Enhancing Trust in Collaborative Assembly through Resilient Adversarial Reinforcement Learning

Dario Antonelli *, Khurshid Aliev, Bo Yang

Posted Date: 5 March 2026

doi: 10.20944/preprints202603.0401.v1

Keywords: human-robot collaboration; trust and safety in robotics; human-centered automation; adversarial reinforcement learning; automated assembly planning

 doi:10.20944/preprints202603.0401.v1

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

[2] State Key Laboratory of Mechanical Transmission, Chongqing University, Chongqing 400044, China

* 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 (cobots) are designed to improve productivity and safety in industrial settings. However, to be 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 human behavior, particularly when the human teammate has a behavior unpredictable and outside the box. To achieve this adaptability, we propose an Adversarial Reinforcement Learning (ARL) framework to the activity planning of the robot. The assembly process is modeled 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 acts as an adversary trained with the same algorithm to introduce disturbances and delays. The proposed approach was applied to a simple industrial case study and evaluated on complex assembly sequences generated synthetically. While the ARL-trained robot did not outperform conventional assembly optimization algorithms in terms of task completion time, it guaranteed robustness against human variability, ensuring task completion within a bounded timeframe regardless of human actions. By demonstrating consistent performance and adaptability (Ability) in the face of uncertainty, the robot exhibits characteristics that align with the Ability and Benevolence components of the ABI model of trust, thereby fostering 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

1. Introduction

The manufacturing sector is currently going through an era of unique transformation, characterized by the convergence of high-performance computing, advanced mechatronics, and autonomous decision-making systems. Central to this shift is the domain of Automated Assembly Planning (AAP), a critical research area that addresses the challenge of transforming static digital designs of a product into executable manufacturing processes [1]. In the context of rising challenges in global markets, characterized by intensifying labor shortages, escalating operational costs, and demand for extreme product customization, there is a request for the development of highly flexible and intelligent assembly systems, underscoring the need for innovative solutions to address these prevailing issues. [2]. Flexibility in automation can be achieved by the deployment of collaborative robots, or 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

 doi:10.20944/preprints202603.0401.v1

implementation costs, safety concerns, and the complexity of programming robots for collaborative environments [5].

To overcome these barriers, it is essential to look beyond the technical specifications and consider the dynamics of the team itself. Effective human collaboration relies on shared goals, open communication, defined roles, and, crucially, trust [6,7]. While task assignment and scheduling are well-researched, the psychological aspects of HRC, specifically trust, remain less explored in the context of robot control strategies.

Trust in organizational and collaborative settings 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 are the basis of an objective analysis of trust in HRC situations [9]. In typical industrial scenarios, robots are programmed for rigidity and repetition, which clashes with the flexibility required in human teams. Flexibility, the ability to adapt to changing circumstances and resilience in the face of stress, is a characteristic of effective teamwork [10,11]. If a robot cannot cope with human variability (e.g., mistakes, fatigue, or creative problem solving), it fails to demonstrate Ability (robustness) and Benevolence (helpfulness), thereby eroding trust.

This paper proposes that trust in HRC benefits directly from the robot's ability to adapt to human behavior. The following research question (RQ) is addressed:

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

Adversarial Reinforcement Learning (ARL) is employed to train task planning on 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 forces the robot to learn conservative, resilient strategies that mitigate the impact of human variability at the expense of a slight increase in process times. By guaranteeing task completion within a reasonable time regardless of human actions, the robot demonstrates high Ability and a form of functional Benevolence, laying the foundation for a high-trust collaborative relationship.

The original contribution of this study lies in its proposal of an alternative approach to the AAP problem. Rather than focusing solely on achieving the minimum completion time, the optimal goal is identified as the establishment of a high level of trust between the human operator and the robot. This objective is to be realized through the adoption of ARL framework during the training process to robotic planning.

2. State of the Art in AAP

The objective of AAP, in a nutshell, is to translate the CAD designs of products into executable robotic code. The computational core of AAP is branched into two primary, interdependent subproblems: Assembly Sequence Planning (ASP) and Assembly Path Planning (APP). Both problems are computationally intensive [12]. APP serves to generate a collision-free trajectory for each component from its initial kitting location to its final pose in the assembly and is out of the scope of present paper.

 doi:10.20944/preprints202603.0401.v1

ASP involves determining the optimal order in which components should be combined. The classic solution methods utilize interference matrices, precedence graphs, and sequence-relation matrices to define the geometric and technological constraints of a product [12]. The optimization of the graphs is achieved using heuristic and meta-heuristic search algorithms: Genetic Algorithms [13], Particle Swarm Optimization [14], Ant Colony Optimization [15], and Cuckoo Search [16]. Recent advances have introduced the Q-Learning-based Genetic Algorithm, which enhances traditional Genetic Algorithms by incorporating Reinforcement Learning (RL) [17]. This hybrid approach allows the algorithm to jump out of local optima and handle complex parallel constraints more effectively 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 [18]. Otherwise, RL allows agents to learn optimal strategies by interacting with a simulated environment and maximizing a cumulative reward [17]. In the context of ASP, the agent's objective is to disassemble or assemble a product in the shortest time respecting all the assembly constraints [19]. Recent research focuses on integrating RL with classical planning to account for the inner hierarchical nature of assembly task [20].

