate the statistical distribution of the results. The results are reported in Table 4.

Table 4. Performance metrics on repeated experiment: TCT, WCPL, RR.

Trial	Total Nodes	TCT	WCPL	Avg Random Length	RR
1	76	5	11	11.3	62.60%
2	70	6	17	13.32	97.00%
3	59	2	11	10.29	54.50%
4	56	4	8	7.54	61.50%
5	62	3	5	9.32	28.30%
6	64	6	10	10.16	63.00%
7	73	3	16	12.31	98.70%
8	69	4	15	10.23	97.70%
9	77	6	13	12.11	62.50%
10	68	5	11	10.77	66.80%

5. Discussion

5.1. Analysis of the Experimental Plan

We begin by analyzing a specific instance in detail: the 15/10 graph configuration. This randomly generated graph consists of 60 nodes and 162 edges, starting at Node 0 and ending at Node 59 (Figure 10). The performance metrics are: TCT five steps, WCPL ten steps, RR 93.3%. The Optimal Robot Policy against the optimal human adversary is (0, 3, 10, 18, 30, 33, 44, 47, 54, 56, 59).

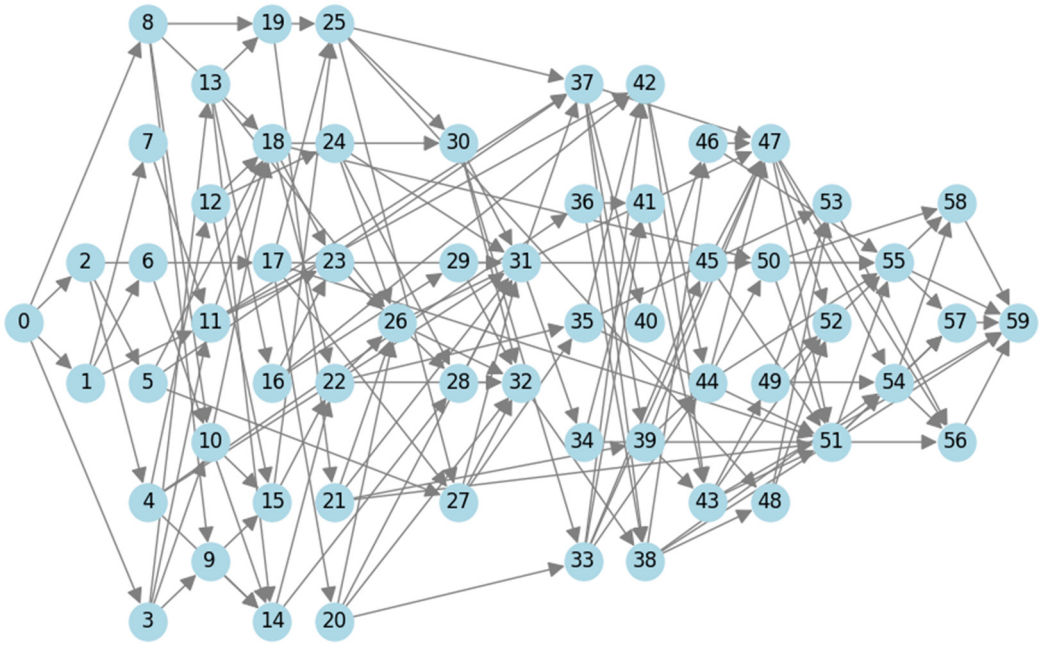


Figure 10. Random generated DAG with fifteen inner layers, ten nodes maximum per layer, and four actions maximum per node.

Figure 11 shows the cumulative rewards earned by the two competing agents, robot and human, over approximately 7500 episodes of training. To ensure clarity amidst the noise of the raw data (the faint background lines), both trajectories are smoothed using a 50-episode moving average (MA50). By the latter stages of training, the human agent achieves a stable, high-performance state, plateauing at a reward of approximately 6.0. Conversely, the robot agent settles into a lower-reward equilibrium of 5.0. The reduction in the amplitude of the moving average lines toward the end indicates that both policies have reached a point of convergence.

