



POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Proactive Human Avoidance in Collaborative Logistics: An RL-Based Dynamic Scheduling Framework

TESI DI LAUREA MAGISTRALE IN
AUTOMATION AND CONTROL ENGINEERING

Author: **Lorenzo Marzi**

Student ID: 11015873

Advisor: Prof. Paolo Rocco

Co-advisors: Christian Cella, Marco Faroni

Academic Year: 2024-25

Abstract

Human-Robot Collaboration (HRC) in modern manufacturing aims to combine human cognitive flexibility with robotic precision. However, standard reactive safety protocols, such as Speed and Separation Monitoring (SSM), frequently degrade system throughput by forcing protective stops or decelerations whenever a human enters a shared zone. To mitigate these inefficiencies, this thesis proposes a dynamic task-scheduling framework based on Deep Reinforcement Learning (DRL) to optimize real-time temporal and spatial coordination.

The shared workstation is formalized as a Markov Decision Process (MDP), where the human operator is modeled as an autonomous agent governed by a probabilistic Finite State Machine. A robotic orchestrator is trained using a Masked Proximal Policy Optimization algorithm within a high-fidelity digital twin developed in Siemens Tecnomatix Process Simulate. A dynamic action-masking mechanism enforces strict production and physical constraints, significantly reducing the exploration space and accelerating convergence.

The learned policy optimizes a dual objective: minimizing cycle time while proactively reducing spatial interference with the human operator. Compared to a reactive greedy baseline that prioritizes execution speed, the proposed approach substantially decreases SSM induced safety halts. Although the reactive strategy achieves a slightly shorter nominal cycle time, it incurs frequent disruptive stops. In contrast, the RL-based orchestrator learns proactive coordination strategies, including *strategic waiting* and *Opposite Side Tasking*, enabling continuous parallel operations and stable zone separation.

Simulation results demonstrate that the proposed framework preserves high operational throughput while virtually eliminating mid-task safety interruptions. By anticipating human motion rather than reacting to proximity violations, the orchestrator ensures smoother process execution and more predictable human-robot interaction, supporting efficient and fluid collaboration.

Keywords: Human-Robot Collaboration (HRC), Reinforcement Learning, Dynamic Task Scheduling, Proximal Policy Optimization (PPO), Collaborative Logistics

Abstract in lingua italiana

La Collaborazione Uomo-Robot (HRC) unisce la flessibilità cognitiva umana alla precisione robotica. Tuttavia, protocolli di sicurezza reattivi come lo Speed and Separation Monitoring (SSM) degradano frequentemente la produttività, forzando arresti o decelerazioni all'avvicinarsi dell'operatore. Per mitigare tali inefficienze, questa tesi propone un framework di schedulazione dinamica basato sul Deep Reinforcement Learning (DRL) per ottimizzare il coordinamento spaziale e temporale in tempo reale.

La postazione condivisa è formalizzata come Processo Decisionale di Markov (MDP), dove l'umano è modellato come agente autonomo governato da una Macchina a Stati Finiti probabilistica. Un orchestratore robotico è addestrato tramite l'algoritmo Masked Proximal Policy Optimization all'interno di un digital twin (Siemens Tecnomatix Process Simulate). Un meccanismo di action-masking dinamico impone rigidi vincoli fisici e produttivi, riducendo drasticamente lo spazio di esplorazione e accelerando la convergenza.

La policy appresa ottimizza un duplice obiettivo: minimizzare il tempo di ciclo e ridurre proattivamente le interferenze con l'operatore. Rispetto a una baseline reattiva basata sulla velocità, l'approccio proposto abbatte gli arresti di sicurezza indotti dall'SSM. Pur ottenendo un tempo di ciclo nominalmente inferiore, la strategia reattiva subisce continue interruzioni. Al contrario, l'orchestratore RL adotta strategie proattive, tra cui l'attesa strategica (*Wait*) e l'*Opposite Side Tasking*, garantendo operazioni parallele e una separazione stabile delle zone.

Le simulazioni dimostrano che il framework preserva un'elevata produttività, eliminando virtualmente gli arresti di sicurezza durante le attività. Anticipando i movimenti umani anziché reagire alle violazioni di prossimità, l'orchestratore assicura un'esecuzione più fluida e un'interazione prevedibile, supportando una collaborazione efficiente.

Parole chiave: Collaborazione Uomo-Robot, Apprendimento per Rinforzo, Schedulazione Dinamica dei Task, Proximal Policy Optimization, Logistica Collaborativa.

Contents

Abstract	i
Abstract in lingua italiana	iii
Contents	v
1 Introduction	1
1.1 Thesis Purpose and Methodology	2
1.2 Thesis Achievements	3
1.3 Thesis Organization	3
2 State of the art	5
2.1 Cobots & HRC	6
2.1.1 From industrial robotics to HRC	6
2.1.2 Classification of Human-Robot Relationships	8
2.2 Safety in Human-Robot Collaboration	9
2.3 Task Scheduling in HRC	11
2.3.1 Definition and Problem Statement	11
2.3.2 Static Task Scheduling (Offline)	12
2.3.3 Dynamic Task Scheduling (Online)	13
2.3.4 Learning based Task Scheduling	14
2.4 Contribution of this work	16
3 Theoretical Background	17
3.1 Reinforcement Learning	17
3.1.1 Core Components of RL	18
3.1.2 Proximal Policy Optimization (PPO)	21
4 System Description and Problem Formulation	25
4.1 The Operational Scenario	26

4.1.1	Task Definition and Constraints	30
4.2	The Production Cycle	32
4.3	Problem Statement and Objective	33
4.4	High Level System Architecture	34
5	Methodology and Implementation	37
5.1	Formal Modeling of the Collaborative System	37
5.1.1	Human Modeling	37
5.1.2	MDP Formulation	40
5.2	Implementation and Training Framework	48
5.2.1	Algorithm and Policy Implementation	48
5.2.2	The Simulation Environment (Tecnomatix)	48
6	Results and Discussion	51
6.1	Training Settings	51
6.1.1	Learning Curves	52
6.2	Policy Evaluation and Qualitative Analysis of Emergent Behaviors	53
6.2.1	End-to-End Task Execution	53
6.2.2	Cycle Time and Efficiency	54
6.2.3	Proactive vs. Reactive: A Comparative Analysis	55
6.2.4	Human Avoidance and Spatial Coordination	56
7	Conclusion and Future Work	61
7.1	Conclusion	61
7.2	Limitations and Future Work	61
7.2.1	Human Intent Recognition and Goal Inference	62
7.2.2	Experimental Validation and Sim-to-Real Transfer	62
	Bibliography	63
	List of Figures	67
	List of Tables	69

1 | Introduction

Industrial automation is experiencing a profound shift, moving away from fully isolated robotic systems toward Human-Robot Collaboration (HRC). Within the framework of Industry 4.0 and the growing human-centered perspective of Industry 5.0, the objective extends beyond simple automation to integrating human cognitive adaptability with the accuracy and consistency of robotic technologies.

To enable this synergy, a specific class of manipulators known as collaborative robots, or cobots, has been introduced. Unlike traditional industrial robots that operate blindly behind physical fences, cobots are designed to share the workspace safely with human operators. This collaborative approach is particularly valuable in situations where complete automation is either too costly or technically impractical, such as in highly customized production lines or environments requiring frequent changeovers. Furthermore, HRC significantly contributes to operator well-being. By reallocating physically demanding, monotonous, or non-ergonomic tasks to the robot, the system mitigates the risk of work-related injuries while allowing the human to focus on high-value, complex problem-solving activities.