With Industry 5.0 and the growing prominence of SMEs, assembly planning evolves from a narrow pursuit of time optimization to a broader commitment to enhancing the overall quality of the work environment [21]. Similarly, this study moves beyond the optimization of standard planning metrics; in its place, it incorporates human variability into the decision-making framework, enabling the robot to demonstrate the adaptability and reliability necessary to foster trust. Therefore, the desired result is that the robot follows an optimal assembly sequence if supported by the human partner, otherwise it searches for the best obtainable sequence based on the choices made by the human. To reach this scope the RL solution to the AAP problem is extended to introduce the human factor as a contrasting agent in an ARL framework.

3. Materials and Methods

3.1. Task Decomposition for Collaborative Assembly

To address the AAP problem, we adopt a hierarchical model of assembly [22]. The assembly job is decomposed into tasks and operations as described by Figure 1.

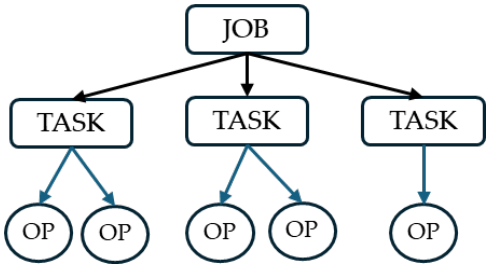

Figure 1. Hierarchical model of assembly job decomposed in tasks and related operations.

The assembly sequence can be formalized by defining a set of operations and applying specific constraints to generate an optimal Assembly Sequence Table (AST). By referring to the study of Gottipolu et al. [23], 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 AT_i:

$$AT_i = \langle P, R, O \rangle, \quad (1)$$

where $P = \{P_1, P_2, \ldots, P_N\}$ 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.

 doi:10.20944/preprints202603.0401.v1

Operations are further defined as $O_{ij} = \langle R_{ij}, OC_{ij} \rangle$ 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:

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

The terms GC_{ij} determine if the touch contact function or the feasibility constraint are applied and GT_{ij} are general translational statements [24].

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.

$$RTC = TC_1 \vee TC_2 \vee TC_3 \vee TC_4 \vee TC_5 \vee TC_6 \quad (3)$$

   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 TT_i, before to sum them:

$$TT_i = T_i^{B,F1} \wedge T_i^{S,F1} \quad (4)$$

$$RTT = TT_1 \vee TT_2 \vee TT_3 \vee TT_4 \vee TT_5 \vee TT_6 \quad (5)$$

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 turned into an MDP graph after making some assumptions that are assembly specific [25]. The conversion in MDP is presented in Section 4 for the chosen case study.

3.2. Synthetic generation of Assembly Sequences

In order to generalize the problem, we can assume that the assembly sequence can be represented by a graph displaying the assembly tasks as defined in (1). The level of detail of the graph, i.e., the precise definition of an elementary operation, is left intentionally to the discretion of the applicants. The assembly process is expressed as a Directed Acyclic Graph (DAG), where nodes are assembly operations (states) and edges correspond to feasibility or precedence constraints between components. Feasibility constraints are applied by removing the 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) [26]. An edge $(u, v) \in E$ represents a directed connection from u to v. In a DAG, for any vertex v, there isn't 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 $(u, v) \in E$, vertex u comes before v in the ordering. This represents a valid sequence of assembly operations and ensures that the process contains no cycles, preventing loops or deadlocks during execution.

It should be noted that the choice of DAG is not imposed by the problem but is a deliberate design choice, aimed at applying ARL to the system represented by the graph. As a matter of fact, DAG doesn't correspond directly to the assembly graph, generated by (1) incorporating all the feasible assembly sequences. DAG must respect additional ordering constraints that can be only

 doi:10.20944/preprints202603.0401.v1

obtained by duplicating every node that could be accessed in multiple orders. An automatic procedure can be used to convert a standard assembly graph into a DAG assembly sequence. Figure 2 explains the procedure. If both sequences $A \rightarrow B \rightarrow C$ and $A \rightarrow C \rightarrow B$ are admissible, the assembly task representation, non-ordered, could be the one of Figure 2 a). Both directions are admissible for moving through tasks B and C. If we force a sequence, either task B is executed during one step or task C, the other must be executed during next time step. Therefore, the DAG representation becomes as in Figure 2 b).

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

In order 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 preferentially drawn towards adjacent layers to reduce the chance of creating funnel-like structures. In Figure 3, an example of a random generated DAG is provided.