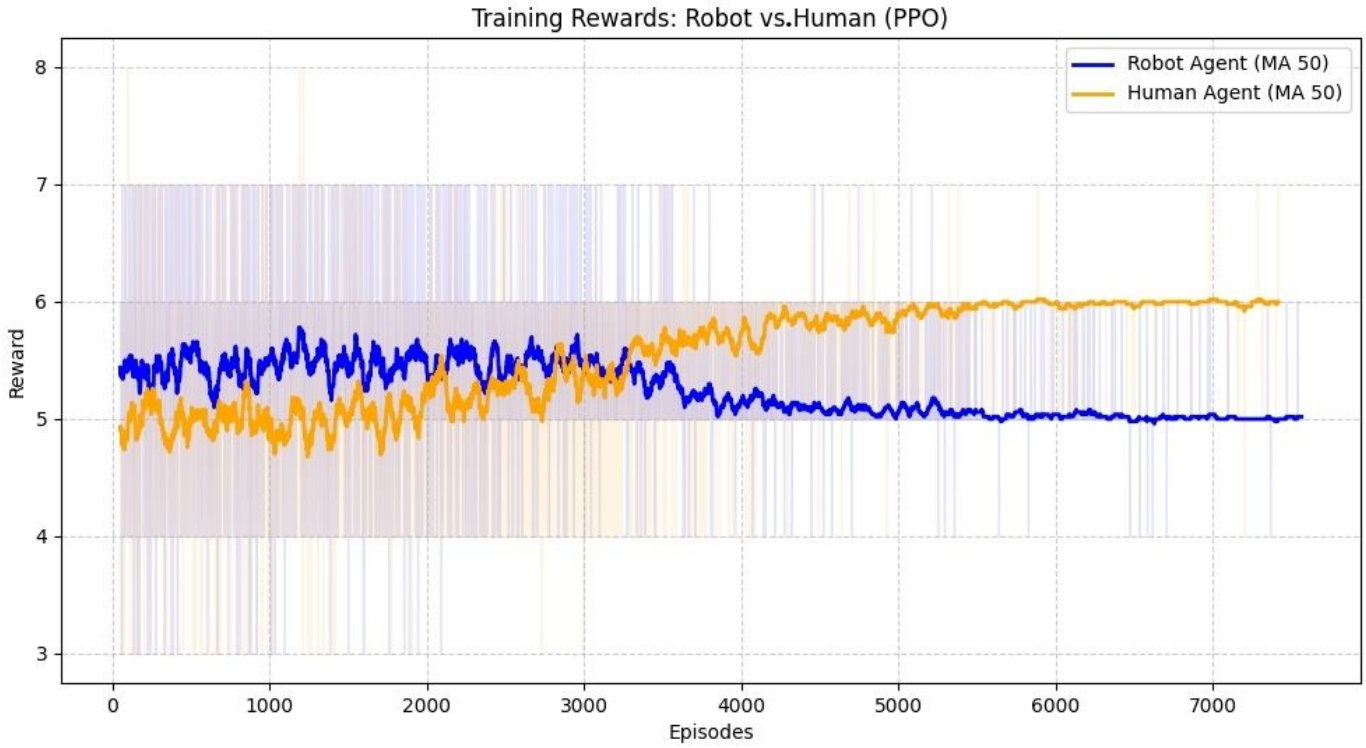


Figure 11. Training reward dynamics for ARL using PPO. The human agent (orange) eventually stabilizes at a higher reward plateau than the robot agent (blue).

Note that the human policy is not reported among the results because it is not used during execution; it serves strictly as an adversarial tool during training. To correctly interpret these results, the peculiar characteristics of this model must be clarified. Although this framework is frequently referred to as a "game" in reinforcement learning contexts, it does not represent a fair game. The random generation of the graph structure inherently favors either the human or the robot policy. Consequently, for certain graph topologies, the training phase will predictably conclude with a higher accumulated reward for the human agent, while for others, the robot will always gain a better reward.