However, introducing cobots into shared workspaces creates significant safety and efficiency challenges. Traditional safety standards, such as Speed and Separation Monitoring (SSM), ensure human safety by dynamically reducing the robot's speed or forcing a complete protective stop when the separation distance falls below a critical threshold. While effective for safety, these purely reactive strategies frequently result in prolonged idle times, hence severely reducing the overall system throughput.

Consequently, a major portion of current research explores HRC deployment strategies aimed at maximizing both the time efficiency of cobot operations and the ergonomic well-being of the human operator. While most of this research focuses on assembly scenarios, where the primary challenge is deciding *who* performs which task (task allocation), collaborative logistics represents another critical and highly effective use case. In logistics contexts, such as packaging and palletizing, task allocation is frequently predetermined based on the distinct physical capabilities of the agents. For instance, robots excel at

the repetitive manipulation of heavy payloads without fatigue, whereas human operators can easily navigate complex, dynamically changing environments while also providing the manual dexterity needed for more flexible operations. Therefore, the primary challenge shifts from task allocation to determining the optimal temporal sequencing and spatial coordination of these tasks in real time.

1.1. Thesis Purpose and Methodology

Within the context of collaborative logistics, this thesis explores how to maximize operational fluidity in a shared workstation. The primary purpose is to develop an intelligent robotic orchestrator capable of operating in a complex environment characterized by the stochasticity and autonomy of the human operator. By continuously utilizing up to date state information, the goal is to dynamically schedule the robot's actions to reduce overall cycle times while proactively avoiding workspace conflicts with the human partner.

To contextualize this objective, reference is made to the experimental setup illustrated in Figure 1.1. The environment features a human operator and a robotic agent working collaboratively to achieve a common palletizing goal. The robot is tasked with automated pick-and-place operations, moving components from the assembly line into crates for palletization. Simultaneously, the human operator is responsible for replenishing components on the line, providing overall supervision, and performing manual packaging tasks that require high dexterity.

Consequently, the central challenge is that the robot's actions must dynamically adapt to the human's immediate goals. For instance, if the human intends to perform an operation in a specific zone of the workstation, the robot is expected to proactively select a task in a non conflicting, physically distant zone. Furthermore, the orchestrator must not only ensure spatial avoidance but also maximize operational efficiency. This includes learning complex, time saving behaviors, such as clustering tasks that require the same end-effector to minimize unnecessary and time consuming tool changes.

Therefore, the goal of this thesis is to tackle this highly dynamic problem using a learning-based method, specifically Reinforcement Learning (RL). Given the unpredictable, stochastic nature of human decision-making, RL is utilized to naturally learn adaptive scheduling behaviors through continuous interaction with a simulated environment.

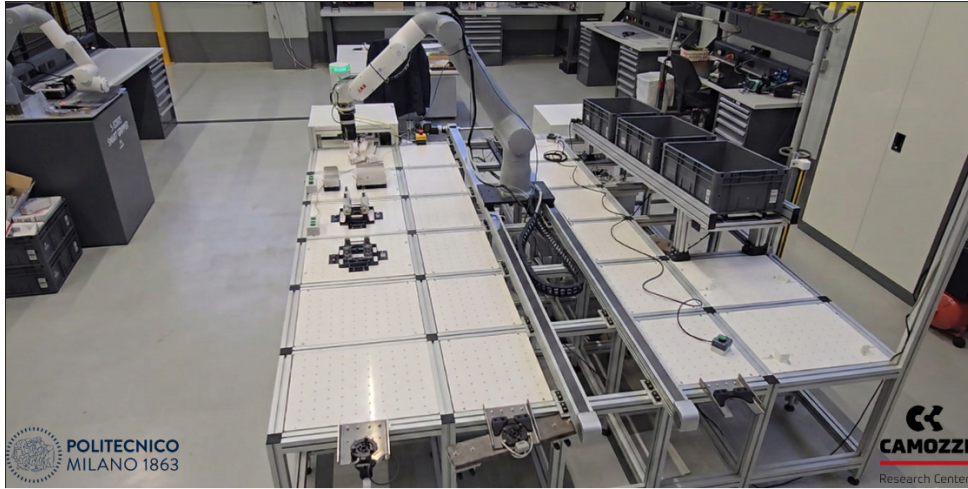


Figure 1.1: The collaborative logistics environment realized in the joint laboratory between Camozzi Automation S.p.A. and Politecnico di Milano.

1.2. Thesis Achievements

The core contributions of this thesis can be summarized as follows:

1. **Digital Twin-Driven MDP Framework:** This research formalizes the collaborative logistics cell as a Markov Decision Process (MDP), providing a clear mathematical structure for the scheduling problem. The implementation utilizes a custom software bridge to synchronize a Python based Reinforcement Learning suite with a high-fidelity Siemens Tecnomatix Process Simulate digital twin. This integrated environment enables the agent to learn complex coordination behaviors within a high-fidelity virtual replica of the Camozzi laboratory.
2. **Dual-Objective Orchestrator for Process Fluidity:** The thesis developed an intelligent orchestrator that maximizes system throughput while proactively minimizing spatial interference. Acknowledging that the workstation's safety is already guaranteed by underlying SSM protocols, the orchestrator's true achievement is preventing disruptive mid-air safety halts. It maintains a continuous, fluid, and intuitive workflow for the human operator through two learned spatial coordination strategies: *Opposite Side Tasking* and a strategic *Wait* action.

1.3. Thesis Organization

The remainder of this thesis is structured to progressively introduce the context, theoretical foundations, and methodological developments underpinning this approach:

- **Chapter 2: State of the Art** reviews the core concepts of collaborative robotics, the governing safety standards (such as ISO/TS 15066), and existing approaches to task scheduling in HRC, highlighting the transition from static to dynamic and learning-based methods.
- **Chapter 3: Theoretical Background** establishes the mathematical foundations of the study, detailing the Markov Decision Process (MDP) framework, the principles of Reinforcement Learning, and the specific mechanics of the Proximal Policy Optimization (PPO) algorithm.
- **Chapter 4: System Description and Problem Formulation** introduces the operational scenario, defining the layout of the collaborative logistics workstation, the atomic tasks of both agents, the physical constraints, and the formal optimization problem.
- **Chapter 5: Methodology and Implementation** details the practical development of the solution. It covers the probabilistic modeling of the human operator via a Finite State Machine, the exact formulation of the MDP (State, Action, Reward, and dynamic Action Masking), and the integration of the Python-based RL algorithm with the Siemens Tecnomatix high-fidelity simulation environment.
- **Chapter 6: Results and Discussion** presents the findings. It analyzes the training convergence of the Masked PPO agent and provides a qualitative evaluation of the resulting human-robot interactions, specifically validating the system's ability to maintain efficiency through proactive collision avoidance.
- **Chapter 7: Conclusion and Future Work** synthesizes the core contributions of the RL orchestrator, critiques the reliance on ground-truth intent data within the digital twin, ultimately delineating necessary steps for future experimental deployment.