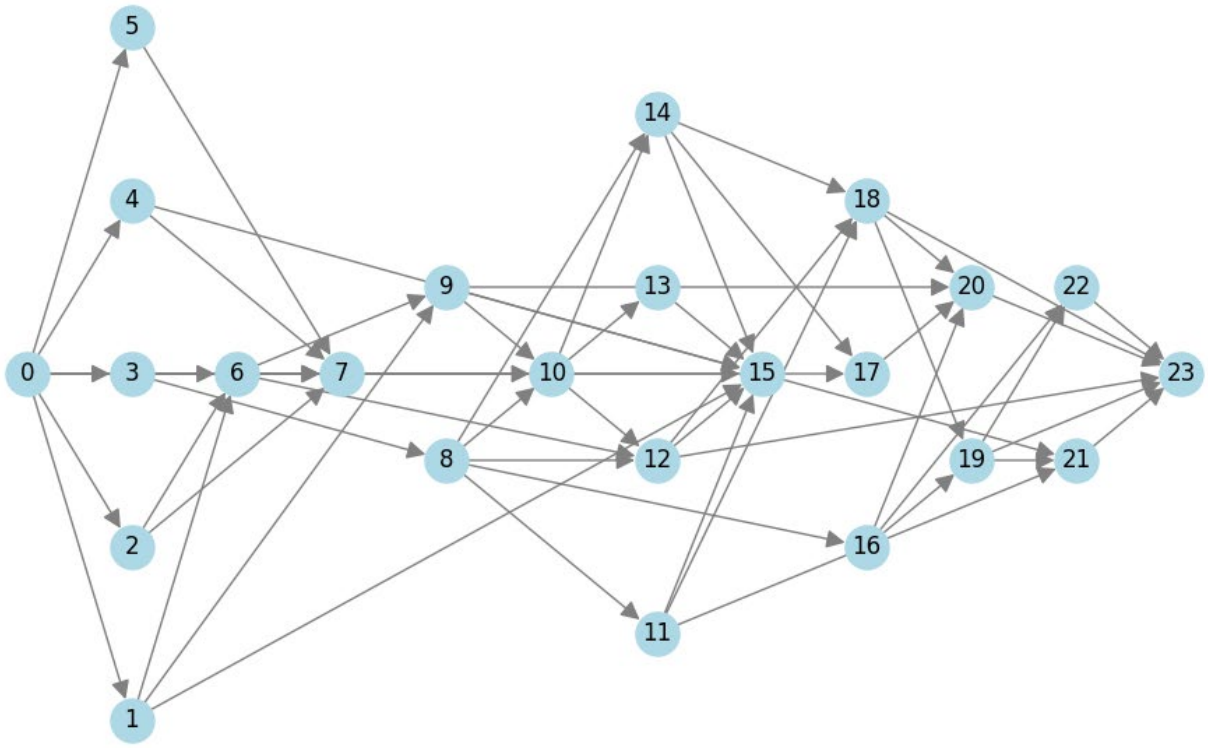

Figure 3. Random generated DAG for testing purposes with 10 inner layers, 10 nodes maximum per layer, 4 actions maximum per node.

3.3. Reinforcement Learning Models

 doi:10.20944/preprints202603.0401.v1

To endow the robot with the decision-making autonomy necessary for flexible collaboration, RL is employed. RL is a machine learning paradigm where an agent learns to make decisions by performing actions in an environment and receiving feedback in the form of numerical rewards or penalties [27]. 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 (S, A, P, R, γ), where [28]:

- 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 $R(S, A)$ 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 RL algorithm involves estimating the value function, specifically the Q-value, denoted as $Q(S, A)$. The Q-value represents the expected cumulative reward an agent can achieve starting from state S, taking action A, and following a specific policy thereafter. These values are updated iteratively to converge towards the optimal policy.

On-policy algorithms (like SARSA and PPO) learn the value of the policy being executed, 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 utilize **Proximal Policy Optimization** (PPO), a state-of-the-art on-policy gradient algorithm [29]. Unlike value-based methods that 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:

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

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," improving training stability.

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

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

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, the action performed yielded a better outcome than the average expectation 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 outcome than the average 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.

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 [30]. The original idea behind this study stems from the observation that humans

 doi:10.20944/preprints202603.0401.v1

and robots have very different ways of acting and that it is not possible to force humans to faithfully execute a certain strategy, especially when the chosen strategy does not lead to clear advantages. Therefore, the robot cannot rely on humans adhering to the plan. Instead, it must implement robust strategies (fail safe) 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 the human are modeled as two agents competing in a pseudo game, similar to [31].

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 Directed Acyclic Graph (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 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, 3 steps long, but if robot choses node 2, the human from node 2 could chose the path (1,2,6,7,8,9), 6 steps, that is the longest path in the graph. On the contrary 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. Therefore, the policy that includes node 3 can be considered robust even if non optimal.

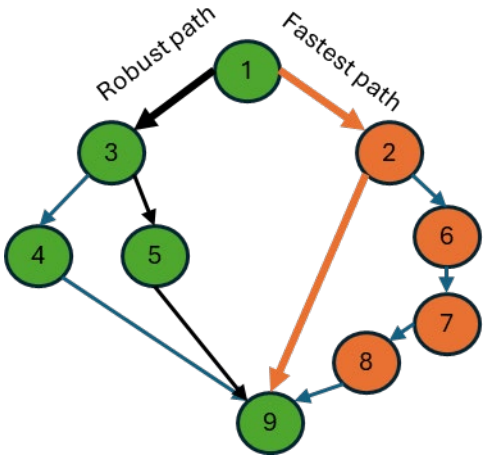

Figure 4. Example of robust against fast path.