It is important to understand that a higher human reward does not mean the robot "lost" the scenario. Rather, it indicates the graph topology naturally has longer traversal paths, resulting in a higher score for the delay-seeking human. A robot's objective is not to "win" by achieving the highest score, but rather to prevent a human from wasting time on inefficient operations. This dynamic accurately mirrors real-world manufacturing environments: some assembly processes are faster and simpler than others and present fewer opportunities for error or alternative choices. The primary metric of success is that the robot can drive the assembly to completion in every scenario without freezing or getting stuck. This remains a common challenge in industrial applications involving collaborative robots. Therefore, the resilience ratio is the most significant metric to consider in this study.

The robot's strategy of selecting actions that prevent the human from entering "dead-end" or highly inefficient states demonstrates benevolence. It does not simply maximize its own reward; it adapts its behavior to protect the collaborative effort from its partner's potential unpredictability. In a real industrial setting, this means the robot effectively "covers" for the human worker, reducing anxiety and frustration associated with errors.

Shifting from optimizing for pure speed to optimizing for resilience and adaptability directly addresses the "flexibility" barrier in human–robot collaboration (HRC). Ensuring the system is robust against the variable behavior of the human "adversary" (a proxy for human variability) creates a system that warrants trust.

5.2. Analysis of the Repeated Experiment

The repeated experiment on the 20/10 configuration gives the results of Table 4. Three main considerations can be drawn from them.

High Environmental Variance: The randomized graph structure significantly impacts the robot's likelihood of success. As shown in Figure 12, the resilience ratio was 97% or higher in trials two, seven, and eight, meaning the optimal robot policy nearly always outperformed random behavior. Conversely, trial five had a resilience ratio of just 28.3%. This suggests that certain graph topologies likely contain choke points or edge configurations that favor the human adversary, preventing the robot from guaranteeing a path that is significantly better than random.

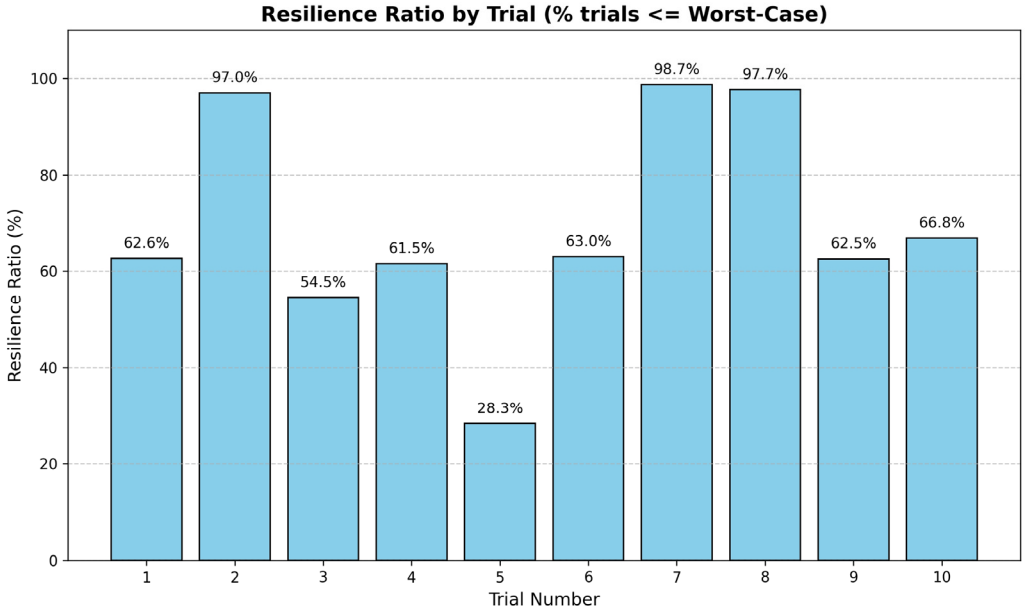


Figure 12. Resilience ratio for the different trials.

The "Adversarial Tax": The difference between the shortest possible theoretical paths (ranging from two to six steps) and the robot's optimal policy (ranging from five to seventeen steps) is substantial. This metric perfectly illustrates the effectiveness of the adversary. By acting optimally, the human forces the robot to take paths that are, on average, over two and a half times longer than the shortest possible route (Figure 13). Obviously, this is an extreme scenario, not standard practice. The human coworker is expected to follow a collaborative policy most of the time.