2 | State of the art

The landscape of industrial automation is undergoing a fundamental transformation, driven by the transition from total robotic isolation towards Human-Robot Collaboration (HRC). In the paradigm of Industry 4.0 and the emerging human centric focus of Industry 5.0, the goal is no longer solely to automate processes, but to combine the cognitive flexibility of human operators with the precision and repeatability of robotic systems.

To contextualize this shift, this chapter reviews the frameworks, safety standards, and algorithms that govern modern HRC. The structure is organized as follows:

- Section 2.1 introduces the core concepts of collaborative robotics, classifying the levels of human-robot interaction and tracing the evolution from traditional industrial robots to cobots.
- Section 2.2 examines the constraints of safety in shared workspaces, reviewing the governing ISO standards and the strategies used to implement them.
- Section 2.3 addresses the decision making core of HRC: task scheduling. It surveys conventional static and dynamic approaches before focusing on learning based methods, specifically Reinforcement Learning.
- Finally, Section 2.4 details the contributions of this thesis, highlighting how the proposed solution adapts these concepts from assembly scenarios to the domain of collaborative logistics.

2.1. Cobots & HRC

2.1.1. From industrial robotics to HRC

Human-Robot Collaboration (HRC) represents a paradigm shift in industrial robotics, transitioning from segregated automation to a shared workspace where humans and robots operate in close proximity. The primary objective is to achieve shared production goals by combining the complementary strengths of both agents. Specifically, HRC leverages the cognitive flexibility and problem solving skills of the human operator along with the high precision, repeatability, and physical endurance of the robot [12].

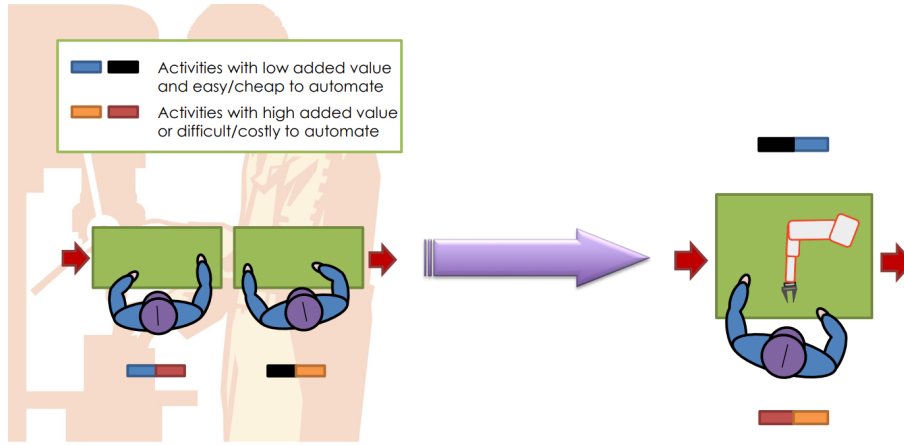


Figure 2.1: The diagram illustrates the reallocation of tasks: the robot executes activities with low added value or those easily automated (blue/black blocks), allowing the operator to focus on high value, complex tasks (orange/red blocks).

To enable Human-Robot Collaboration, a specific class of manipulators known as collaborative robots or cobots is used. To ensure effective interaction, these systems require specific capabilities: *sensing* (perceiving both the environment and the human), *adaptation* (modifying behavior based on the state of the agents), and *communication* (ensuring mutual understanding) [16].

It is important to emphasize that cobots are not intended to replace standard industrial robotics, but rather to complement them by extending the scope of automation. HRC specifically targets processes where full automation is technically prohibitive or economically unjustified, or where high flexibility is required such as in scenarios characterized by small batch sizes (high mix, low volume) and high customization requirements [12, 27].

Furthermore, HRC contributes to operator well being. By reallocating physically demanding, repetitive, or non ergonomic tasks to the robot, the system mitigates the risk of work

Table 2.1: Collaborative robots vs traditional robots (extension of [6])

Collaborative robots	Traditional robots
1. Do not require relevant prior knowledge to be programmed and installed;	1. Require experience in robotics and automation;
2. Integrate external force sensors (after the first series);	2. Blind and unaware of what surrounds them;
3. Do not require fences, occupying limited space;	3. Require fences, becoming cumbersome;
4. Focused on safety, flexibility and ease of use;	4. Focused on performance, accuracy and repetitiveness;
5. Capable of carrying out tasks exactly how they are performed by the operator;	5. Usually, weigh more than 50 kg;
6. Usually, weigh less than 30 kg;	6. Programmed for a specific task;
7. Can be easily integrated into the production system and can be “instructed”;	7. Require relevant integration efforts and dedicated equipment, including fences;

related injuries while enhancing productivity [16, 27].

Specific examples of industrial applications where cobots are employed are reported in [16]. Some of the use cases include:

- **Assembly:** This process involves the sequential aggregation of components to form a complex final product. It is typically characterized by high product variability, the need for dexterous grasping, and frequent changeovers. Due to these requirements, human operators have traditionally been the preferred choice over classical industrial robots. However, introducing HRC allows for the reallocation of non ergonomic and repetitive sub-tasks, such as screwing or tightening to the cobot, which can apply precise torques and forces, while the human usually performs complementary tasks like part positioning, nuts insertion.
- **Material Handling:** Applications such as palletizing and packaging often expose human workers to significant fatigue and long term health complaints due to heavy lifting. While traditional automation could solve this, it lacks flexibility and requires large, fenced workspaces. Cobots offer a viable alternative where higher flexibility is required (e.g., frequent workstation reallocation) or where space is limited. A typical collaborative workflow involves the operator preparing the pack while the cobot handles the heavy lifting to stack the pallet.
- **Finishing:** Finishing processes (e.g., polishing, sanding, deburring) modify the

surface of a manufactured part to achieve a desired appearance or durability. These activities are often physically exhausting and repetitive, significantly increasing the risk of injury and occupational health issues. The introduction of collaborative robots opens new opportunities for symbiotic approaches, where the robot handles the strenuous force application while the human provides the sensory feedback and quality assessment required for the required finish.

2.1.2. Classification of Human-Robot Relationships

The relationship between humans and robots is complex, and the terminology describing these interactions is often subject to ambiguity. To establish a clear framework, it is necessary to first analyze the interaction through distinct analytical perspectives. As summarized by Wang et al. [28], the nature of the relationship can be characterized by five key variables:

1. **Workspace:** Indicates whether the human and the robot operate within a shared physical environment, without physical or virtual fences for separation.
2. **Direct Contact:** Indicates whether physical contact between the robot and the human is expected or allowed during operations.
3. **Shared Working Task:** Represents whether the human and the robot work towards a common operational objective or goal.
4. **Simultaneous Process:** Indicates that the human and the robot are active at the same time, regardless of whether they are working on the same task.
5. **Sequential Process:** Represents a workflow where operations do not overlap temporally; actions are arranged one after another on the time scale.

Based on the combination of these variables, one can identify four distinct classes of human-robot relationships, ordered by increasing levels of interaction [28]:

Human-Robot Coexistence: This represents the most basic level of interaction. The industrial robot and the human share a general physical environment but operate in separate, non overlapping workspaces. There is no direct contact, and while they may exchange a work object, their respective tasks are performed independently and in parallel (simultaneously) without interference.

Human-Robot Interaction (HRI): In this scenario, the human and robot share a working environment but perform different tasks. Unlike coexistence, the workflow is typically sequential rather than parallel. Interaction inherently requires commu-

nication (cognitive level) where one party may guide or control the other, but direct physical contact is not required and usually avoided.