On-Policy Learning in Adversarial Settings: In our adversarial framework, two agents (Robot and Human) compete. PPO is an on-policy algorithm, it requires data generated by the current policy to perform updates. In a multi-agent adversarial environment, the environment is typically non-stationary because the opponent is learning simultaneously. To address this, we employ an alternating training scheme. By fixing the opponent's policy while training the active agent, the environment becomes temporarily stationary. This allows the active agent to collect valid on-policy trajectories against the specific strategy currently deployed by the opponent, effectively optimizing a response to the current "goal" of the game.

3.4.1. Problem Formulation

 doi:10.20944/preprints202603.0401.v1

The State Space is composed by the nodes of the DAG representing assembly states. The Action Space is the selection 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.

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 will propose a robust policy to execute the assembly in a workplace where the humans are allowed to deviate from the optimal assembly sequence. Human policy is discarded at the end as it was used just to train the robot to operate in the worst practical conditions. If the actual human instead is truly collaborative, the robot can follow the fastest policy. It is not guaranteed that this fastest policy be the global optimal one. It could be a sub-optimal policy, acceptable in practical applications.

The procedure is formalized in the pseudo-code of Figure 5.

```
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.

 doi:10.20944/preprints202603.0401.v1

The pseudo-code has been used to prompt a Large Language Model (LLM), namely Google Gemini 3.1 Pro, to generate the executable training program. The implementation and training were conducted using a Python-based machine learning stack. The PPO reinforcement learning algorithms were deployed using the Stable-Baselines3 library, which leverages PyTorch as its underlying deep learning backend. To ensure seamless integration with the learning agents, the custom collaborative assembly environment was built following the Gymnasium API standard. Furthermore, NumPy was utilized for efficient numerical computations and matrix operations, while Cloudpickle handled the serialization and deserialization of the models and environment states.

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 & 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 ↔ 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: 5 checks.
Convergence Threshold: 0.5 (Maximum difference in mean reward over the last 5 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

 doi:10.20944/preprints202603.0401.v1

To validate the performance of the proposed ARL algorithm within the context of HRC and of the ABI trust model, a combination of approaches is used, including both quantitative efficiency metrics and robustness indicators [32].

- **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 1,000 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 introduce a laboratory case study, extracted by a longer industrial process. The case study involves the assembly of a turbomolecular vacuum pump (Figure 6), conducted at the collaborative robotics laboratory of the Politecnico di Torino. The process utilizes a collaborative cell featuring two UR3e cobots (named R1 and R2) and a human operator (named H) working in a shared workspace to improve efficiency (Figure 7). 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 the Bottom cap (BC). The assembly is then mounted onto an Envelope (EV) and completed with a Foreline flange (FOR). Fastening is handled using various screws, specifically VT1 (M3X8), VT2 (hex M4X10), VT3 (hex M5X20), and VT4 (hex M3X8). The process is suitable for a collaborative execution by human and cobots.

 doi:10.20944/preprints202603.0401.v1

Figure 6. Case study for the collaborative assembly process: a turbomolecular vacuum pump whose main components must be assembled by screwing.