Optimal vs. Random Expectations: Interestingly, the overall average of the optimal robot's worst-case policy (11.7 steps) is slightly higher than the average random length (10.73 steps). This occurs because the worst-case metric measures a guaranteed upper bound against an optimal adversary. Random trials average out human moves and reflect scenarios in which a smart robot can capitalize on human choices to cross the graph much faster.

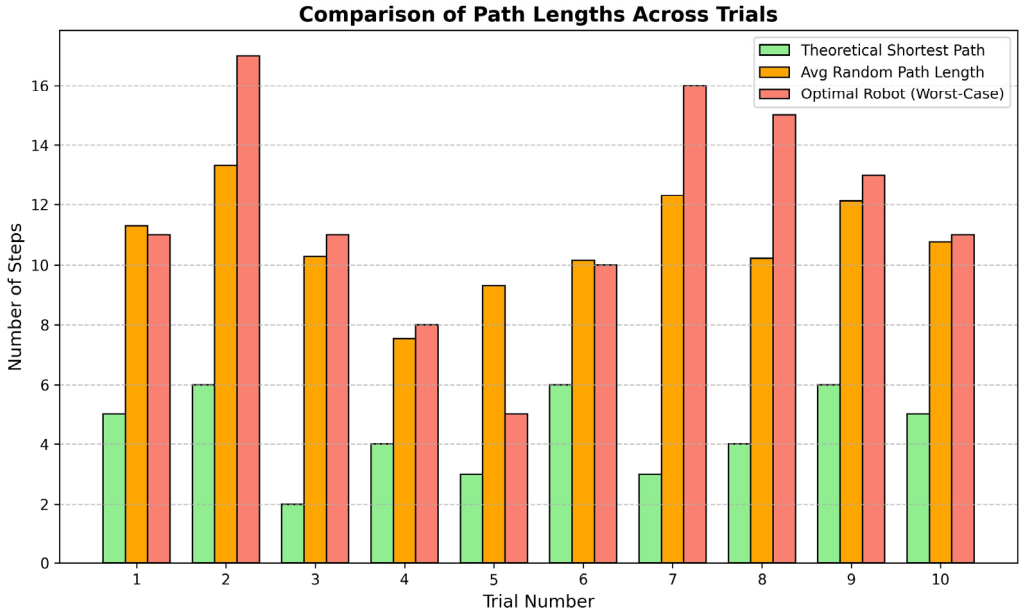


Figure 13. Comparison of path lengths: shortest path, average random, optimal policy.

5.3. Implications for Trust

The results of the experimental analysis highlight a critical conceptual connection between control strategy and trust. However, it is important to explicitly acknowledge that the connection to the ABI trust model presented here remains theoretical; trust is inferred from the mathematical guarantees of the resilient policy rather than experimentally validated through human–subject testing. In all simulated scenarios, the robot did not simply "hope" for the optimal path. Instead, it secured a path that serves as a guaranteed upper bound. This capability maps directly to the ABI model:

- Ability: The robot demonstrates competence by consistently managing complex task sequences and avoiding deadlocks or excessive delays.
- Benevolence: By adapting its strategy to mitigate potential human errors (simulated by the adversary), the robot acts in the best interest of the team, reducing the burden on the human operator to perform perfectly.

By converting a potentially chaotic distribution of assembly sequences into a bounded, predictable duration, the ARL framework establishes a foundation for high-trust collaboration.

5.4. Limitations

Despite the promising technical results, this study has some limitations that must be acknowledged.

The conceptual link between the proposed ARL framework and the ability and benevolence dimensions of the ABI trust model offers a theoretical rationale for increased human–robot trust. However, this relationship has not yet been empirically validated through experimentation with human subjects. The present study relies on simulations and controlled laboratory experiments, but it does not incorporate validated trust questionnaires, behavioral indicators, or psychometric assessment tools. Therefore, while the framework establishes performance guarantees and transparency mechanisms that are theoretically conducive to trust formation, direct experimental confirmation of increased human trust is still needed.