Human-Robot Cooperation: Cooperation involves agents (human and robot) retaining their own autonomy while working towards a common goal, objective, or utility. This relationship allows the system to harness the specific strengths of both agents. A shared workspace is required, but there is typically no direct physical contact. Crucially, the agents share resources but complete their respective sub tasks in a sequential order.

Human-Robot Collaboration: Collaboration represents the highest level of synergy. It is defined as a joint activity in a shared, symbiotic working environment with the definite objective of accomplishing a set of tasks together. It is characterized by coordinated, simultaneous activity where direct physical contact is allowed and often necessary. Collaboration implies a focused, goal oriented partnership where capabilities and resources are blended in real time.

The distinguishing features of these categories are summarized in Table 2.2.

Table 2.2: Features of different human-robot relationships (Adapted from Wang et al. [28]).

Feature	Coexistence	Interaction	Cooperation	Collaboration
Shared Workspace	✓	✓	✓	✓
Direct Contact	–	–	–	✓
Shared Working Task	–	–	✓	✓
Shared Resource	–	–	✓	✓
Simultaneous Process	✓	–	–	✓
Sequential Process	–	✓	✓	–

2.2. Safety in Human-Robot Collaboration

Collaborative robots introduce novel challenges that do not exist in traditional industrial robotics. In standard automation, safety is typically ensured by relying on safeguards and physical fences to impose a strict separation between the robot and the operator. In such cells, safety sensors (e.g., light curtains) are designed to detect human entry and immediately shut down the plant.

In contrast, HRC is intended to facilitate a shared workspace where humans and robots work in close proximity. Consequently, standard isolation measures are insufficient, and specific safety standards must be applied. The primary reference is ISO/TS 15066 [9] [14],

which defines four primary modes of collaborative operation:

1. **Safety Rated Monitored Stop (SRMS):** In this mode, the robot and human do not operate simultaneously in the shared workspace. The operator is permitted to enter the workspace only when the robot is in a monitored stop status. Safety is typically enforced using external sensors such as cameras, laser scanners, or pressure mats.
2. **Hand Guiding (HG):** This mode allows the operator to directly manipulate or move the robot using a manual guidance device. It acts as an augmentation of SRMS, where the robot is active only while the human is within a protective range and in direct control.
3. **Speed and Separation Monitoring (SSM):** This mode allows the human and robot to move simultaneously within the collaborative workspace. Safety is ensured by maintaining a minimum protective distance, monitored by tracking devices. The robot's control system dynamically adjusts its operating speed in proportion to its distance from the human, stopping completely if the separation falls below a critical threshold.
4. **Power and Force Limiting (PFL):** This mode enables high level collaboration (incidental contact). Safety is achieved by ensuring that, in the event of a collision, the contact force and energy transfer remain below specific bio mechanical thresholds defined to prevent injury. This is realized through inherent robot design (passive compliance) or active control measures (torque/force limitation).

While these standards define the requirements, Jain et al. [10] note that there is a lack of detailed implementation solutions specifying how to apply them to practical, dynamic industrial settings. Consequently, numerous implementation strategies have been developed; in [12], these approaches are classified into two main categories: *pre-collision* and *post-collision*.

Pre-collision Strategies

Pre-collision strategies aim to prevent harmful contact entirely by detecting or predicting the behaviors of the robot and the human in advance. These approaches rely on sensing and perception (e.g., vision systems) to establish awareness of the environment, enabling the effective adoption of SRMS and SSM modes.

Post-collision Strategies

Post-collision safety strategies do not prevent contact but rather mitigate its consequences. The essence of these strategies is to limit the contact force and energy exchange within safety thresholds to avoid injury to humans and damage to robots. Such strategies align with the PFL safety mode and do not typically require safety level sensing devices to monitor separation distance. Instead, passive compliance or active control algorithms are used to ensure safety during interaction.

2.3. Task Scheduling in HRC

2.3.1. Definition and Problem Statement

Scheduling is a decision making process fundamental to many manufacturing and service industries. As defined in [20], it deals with the allocation of resources to tasks over given time periods, with the goal of optimizing one or more objectives (e.g., makespan, cost).

In the specific context of Human-Robot Collaboration (HRC), the available resources are the human operator and the collaborative robot. As highlighted in Table 2.3, these agents have different strengths and weaknesses. When combining these heterogeneous capabilities with the strict safety requirements inherent to a shared workspace, the general scheduling problem can be seen as three interconnected sub problems [27]:

1. **Task Allocation (Who):** Decides the assignment of tasks based on agent capabilities. It determines which operations are best suited for the human, which for the robot, and which should be collaborative, considering factors such as execution time, precision, and ergonomics.
2. **Task Sequencing (When):** Defines the optimal chronological order of operations. This step arranges the allocated tasks into a feasible sequence that respects precedence constraints and minimizes idle time.
3. **Coordination (How):** Manages the real time interaction between agents. It addresses the delays and obstacles introduced by the stochastic nature of human behavior, ensuring synchronization and collision avoidance during execution.

Petzoldt et al. [19] provide an overview of the state of the art in assembly task allocation, categorizing solutions into two main families: *Static (Offline)* and *Dynamic (Online)* approaches. Additionally, we introduce task scheduling solutions based on Reinforcement Learning (RL). While these methods could fundamentally be classified as dynamic, we

discuss them in a separate section to address recent advancements and specific characteristics not covered by Petzoldt et al.

Table 2.3: Comparison of strengths and weaknesses between Humans and Collaborative Robots.

	Humans	Collaborative Robots
Strengths	• Learn new tasks efficiently and fast. • Able to sense, adapt to circumstances, apply tolerance compensation, and independently solve problems. • Move freely within the production system. • Handle complex parts. • Can be assigned to completely different tasks. • Limited associated fixed costs.	• Execute tasks effortlessly with high accuracy and repeatability. • Able to maintain constant performance over time. • Capable of working non stop. • Able to perform repetitive and non ergonomic tasks, and handle hazardous parts without health risks.
Weaknesses	• Subject to physical and performance limits (e.g., low accuracy and repeatability). • Subject to performance deterioration over time (fatigue). • Work for a limited period. • Subject to health issues and regulatory limitations (e.g., force, duration, velocity limits).	• Able to perform only programmed tasks. • Unable to assess and react to the result of the performed task. • Execute tasks only within a fixed working area. • Handle parts only compliant with the end effector. • Hardly relocatable; represents a fixed cost even if workload fluctuates. • High costs for installation, implementation, maintenance, and certification.

Source: Adapted from [16]

2.3.2. Static Task Scheduling (Offline)

Static approaches tackle the problem of task allocation prior to the actual execution of the assembly process. The defining characteristic of these methods is their rigidity: once the task allocation and sequence are determined, they generate a **fixed process plan** that cannot be changed during execution, regardless of real time variations.

The preliminary step in static scheduling is the *suitability assessment*, which evaluates the fitness of humans and robots for specific tasks. After that a decision logic is applied to generate the final schedule. This process is typically performed in one of two ways:

- **Expert based Planning:** An expert planner manually defines the process plan based on the suitability information [17].
- **Optimization based Planning:** Mathematical optimization methods are applied

to minimize specific objective functions, such as execution time [13], production cost [25], or multi criteria indices that balance efficiency and ergonomics [6]. The literature highlights various approaches, ranging from exact Branch and Bound algorithms to heuristic genetic algorithms.