Figure 7. Collaborative workstation equipped with ceiling-mounted robotic arms: the left is equipped with a gripper, the right with a screwdriver.

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 safely 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 8). 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, lifts and holds components steady during manual assembly phases.

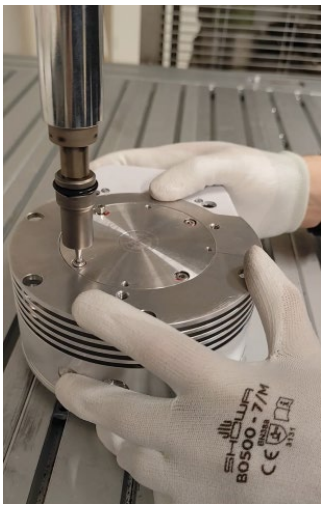

Figure 8. Collaborative operation: R1 is screwing while H holds the part in position.

The assembly sequence (Table 2) is composed of 5 main tasks and several sub-tasks. In task 1 the plastic case is screwed onto the main body by R1. In task 2 the bottom case is positioned onto the body and subsequently screwed in place by R1. In both tasks R2 handle the parts to be assembled. In task 3 R2 pick and place tools and parts. This is a totally automatic task without human collaboration.

 doi:10.20944/preprints202603.0401.v1

In task 4, H1 executes a manual screwing of the plastic case while R2, lifts and supports the body. In task 5, R2 positions the assembled body onto the envelope, and the human operator (H1) manually screws them together to complete the process.

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 2x 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 4x 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 2x 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 6x VT3 (3 Nm)	**H**
5.4	Final assembly	Position and screw FOR and VT4 (1.5 Nm)	**H**

The assembly plan is reported on the DAG of Figure 9. For sake of simplicity the tasks 3, 4 and 5 collapse in one final node. In this case study the human is expected to replace hex bit with cross bit on the screwdriver on R1. The expected assembly sequence is therefore the bottom path on the graph with increasing task numbers. It is likely that H forgets to change the bit and leaves the hex bit mounted and positions BC. It is the same as if task 21/22 were executed instead of 12. In conventional manufacturing, this would cause an exception stop, and the operator would lose some time to understand what the cause of stop is. Adopting the ARL strategy, the robot finds an alternative path on the graph, the upper line and executes the process alike.

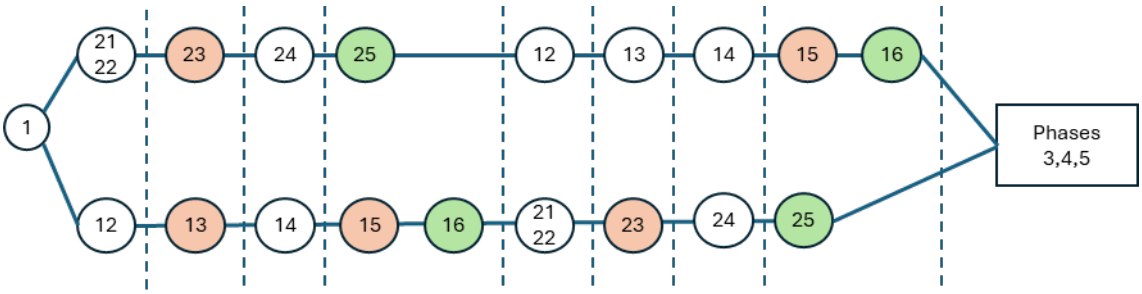

Figure 9. DAG for the vacuum pump assembly. White is used for operations assigned to H, orange to R2 and green to R1. Dash lines separate robot turns from human ones. The assembly sequence has been simplified as phases 3, 4 and 5 were not detailed.

The strategy proves effective, but the use of ARL in this example is clearly an overuse of force. It is necessary to test ARL on more complex work situations.

 doi:10.20944/preprints202603.0401.v1

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%

Because 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 5 steps, WCPL 10 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].

 doi:10.20944/preprints202603.0401.v1

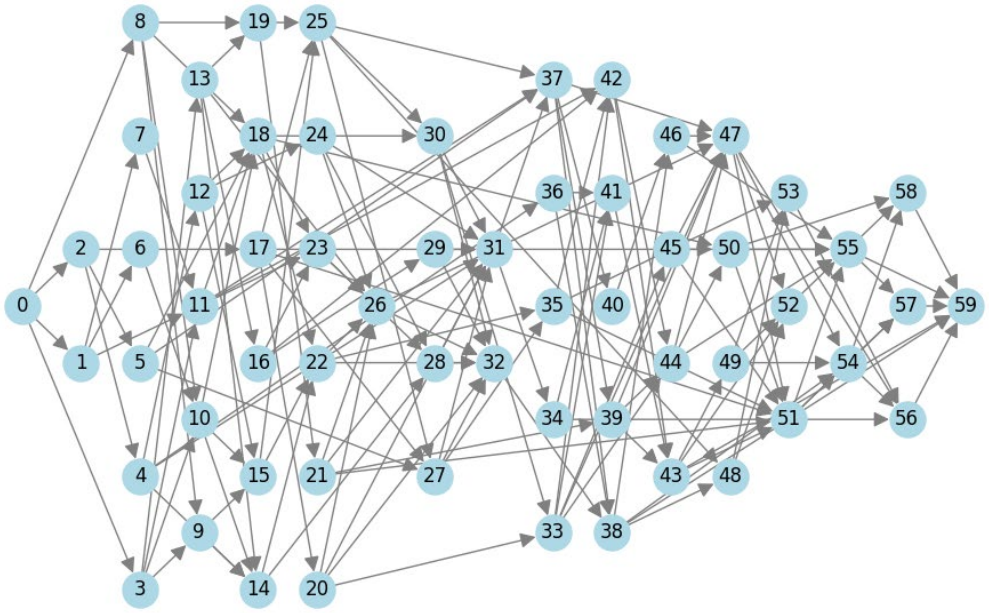

Figure 10. Random generated DAG with 15 inner layers, 10 nodes maximum per layer, 4 actions maximum per node.

The graph of Figure 11 tracks the cumulative rewards earned by the two competing agents, Robot and Human, over a training span of approximately 7,500 episodes. To ensure clarity amidst the raw data noise (the faint background lines), both trajectories are smoothed using a 50-episode Moving Average (MA 50). 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.

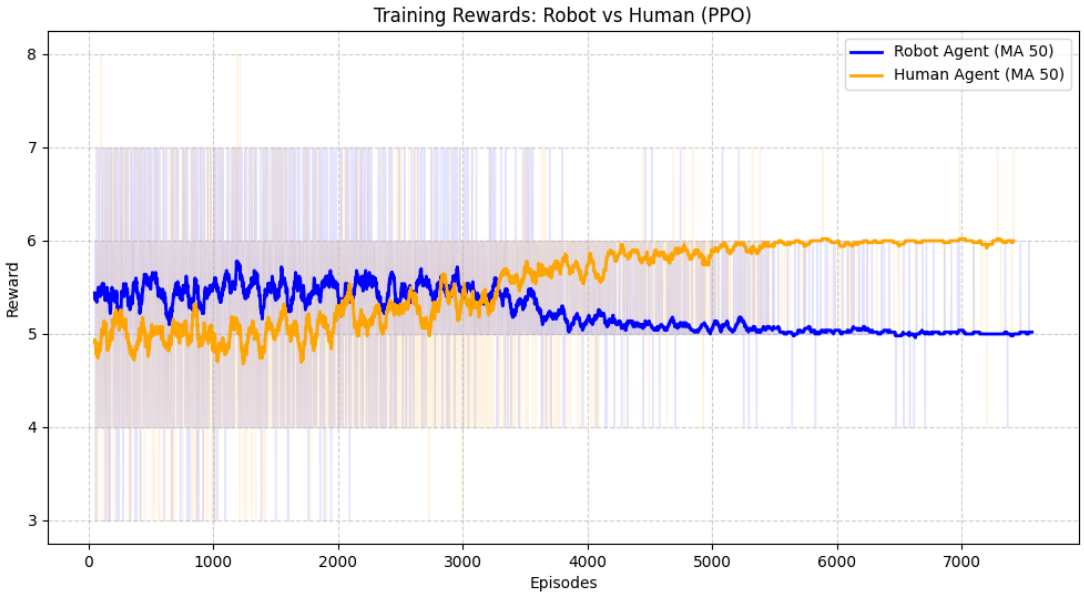

Figure 11. Training reward dynamics for ARL using PPO. The Human Agent (orange) eventually stabilizes at a higher reward plateau than the Robot Agent (blue).

 doi:10.20944/preprints202603.0401.v1