Furthermore, real-world validation is currently limited to a laboratory case study involving a turbomolecular vacuum pump, which primarily serves as a proof of concept. While this scenario is suitable for demonstrating feasibility, it does not fully challenge

the adaptive and robust capabilities of the proposed methodology in complex industrial environments. Consequently, part of the algorithmic evaluation relies on synthetic, randomly generated datasets. While this allows for controlled and systematic benchmarking, it does not fully capture the variability and uncertainty that are typical of real operational conditions. Therefore, broader validation in high-complexity industrial settings is necessary to demonstrate scalability and practical robustness.

6. Conclusions

This study introduces an ARL framework designed to promote trust in human–robot collaborative assembly. By modeling the assembly process as an MDP on a DAG and utilizing an alternating PPO training scheme, the robot learns a robust policy that can mitigate the unpredictability of human partners. Experimental results on synthetic data and a case study demonstrate the robot's ability to complete tasks within a timeframe, despite human deviations. This aligns with the ability and benevolence components of the ABI model of trust. While an ARL-trained robot may not achieve the absolute minimum TCT under ideal conditions, it offers a reliable and resilient alternative to traditional, rigid assembly optimization algorithms. However, the effectiveness of the ARL strategy depends heavily on the assembly graph topology. While most trials showed high resilience (e.g., RR of 97% or higher), some configurations revealed that certain "choke points" or complex edge configurations can significantly limit the robot's ability. Adopting a robust strategy may result in path lengths longer than the shortest theoretical route, but this is a necessary cost for avoiding work interruptions due to a lack of agreement between humans and robots. In summary, the proposed ARL framework shifts the focus from narrow time optimization to a broader commitment to work environment quality and system reliability. By demonstrating adaptability in the face of uncertainty, the robot fosters a collaborative environment where human operators feel supported rather than constrained.

To bridge the gap between this high-level logical framework and physical execution, ongoing research involves integrating a global camera view of the workstation with a vision-language model (VLM) to dynamically identify and track the specific assembly tasks performed by the human operator.

Future research will explore the implementation of Hybrid Reinforcement Learning (HRL). By integrating the PPO agent with deeper, logic-based structures—such as an Assembly Sequence Graph—we can further ensure that the RL model remains robust and strictly adheres to kinematically feasible constraints, bridging the gap between high-level task planning and lower-level execution (as explored in recent HRL literature, e.g., [35]).

Supplementary Materials: The following supporting information can be downloaded at: https://www.mdpi.com/article/10.3390/app16073244/s1. The verified Python script and ARL environment code utilized for this study are provided in the Supplementary Materials.

Author Contributions: Conceptualization, D.A.; methodology, D.A.; software, D.A.; validation, D.A., K.A. and B.Y.; formal analysis, D.A.; investigation, K.A.; resources, K.A.; data curation, D.A.; writing—original draft preparation, D.A.; writing—review and editing, K.A. and B.Y.; visualization, D.A.; supervision, D.A.; project administration, K.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original contributions presented in the study are included in the article/Supplementary Material, further inquiries can be directed to the corresponding author.

Acknowledgments: During the preparation of this study, the authors used GEMINI 3.1 PRO to generate the code for the ARL algorithm. The authors have reviewed and edited the output and take full responsibility for the content of this publication.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

HRC	Human–Robot Collaboration
RL	Reinforcement Learning
ARL	Adversarial Reinforcement Learning
DAG	Directed Acyclic Graph
AAP	Automated Assembly Planning
SME	Small and Medium-sized Enterprises
ABI	Ability, Benevolence, Integrity
ASP	Assembly Sequence Planning
APP	Assembly Path Planning
PPO	Proximal Policy Optimization
MDP	Markov Decision Process
TCT	Task Completion Time
WCPL	Worst-Case Path Length
RR	Resilience Ratio
VLM	Vision Language Model