2.3.3. Dynamic Task Scheduling (Online)

The reliability of the actual execution of an offline task allocation plan is limited due to uncertainties and changes in the workspace as well as unpredictable events and human decisions [3]. There are multiple reasons why this can happen, e.g. safety related evasive movements, stops caused by unintentional contact with workers, worker attention and fatigue throughout the day. Consequently, these disturbances result in the ineffectiveness and sub optimality of offline optimized task assignment [1].

Dynamic solutions have been proposed; they address these limitations by continuously adapting to the operational context in real time. However, to enable this level of responsiveness, the system must satisfy specific functional and computational requirements:

1. **Real time Task Monitoring:** The system requires continuous feedback on the execution status of both agents. While the robot's progress can be deterministically monitored via its controller, detecting the human's task completion is more complex. It typically relies on explicit confirmation interfaces (e.g., digital buttons, pedals) or advanced perception systems capable of tracking the operator's state and classifying their actions.
2. **Computational Efficiency:** Unlike static optimization methods which are computed offline with ample time, dynamic approaches must operate within strict time constraints. The decision making algorithms must be computationally efficient to ensure the system can react online without introducing significant latency to the production cycle.

The survey [19] further proposes two different categories: reactive and ad-hoc task allocation and proactive task allocation.

Reactive and ad-hoc task allocation

Reactive task allocation approaches have a central planning instance in common, which assigns tasks to the human or the robot. There are basically two approaches, one is based on an initial plan that is dynamically adapted in case of disturbances, e.g. by changing individual assignments [1], or by triggering a re-planning of all remaining tasks [18] [11].

As for the second type, the allocation is based on an ad-hoc decision logic for allocation, which is applied online for assigning each task, e.g. based on the task classification [2] or based on human fatigue [15].

A crucial distinction lies in the decision making authority: some approaches are suggestion based, thus allowing the human to decide whether or not to perform the assigned task [15], some follow a command based methodology, i.e. the decision making authority belongs only to the central task planning [1] [18].

Proactive Task Allocation

In proactive collaborative frameworks, the human operator retains full autonomy over their actions, effectively assuming the role of the *team leader*. Consequently, the robot adopts an *assistant* role (or follower), dynamically selecting actions to either alleviate the operator's workload or minimize the overall cycle time based on the human's current behavior. To achieve this, the system must deal with a critical trade-off between responsiveness and reliability [19]:

- On one hand, minimizing idle time requires the robot to commit to an auxiliary action as early as possible (anticipation).
- On the other hand, to avoid synchronization errors or hazardous interferences, the system requires a high probability of correctly predicting the human's intention and task completion time, which typically necessitates a longer observation window.

To address the challenge of estimating human intentions, the literature primarily utilizes inference mechanisms combined with probabilistic statistical models. Techniques include Dynamic Bayesian Networks (DBN), Higher order Markov Chains, and Gaussian Mixture Models (GMM) [19].

From an operational perspective, the primary benefit of proactive allocation lies in the human centric improvement of working conditions, specifically through increased flexibility and reduced monotony. Conversely, these benefits can compromise system determinism, potentially resulting in lower efficiency, volatile cycle times, and a lack of uniformity in performance across the production line.

2.3.4. Learning based Task Scheduling

With the rapid advancement of Artificial Intelligence (AI), learning based methods have emerged as a robust alternative to conventional optimization techniques for solving task scheduling problems. Unlike these established approaches, Reinforcement Learning (RL)

operates through a trial and error process typically organized into episodes. In this framework, an agent receives scalar rewards or penalties based on its actions, allowing it to gradually improve its decision making policy without being explicitly programmed or given an explicit model of the environment dynamics.

Learning based solutions can be categorized into multi agent and single agent frameworks.

Multi agent approaches formulate the collaborative planning problem as a joint scheduling task, where both the human and the robot are modeled as controllable agents. Tasks are allocated according to a global optimality criterion to achieve the shared goal. These methods can be classified as either static requiring the human to adhere to a prescribed plan or dynamic/reactive, where both agents must conform to a learned joint policy online [4, 5]. As previously noted, a significant limitation of this paradigm is the constraint it places on human autonomy, often reducing the collaborative nature of the interaction by treating the operator as a system component to be optimized.

Single agent solutions, conversely, can be classified as dynamic reactive approaches where the robot is the sole learning agent. While this restores human autonomy, it introduces the challenge of requiring accurate predictions of human behavior to inform the policy. A recent work [7] proposed formalizing the problem as a Discrete Event Markov Decision Process. This framework integrates a Hierarchical Task Model (HTM) with Bayesian inference for human intent recognition, allowing the system to model diverse action types (e.g., parallel, sequential, joint, asynchronous). This formulation enables the RL agent to handle complex dynamic events such as human changes of mind and execution failures while overcoming the computational complexity limits of graph based solutions. Validation on real world assembly tasks (e.g., IKEA furniture) demonstrated that this approach significantly outperforms standard greedy or random policies in terms of efficiency and collaboration quality.

Fundamentally, Reinforcement Learning acts as a computational solver for the task scheduling problem. Since the underlying problem formulation comprising states, resources, and constraints remains consistent with conventional setups, RL serves as a robust alternative technique for deriving the optimal control policy. This approach is particularly advantageous in online domains, as the computational burden of policy optimization is relieved to an offline training phase, typically conducted within a simulation environment. Consequently, the learned policy can be evaluated in real time with minimal latency (fast inference), enabling the execution of complex, adaptive behaviors without the runtime computational costs associated with traditional online replanning.

2.4. Contribution of this work

While a significant portion of the state of the art research explores applications to collaborative assembly scenarios, this thesis addresses a distinct yet equally critical domain: collaborative logistics, specifically focusing on packaging and palletizing operations.

Although the fundamental dynamics of human-robot interaction remain consistent across these domains, the specific operational challenges differ. In assembly, the complexity often lies in task allocation (determining who performs a task based on capability). In contrast, within logistics contexts, task allocation is frequently pre-determined. Consequently, the critical challenge shifts to the optimal temporal sequencing and coordination of these tasks in real time.

The primary objective of this work is to bridge this gap by adapting learning-based strategies, traditionally studied in assembly, to the logistics domain. The proposed solution utilizes Reinforcement Learning to develop a human-aware orchestrator capable of governing an environment where spatial and temporal precision are paramount. This framework contributes a dual-objective policy: it optimizes the sequencing of robot actions to maximize efficiency while proactively adapting to the human operator's workspace. By anticipating human movements and staying clear of their operating zones, the orchestrator minimizes the triggering of underlying hardware safety stops, thereby preserving process fluidity and maintaining an intuitive workflow.

3 | Theoretical Background

This chapter establishes the mathematical and algorithmic foundations of Reinforcement Learning (RL) used in this thesis. It begins by defining the core components required to formalize the RL problem, drawing on the standard framework of Markov Decision Processes (MDPs). Fundamental concepts of states, actions, and rewards are introduced, alongside the policy and value functions that drive the agent’s decision making process.

Subsequently, practical implementation challenges are addressed, including function approximation via neural networks, the distinction between on-policy and off-policy learning, and the exploration-exploitation trade-off. Primary families of RL algorithms such as value-based, policy-based, and actor-critic methods are categorized to contextualize the specific approach chosen for this work.