It should be noted that the human policy itself is not reported among the results because it is not used during the actual execution; it serves strictly as an adversarial tool during the training phase. To correctly interpret these results, it is necessary to clarify the peculiar characteristics of this model. Although frequently referred to as a "game" in reinforcement learning contexts, this framework does not represent a fair game. The random generation of the graph structure inherently has the potential to favor either the human's policy or the robot's policy. Consequently, for certain graph topologies, the training phase will predictably conclude with a higher accumulated reward for the human agent and conversely for others the robot always gains a better reward.

It is important to understand that a higher human reward does not signify that the robot has "lost" the scenario; rather, it simply indicates that the specific graph topology contains naturally longer traversal paths, which translates into a higher score for the delay-seeking human. The true robot's objective is not to "win" by achieving the highest score, but strictly to prevent the human from wasting excessive time on useless or inefficient operations. This dynamic accurately mirrors real-world manufacturing environments: some assembly processes are inherently faster and simpler than others, presenting fewer opportunities for errors or alternative choices. The primary metric of success is that the robot demonstrates the capability to drive the assembly to completion in every scenario without freezing or getting stuck, which remains a common challenge in industrial applications involving collaborative robots. Therefore, 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. The robot does not simply maximize its own reward locally; it adapts its behavior to safeguard the collaborative effort against the 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.

The shift from optimizing for pure speed to optimizing for resilience and adaptability directly addresses the "Flexibility" barrier in HRC. By ensuring the system is robust against the variable behavior of the human "adversary" (proxy for human variability), we create 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 heavily dictates the robot's chance of success. As reported in Figure 12, in Trial 2 , Trial 7 , and Trial 8, the resilience ratio is 97% or higher, meaning the optimal robot policy nearly always outperforms random behavior. Conversely, Trial 5 had a resilience of just 28.30%. This implies that certain graph topologies likely contain choke points or edge configurations that overwhelmingly favor the human adversary, stripping the robot of its ability to guarantee a significantly better-than-random path.

 doi:10.20944/preprints202603.0401.v1

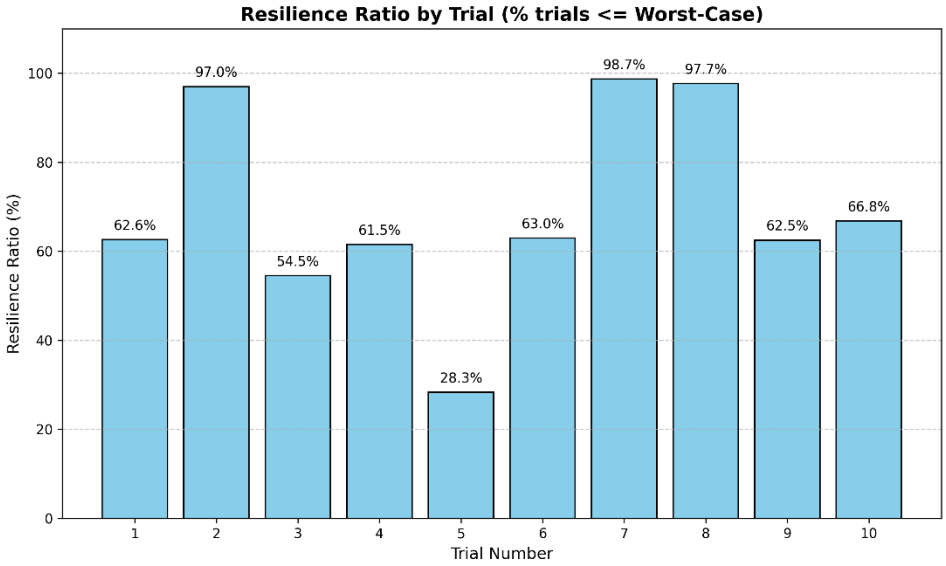

Figure 12. Resilience Ratio for the different trials.

The "Adversarial Tax": The difference between the theoretical shortest paths (ranging from 2 to 6 steps) and the robot's optimal policy (ranging from 5 to 17 steps) is substantial. This metric perfectly highlights the adversary's effectiveness; by acting optimally, the human forces the robot to take paths over 2.5 times longer on average than the shortest possible route (Figure 13). Obviously, this is the worst scenario, not the standard practice. The human coworker is expected to follow a collaborative policy most of the times.

Optimal vs. Random Expectations: It is interesting to note that the overall average of the optimal robot's worst-case policy (11.7 steps) is actually slightly higher than the overall average random length (10.73 steps). This happens because the worst-case metric measures a guaranteed upper bound against an optimal adversary. The random trials average out the human's moves, reflecting scenarios where the human sometimes makes choices that a smart robot can capitalize on to cross the graph much faster.

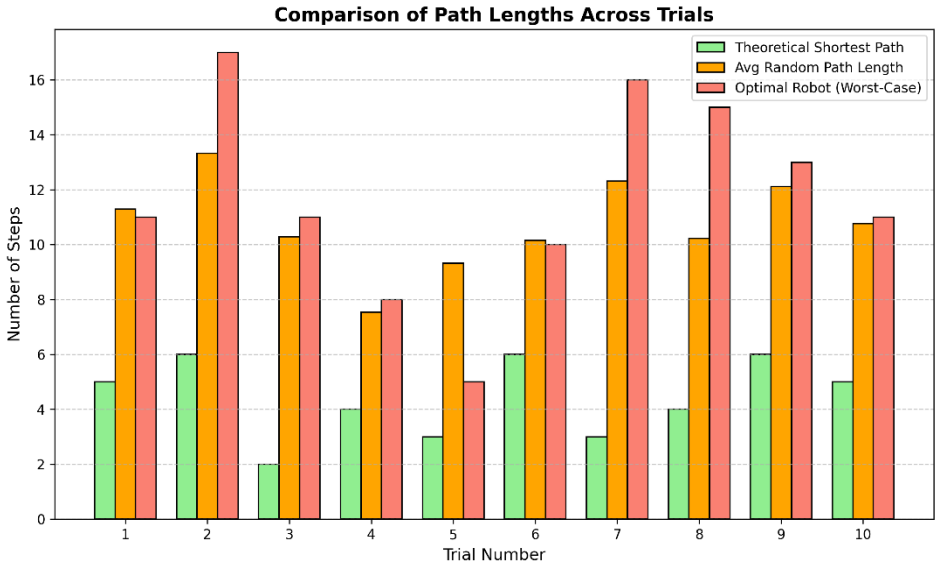

Figure 13. Comparison of path lengths: shortest path, average random, optimal policy.

5.3. Implications for Trust

 doi:10.20944/preprints202603.0401.v1