References

1. Wang, L.; Keshavarzmanesh, S.; Feng, H.Y.; Buchal, R.O. Assembly process planning and its future in collaborative manufacturing: A review. *Int. J. Adv. Manuf. Technol.* **2009**, *41*, 132–144. [CrossRef]
2. Del Real Torres, A.; Andreiana, D.S.; Ojeda Roldan, A.; Hernandez Bustos, A.; Acevedo Galicia, L.E. A review of deep reinforcement learning approaches for smart manufacturing in industry 4.0 and 5.0 framework. *Appl. Sci.* **2022**, *12*, 12377. [CrossRef]
3. Dieber, B.; Schlotzhauer, A.; Brandstötter, M. Safety and Security—Success factors of sensitive robotic technologies. *Elektrotechnik Informationstechnik* **2017**, *134*, 299–303. [CrossRef]
4. Baumgartner, M.; Kopp, T.; Kinkel, S. Analysing factory workers' acceptance of collaborative robots: A web-based tool for company representatives. *Electronics* **2022**, *11*, 145. [CrossRef]
5. Bragança, S.; Costa, E.; Castellucci, I.; Arezes, P.M. A brief overview of the use of collaborative robots in industry 4.0: Human role and safety. In *Occupational and Environmental Safety and Health*; Springer: Cham, Switzerland, 2019; pp. 641–650.
6. Jain, R.; Garg, N.; Khera, S.N. Comparing differences of trust, collaboration and communication between human-human vs human-bot teams: An experimental study. *CERN IdeaSquare J. Exp. Innov.* **2022**, *7*, 8–16.
7. Haas, M.; Mortensen, M. The secrets of great teamwork. *Harv. Bus. Rev.* **2016**, *94*, 70–76.
8. Mayer, R.C.; Davis, J.H.; Schoorman, F.D. An integrative model of organizational trust. *Acad. Manag. Rev.* **1995**, *20*, 709–734. [CrossRef]
9. Khalid, H.; Helander, M.; Lin, M. Determinants of trust in human-robot interaction: Modeling, measuring, and predicting. In *Trust in Human-Robot Interaction*; Academic Press: Cambridge, MA, USA, 2021; pp. 85–121.
10. Maderna, R.; Pozzi, M.; Zanchettin, A.M.; Rocco, P.; Prattichizzo, D. Flexible scheduling and tactile communication for human–robot collaboration. *Robot. Comput. Integr. Manuf.* **2022**, *73*, 102233. [CrossRef]
11. Inkulu, A.K.; Bahubalendruni, M.R.; Dara, A. Challenges and opportunities in human robot collaboration context of Industry 4.0—A state of the art review. *Ind. Robot Int. J. Robot. Res. Appl.* **2022**, *49*, 226–239. [CrossRef]
12. Masehian, E.; Ghandi, S. Assembly sequence and path planning for monotone and nonmonotone assemblies with rigid and flexible parts. *Robot. Comput. Integr. Manuf.* **2021**, *72*, 102180. [CrossRef]
13. Peta, K.; Suszyński, M.; Wiśniewski, M.; Mitek, M. Analysis of Energy Consumption of Robotic Welding Stations. *Sustainability* **2024**, *16*, 2837. [CrossRef]
14. Peta, K.; Wiśniewski, M.; Kotarski, M.; Ciszak, O. Comparison of Single-Arm and Dual-Arm Collaborative Robots in Precision Assembly. *Appl. Sci.* **2025**, *15*, 2976. [CrossRef]