Finally, the chapter introduces Proximal Policy Optimization (PPO), the primary algorithm used in this thesis. It outlines the stability and efficiency that make PPO a standard choice, alongside a discussion on how Action Masking improves training performance.

3.1. Reinforcement Learning

Reinforcement Learning (RL) is a framework designed to solve *sequential decision making problems*, where decisions are taken iteratively over time to optimize an objective function, typically by maximizing cumulative rewards or minimizing associated costs. Each decision influences future system states and subsequent decisions, highlighting the sequential and interactive nature of such problems.

At the core of this framework are two principal components: the agent and the environment. The agent actively observes the environment, takes actions, and influences its evolution. The environment, in turn, transitions between states based on these actions and provides the agent with new observations and feedback in the form of rewards.

The fundamental goal of RL is to enable the agent to learn optimal behavior through this direct interaction, rather than relying on explicit supervision or complete environmental models. To mathematically express this process, the framework relies on the specific

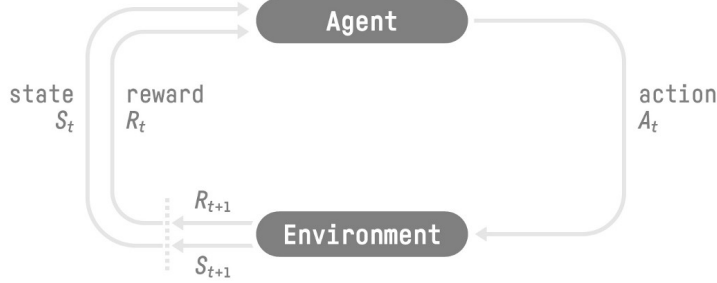


Figure 3.1: The Agent-Environment Interaction Loop. At each time step t , the agent receives state S_t and reward R_t from the environment and executes action A_t , which triggers the transition to state S_{t+1} and reward R_{t+1} .

components detailed below.

3.1.1. Core Components of RL

Drawing on the foundational work of Sutton and Barto [24], the key mathematical definitions required to formalize the Reinforcement Learning problem are introduced here.

Markov Decision Processes (MDP)

A Markov Decision Process (MDP) is a mathematical framework used to represent sequential decision-making problems over discrete decision steps t . It captures the environment's dynamics, where the transition to a new state depends solely on the current state and the action taken by the agent.

Specifically, an MDP is defined as a tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$, where:

- $\mathcal{S}$ is a (finite or infinite) set of *states*,
- $\mathcal{A}$ is a set of *actions*,
- $P(s' \mid s, a)$ is the *state transition probability function*, representing the probability of transitioning to state $s' \in \mathcal{S}$ given the current state $s \in \mathcal{S}$ and action $a \in \mathcal{A}$,
- $R(s, a)$ is the *expected reward* received after executing action a in state s ,

- $\gamma \in [0, 1]$ is the *discount factor* balancing immediate and future rewards.

The foundational assumption of an MDP is the Markov property, indicating that state transitions depend solely on the current state and action, independent of prior history:

$$P(s_{t+1} = s' \mid s_0, a_0, \dots, s_t = s, a_t = a) = P(s_{t+1} = s' \mid s_t = s, a_t = a).$$

Policy

A *policy*, denoted by π , represents the core decision-making strategy of the agent. Simply put, it dictates how the agent behaves at any given moment by mapping the observed states of the environment to specific actions. Policies generally take one of two forms: deterministic or stochastic. A deterministic policy maps a state to one single, guaranteed action. However, in complex or uncertain environments, a stochastic policy is often preferred. Rather than choosing a single action, it provides a probability distribution over all possible actions for a given state, formally denoted as $\pi(a \mid s)$. When the agent selects an action based on this probability distribution, it is expressed mathematically as $a \sim \pi(s)$.

Reward

The *reward signal* defines the core objective of the Reinforcement Learning problem. It represents the instantaneous, numerical feedback r_t that the environment sends to the agent at every discrete decision step t . Essentially, the reward communicates what is "good" and what is "bad" to the agent, specifying the immediate goal without explicitly teaching it *how* to achieve it.

Value Function

While the reward signal indicates the immediate desirability of a specific state transition, the agent's true objective is to maximize its total return over the long term. To achieve this, it relies on the *value function*, which evaluates the long-term potential of states (or state-action pairs) by computing expected cumulative future rewards. Two primary value functions are the state-value function $V(s)$ and the action-value function $Q(s, a)$, defined as follows:

$$V^\pi(s) = \mathbb{E}_{s_0=s, \tau \sim \pi} \left[\sum_{t=0}^{\tau} \gamma^t r_t \right] \quad (3.1)$$

$$Q^\pi(s, a) = \mathbb{E}_{s_0=s, a_0=a, \tau \sim \pi} \left[\sum_{t=0}^T \gamma^t r_t \right] \quad (3.2)$$

Here, $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \dots, s_T, a_T, r_T)$ denotes a trajectory of the MDP generated by following the policy π , starting from the initial state s_0 . The time horizon T defines the number of steps considered, which can either be finite or tend to infinity in the case of continuing tasks.

Bellman Equation

The Bellman equation is fundamental to the study of MDPs, as it expresses the recursive relationship between the value of a state and the values of its successor states under a specific policy π :

$$V^\pi(s) = \mathbb{E}_{a \sim \pi(\cdot|s)} \left[R(s, a) + \gamma \sum_{s'} P(s' | s, a) V^\pi(s') \right]$$

Similarly, the Bellman equation for the action-value function is defined as:

$$Q^\pi(s, a) = R(s, a) + \gamma \sum_{s'} P(s' | s, a) \mathbb{E}_{a' \sim \pi(\cdot|s')} [Q^\pi(s', a')]$$

These equations provide a critical mechanism to iteratively compute value functions and evaluate policies, forming the mathematical backbone of almost all RL algorithms.

Parametrization

In environments with small, discrete state spaces, value functions and policies can be represented exactly using simple tables. However, to efficiently scale RL to complex, high-dimensional problems, one must assume a parametric structure where a set of weights or parameters θ is optimized to approximate the desired functions. While linear function approximation is a classical method, neural network-based approximation deserves special attention. Neural networks are powerful non-linear function approximators that offer strong generalization properties; in fact, nearly all major recent breakthroughs in Deep Reinforcement Learning (DRL) rely heavily on them [8].

On-policy vs. Off-policy Methods

On-policy methods learn directly from experience gathered by the current, active policy. Off-policy methods, however, can utilize experiences collected by other policies (such as older versions of the agent or human demonstrations). While off-policy learning significantly enhances sample efficiency, it introduces additional mathematical complexities regarding convergence and training stability.

Exploration vs. Exploitation Trade-off

A central challenge in RL is the exploration-exploitation dilemma. The agent must continuously balance *exploration* (testing new, untried actions to discover potentially better outcomes) with *exploitation* (greedily selecting actions already known to yield high rewards). Navigating this trade-off effectively is crucial for preventing the agent from prematurely converging on sub-optimal behaviors in unknown environments.

Value-Based vs. Policy-Based vs. Actor-Critic

Reinforcement Learning algorithms generally fall into three distinct categories: value-based, policy-based, and actor-critic approaches.

Value-based methods aim to estimate the optimal value function (commonly the action-value or Q-function). The policy is then derived indirectly, for instance, by always choosing the action with the highest expected Q-value. In contrast, policy-based methods directly parameterize and optimize the policy π without explicitly maintaining a value estimate.