The results of the experimental analysis highlight a critical connection between control strategy and trust. In all 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 presents some limitations that must be acknowledged.

The conceptual linkage between the proposed ARL framework and the Ability and Benevolence dimensions of the ABI trust model provides a theoretically grounded rationale for enhanced human–robot trust. However, this relationship has not yet been empirically validated through human-subject experimentation. The present work relies on simulations and controlled laboratory experiments and does not incorporate validated trust questionnaires, behavioral indicators, or psychometric assessment tools. Therefore, although the framework establishes measurable performance guarantees and transparency mechanisms that are theoretically conducive to trust formation, direct experimental confirmation of increased human trust remains an open research direction.

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 suitable for demonstrating feasibility, the scenario 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 and randomly generated datasets. Although this enables controlled and systematic benchmarking, it does not entirely capture the variability and uncertainty typical of real operational conditions. Broader validation in high-complexity industrial settings is therefore necessary to further demonstrate scalability and practical robustness.

6. Conclusions

This study introduced an ARL framework designed to enhance trust in human-robot collaborative assembly. By modeling the assembly process as a MDP on a DAG and utilizing an alternating PPO training scheme, the robot learns a robust policy capable of mitigating the inherent unpredictability of human partners. The experimental results on synthetic data and on a case-study demonstrate the robot's ability to ensure task completion within a bounded timeframe, regardless of human deviations. It aligns with the Ability and Benevolence components of the ABI model of trust. While the ARL-trained robot does not necessarily achieve the absolute minimum TCT under ideal conditions, it provides a reliable and resilient alternative to conventional, rigid assembly optimization algorithms. The effectiveness of the ARL strategy is heavily dependent 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 could result in path lengths that can be longer than the theoretical shortest route, a necessary cost for avoiding work interruptions due to lack of agreement between human and robot. 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 creates a more resilient collaborative environment where human operators can feel supported rather than constrained.

 doi:10.20944/preprints202603.0401.v1

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: There are no additional data.

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
RQ	Research Question
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

References

1. Wang, L., et al. Assembly process planning and its future in collaborative manufacturing: a review. *The International Journal of Advanced Manufacturing Technology* **2009**, 41.1: 132-144. https://doi.org/10.1007/s00170-008-1458-9
2. Del Real Torres, A., et al. A review of deep reinforcement learning approaches for smart manufacturing in industry 4.0 and 5.0 framework. *Applied Sciences* **2022**, 12.23: 12377. https://doi.org/10.3390/app122312377
3. Dieber, B.; Schlotzhauer, A.; Brandstötter, M. Safety and Security–Success factors of sensitive robotic technologies. *Elektrotechnik und Informationstechnik* **2017**, *134*, 299-303.
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.
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. *Occupational and environmental safety and health* **2019**, 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 Idea Square Journal of Experimental Innovation* **2022**.
7. Haas, M.; Mortensen, M. The secrets of great teamwork. *Harvard business review* **2016**, *94*, 70-76.
8. Mayer, R. C.; Davis, J. H.; Schoorman, F. D. An integrative model of organizational trust. *Academy of management review* **1995**, 20.3, 709-734.