15. Lazzerini, B.; Marcelloni, F. A genetic algorithm for generating optimal assembly plans. *Artif. Intell. Eng.* **2000**, *14*, 319–329. [CrossRef]
16. Li, M.; Wu, B.; Yi, P.; Jin, C.; Hu, Y.; Shi, T. An improved discrete particle swarm optimization algorithm for high-speed trains assembly sequence planning. *Assem. Autom.* **2013**, *33*, 360–373. [CrossRef]
17. Han, Z.; Wang, Y.; Tian, D. Ant colony optimization for assembly sequence planning based on parameters optimization. *Front. Mech. Eng.* **2021**, *16*, 393–409. [CrossRef]
18. Karthik, G.; Deb, S. A methodology for assembly sequence optimization by hybrid cuckoo-search genetic algorithm. *J. Adv. Manuf. Syst.* **2018**, *17*, 47–59. [CrossRef]
19. Malek, N.; Peng, Q. Reinforcement learning for self-adaptive genetic algorithm in assembly sequence planning. *Int. J. Adv. Manuf. Technol.* **2025**, *141*, 4803–4822. [CrossRef]
20. Suszyński, M.; Peta, K. Assembly sequence planning using artificial neural networks for mechanical parts based on selected criteria. *Appl. Sci.* **2021**, *11*, 10414. [CrossRef]
21. Masehian, E.; Ghandi, S. ASPPR: A new assembly sequence and path planner/replanner for monotone and nonmonotone assembly planning. *Comput. Aided Des.* **2020**, *123*, 102828. [CrossRef]
22. Liu, J.-C.; Chang, C.-H.; Sun, S.-H.; Yu, T.-L. Integrating planning and deep reinforcement learning via automatic induction of task substructures. In Proceedings of the Twelfth International Conference on Learning Representations, Vienna, Austria, 7–11 May 2024.
23. Lettera, G.; Natale, C. An Integrated Architecture for Robotic Assembly and Inspection of a Composite Fuselage Panel with an Industry 5.0 Perspective. *Machines* **2024**, *12*, 103. [CrossRef]
24. Mateus, J.; Aghezzaf, E.H.; Claeys, D.; Limère, V.; Cottyn, J. Method for transition from manual assembly to human-robot collaborative assembly. *IFAC-PapersOnLine* **2018**, *51*, 405–410. [CrossRef]
25. Gottipolu, R.B.; Ghosh, K. A simplified and efficient representation for evaluation and selection of assembly sequences. *Comput. Ind.* **2003**, *50*, 251–264. [CrossRef]
26. Deepak, B.B.; Bala Murali, G.; Bahubalendruni, M.R.; Biswal, B.B. Assembly sequence planning using soft computing methods: A review. *Proc. Inst. Mech. Eng. Part E J. Process Mech. Eng.* **2019**, *233*, 653–683. [CrossRef]
27. Aliev, K.; Antonelli, D.; Bruno, G. Task-based programming and sequence planning for human-robot collaborative assembly. *IFAC-PapersOnLine* **2019**, *52*, 1638–1643. [CrossRef]
28. Heath, L.; Pemmaraju, S.; Trenk, A. Directed Acyclic Graphs. *Planar Graphs* **1992**, *9*, 5.
29. Sutton, R.; Barto, A.G. Reinforcement learning. *J. Cogn. Neurosci.* **1999**, *11*, 126–134.
30. Puterman, M.L. Markov decision processes. *Handb. Oper. Res. Manag. Sci.* **1990**, *2*, 331–434.
31. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal policy optimization algorithms. *arXiv* **2017**, arXiv:1707.06347. [CrossRef]
32. Antonelli, D.; Zeng, Q.; Aliev, K.; Liu, X. Robust assembly sequence generation in a Human-Robot Collaborative workcell by reinforcement learning. *FME Trans.* **2021**, *49*, 851–858. [CrossRef]
33. Zhao, H.; Liang, Z.; Ma, T.; Shi, X.; Kapadia, M.; Thrash, T.; Hoelscher, C.; Jia, J.; Liu, B.; Cao, J. Adversarial Reinforcement Learning for Enhanced Decision-Making of Evacuation Guidance Robots in Intelligent Fire Scenarios. *IEEE Trans. Comput. Soc. Syst.* **2024**, *12*, 2030–2046. [CrossRef]
34. Pinto, L.; Davidson, J.; Sukthankar, R.; Gupta, A. Robust adversarial reinforcement learning. In Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, 6–11 August 2017; pp. 2817–2826.
35. Hazem, Z.B.; Saidi, F.; Guler, N.; Altaif, A.H. A Hybrid Reinforcement Learning Framework Combining TD3 and PID Control for Robust Trajectory Tracking of a 5-DOF Robotic Arm. *Automation* **2025**, *6*, 56. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.