Actor-Critic methods combine the strengths of both approaches by maintaining two distinct neural networks (or branches): an *Actor* and a *Critic*. The Actor directly parameterizes and learns the policy, while the Critic estimates the value function. During training, the Critic evaluates the current policy's performance, providing a baseline value estimate to guide and stabilize the Actor's gradient updates.

3.1.2. Proximal Policy Optimization (PPO)

Building on the Actor-Critic framework, Proximal Policy Optimization (PPO) [23] has emerged as a standard algorithm for reinforcement learning. It is an on policy method designed specifically to address the stability challenges inherent in standard policy gradient approaches.

In standard policy gradients, selecting an appropriate step size is critical; a single large update can push the policy into a region of the parameter space where performance

collapses. This scenario is particularly damaging because the agent then generates poor quality trajectories, which are subsequently used for further training, creating a downward spiral that is difficult to recover from.

Proximal Policy Optimization can be viewed as an evolution of the Trust Region Policy Optimization (TRPO) algorithm. To address the stability issues inherent in policy gradients, TRPO ensures that a policy update is performed only if it guarantees an improvement over the current policy¹.

The resulting optimization problem is formulated as follows:

$$\begin{aligned} \max_{\theta} \quad & J^{CPI}(\theta) \\ \text{subject to} \quad & \mathbb{E}_t [\text{KL}(\pi_{\theta}(\cdot|s_t) \parallel \pi_{\theta_{\text{old}}}(\cdot|s_t))] \leq \delta \end{aligned} \quad (3.3)$$

where KL denotes the Kullback Leibler divergence. In this context, A^{π} is the advantage function, defined as the difference between the action value and state value functions: $A^{\pi}(s, a) = Q^{\pi}(s, a) - V^{\pi}(s)$. The surrogate objective J^{CPI} is given by:

$$J^{CPI}(\theta) = \mathbb{E}_{\tau \sim \pi} \left[\sum_{t \geq 0} \frac{\pi_{\theta}(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} A^{\pi}(s_t, a_t) \right] \quad (3.4)$$

This term serves as an approximation of the performance difference $J(\pi') - J(\pi)$. Maximizing this identity is the ideal optimization goal, as any positive result leads to a superior policy; specifically, choosing $\pi' = \pi$ yields a difference of zero. By satisfying the constraint, the desired monotonic improvement property is maintained.

PPO provides a heuristic to solve this problem more efficiently by transforming the constrained task into an unconstrained optimization. This is achieved via the clipped objective function:

$$J^{CLIP}(\theta) = \mathbb{E}_t [\min(r_t(\theta)A_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon)A_t)] \quad (3.5)$$

where the probability ratio is defined as:

$$r_t(\theta) = \frac{\pi_{\theta}(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \quad (3.6)$$

This formulation results in several characteristics that contribute to its selection as a standard choice for control problems:

¹A policy π' is considered strictly better than π if $J(\pi') > J(\pi)$.

- **Stability:** The clipping mechanism acts as a safeguard, preventing catastrophic performance drops by limiting the magnitude of policy changes.
- **Reliability:** It demonstrates consistent performance across diverse tasks, often requiring less hyperparameter tuning than other actor critic methods.

Finally, in many complex control tasks, not all actions within the discrete action space are valid in every state. An invalid action might be physically impossible (e.g., placing an object in an already full slot) or explicitly forbidden by safety constraints.

A common, naive approach to handle this is to allow the agent to select the invalid action but penalize it heavily with a massive negative reward. However, this method is highly sample-inefficient; the agent must still waste valuable training epochs actively exploring and "learning" to avoid these useless paths.

To overcome this, *Action Masking* can be directly integrated into the algorithm. Masked PPO [26] explicitly accounts for these constraints by applying a dynamic binary mask to the policy network's output. By overriding the logits of invalid actions (typically setting them to $-\infty$ prior to the softmax activation), the algorithm associates a probability of exactly zero with masked actions. This ensures the agent never even considers forbidden transitions, dynamically reducing the search space. Consequently, Masked PPO eliminates wasted exploration, providing significantly faster convergence and higher cumulative returns for the same number of training epochs.

4 | System Description and Problem Formulation

This chapter introduces the specific experimental use case that forms the core of this thesis: a collaborative logistics workstation physically realized within the joint laboratory of Camozzi Automation S.p.A. and Politecnico di Milano. It details a cobot environment designed to address the ergonomic challenges of manual packaging, such as musculoskeletal disorders caused by repetitive heavy lifting, while maintaining the layout flexibility required in modern production lines. This workstation combines a 7-degree-of-freedom robotic manipulator with a human operator acting as a fully independent 'Leader', leveraging the cobot's heavy-lifting capabilities while preserving the human's cognitive flexibility.

Structurally, the discussion progresses from the tangible hardware setup to the underlying mathematical framework of the proposed robotic orchestrator. First, it defines the operational scenario, detailing the physical workstation layout, the manipulation objects (pneumatic pieces, small boxes, and crates), and the specific atomic actions assigned to both the human and the robotic agent. Following a clear definition of the production cycle and its strict logical constraints, the chapter then formalizes the primary control challenge. It mathematically frames a dual-objective optimization problem aimed at minimizing the total cycle time (makespan) while actively preventing disruptive Speed and Separation Monitoring (SSM) safety halts. Finally, it establishes the objective function that the dynamic, online scheduler must solve to balance absolute spatial efficiency with proactive social safety and continuous process fluidity.

4.1. The Operational Scenario

The operational scenario is based on a collaborative logistics workstation designed for material manipulation, physically realized within the joint laboratory of Camozzi Automation S.p.A. and Politecnico di Milano. The system comprises a human operator and a robotic agent working simultaneously within a shared workspace to achieve a common palletizing objective.

The motivation for this setup comes from the limitations of manual labor in logistics. As highlighted in the state of the art analysis, section 2.1, tasks such as packaging frequently expose workers to physical strain and musculoskeletal disorders due to repetitive heavy lifting. Although traditional automation offers a solution to these ergonomic challenges, it often introduces rigidity and requires extensive floor space for safety fencing. To resolve this trade-off, the proposed scenario integrates a fenceless collaborative robot, preserving layout flexibility while safely offloading physically demanding tasks from the operator.

The workflow is designed to leverage the distinct strengths of both agents: the human operator handles flexible tasks, such as navigating potentially complex environments to supply components, while the cobot executes heavy lifting and monotonous pick and place operations within a more controlled setting.

The following paragraphs provide a detailed description of this workstation, defining the specific actions performed by both the robot and the human operator, as well as the operational constraints governing their execution.

Robotic Hardware and Layout

The workstation features a 6-degree-of-freedom (DoF) collaborative robot mounted on a linear guide. This adds a seventh DoF, allowing the robotic manipulator to reach the entire surface of the worktable (as illustrated in Figure 4.1).

An inclined gravity slide is mounted on the table to hold up to three crates, visible on the right side of Figure 4.1. This slide features mechanical stops (locks) that hold the crates in specific positions. When a lock is released, gravity smoothly advances the crate down to the next processing stage.

Additionally, the workstation is equipped with strategically placed physical buttons. These interfaces allow the human operator to confirm the completion of manual tasks, such as closing the small box lids and restocking components on the line, as detailed in the following sections.