 doi:10.20944/preprints202603.0401.v1

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 **2021**. p. 85-121.
10. Maderna, R.; Pozzi, M.; Zanchettin, A. M.; Rocco, P.; Prattichizzo, D. Flexible scheduling and tactile communication for human–robot collaboration. *Robotics and Computer-Integrated Manufacturing* **2022**, *73*, 102233.
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. *Industrial Robot: the international journal of robotics research and application* **2022**, *49*, 226-239.
12. Masehian, E.; Ghandi, S. Assembly sequence and path planning for monotone and nonmonotone assemblies with rigid and flexible parts. *Robotics and Computer-Integrated Manufacturing* **2021**, 72: 102180.
13. Lazzerini, B.; Marcelloni, F. A genetic algorithm for generating optimal assembly plans. *Artificial Intelligence in Engineering* **2000**, 14.4: 319-329.
14. Li, M., et al. An improved discrete particle swarm optimization algorithm for high-speed trains assembly sequence planning. *Assembly Automation* **2013**, 33.4: 360-373.
15. Han, Z.; Wang, Y.; Tian, D. Ant colony optimization for assembly sequence planning based on parameters optimization. *Frontiers of Mechanical Engineering* **2021**, 16.2: 393-409.
16. Karthik, G.; Deb, S. A methodology for assembly sequence optimization by hybrid cuckoo-search genetic algorithm. *Journal of Advanced Manufacturing Systems* **2018**, 17.01: 47-59.
17. Malek, N.; Peng, Q. Reinforcement learning for self-adaptive genetic algorithm in assembly sequence planning. *The International Journal of Advanced Manufacturing Technology* **2025**, 1-20.
18. Suszyński, M.; Peta, K. Assembly sequence planning using artificial neural networks for mechanical parts based on selected criteria. *Applied Sciences* **2021**, 11.21: 10414.
19. Masehian, E.; Ghandi, S.. ASPPR: A new assembly sequence and path planner/replanner for monotone and nonmonotone assembly planning. *Computer-Aided Design* **2020**, 123: 102828.
20. Liu, J., et al. Integrating planning and deep reinforcement learning via automatic induction of task substructures. In The Twelfth International Conference on Learning Representations. **2024**.
21. 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**, 12.2: 103.
22. 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.
23. Gottipolu, R.B.; Ghosh, K.. A simplified and efficient representation for evaluation and selection of assembly sequences. *Computers in Industry* **2003**, 50(3), pp.251-264.
24. Deepak, B., et al. Assembly sequence planning using soft computing methods: a review. *Proceedings of the Institution of Mechanical Engineers, Part E: Journal of Process Mechanical Engineering* **2019**, 233.3: 653-683.
25. Aliev, K.; Antonelli, D.; Bruno, G. Task-based programming and sequence planning for human-robot collaborative assembly. *IFAC-PapersOnLine* **2019**, *52*, 1638-1643.
26. Heath, L.; Pemmaraju, S.; Trenk, A. Directed Acyclic Graphs. *Planar Graphs* **1992**, 9: 5.
27. Sutton, R., et al. Reinforcement learning. *Journal of Cognitive Neuroscience* **1999**, 11.1: 126-134.
28. Puterman, M. L. Markov decision processes. *Handbooks in operations research and management science* **1990**, *2*, 331-434.
29. Schulman, J., et al. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* **2017**.
30. Antonelli, D.; Zeng, Q.; Aliev, K.; Liu, X. Robust assembly sequence generation in a Human-Robot Collaborative workcell by reinforcement learning. *FME Transactions* **2021**, *49*, 851-858.
31. Zhao, H., et al. Adversarial Reinforcement Learning for Enhanced Decision-Making of Evacuation Guidance Robots in Intelligent Fire Scenarios. *IEEE Transactions on Computational Social Systems* **2024**, 12.5: 2030-2046.
32. Pinto, L., et al. Robust adversarial reinforcement learning. In: *International conference on machine learning*. PMLR **2017**. p. 2817-2826.

 doi:10.20944/preprints202603.0401.v1

disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.