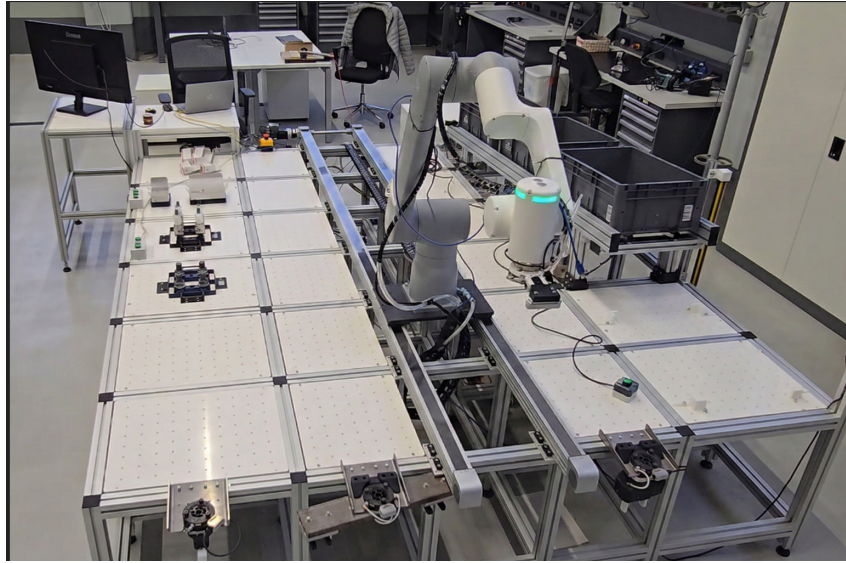


Figure 4.1: The collaborative logistics environment realized in the joint laboratory between Camozzi Automation S.p.A. and Politecnico di Milano.

Finally, to handle different types of objects, the robot is equipped with an automatic tool changer, enabling seamless switching between the following two end-effectors :

- **The Crate Gripper (Figure 4.2a):** A larger, robust tool designed to handle the weight and dimensions of the crates.
- **The Component Gripper (Figure 4.2b):** A smaller, precise tool used to pick up individual components and small boxes.



(a) The Crate Gripper.



(b) The Component Gripper.

Figure 4.2: The two interchangeable tools used during the experiment.

Manipulation Objects

- **Pieces:** The primary pneumatic components undergoing the packaging process. The experimental setup uses two distinct variants, designated as Type A and Type B. Both variants are manipulated using the Component Gripper.



(a) Type A.



(b) Type B.

Figure 4.3: The two types of pieces.

- **Small Boxes:** Intermediate packaging containers designed to house the individual pneumatic components. The robot can grasp and transfer these boxes using the same Component Gripper used for the pieces.



Figure 4.4: A small box.

- **Crates:** The larger containers used for the final consolidation of the packaged components. They can accommodate multiple small boxes. Their physical dimensions and payload require the use of the Crate Gripper.



Figure 4.5: The crate.

Layout and Zones

The workstation layout can be decomposed into distinct zones, categorized by their primary functionality. Figure 4.6 illustrates the top-down view of the setup, while the specific role of each zone is detailed below.

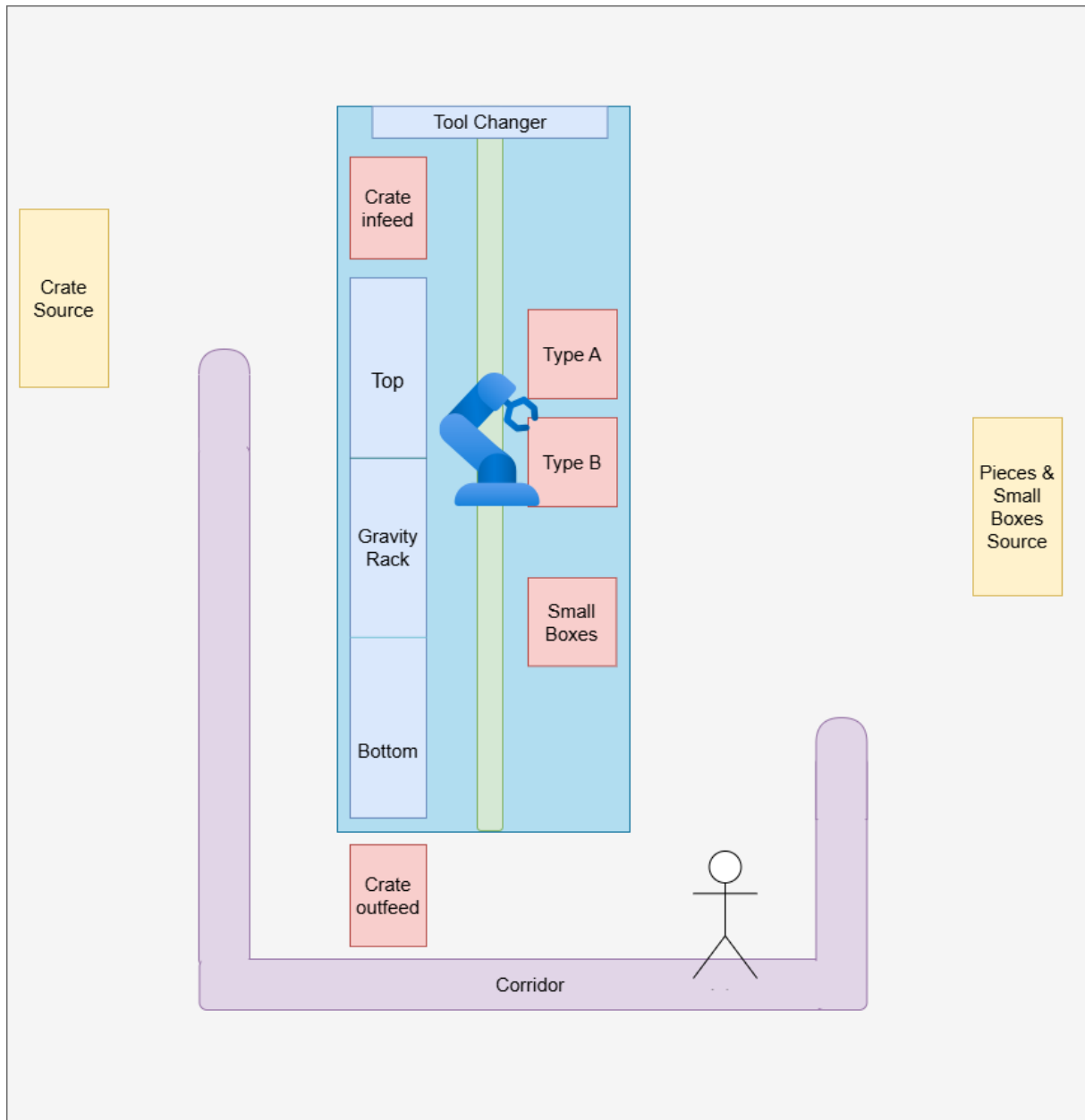


Figure 4.6: Top-down view of the workspace zones.

The Workbench (Robot Area) The primary table where the robot operates is divided into the following zones:

- **Type A Staging Area:** The specific location on the workbench reserved for staging

Type A pieces.

- **Type B Staging Area:** The dedicated placement area for Type B pieces.
- **Small Boxes Staging Area:** The designated buffer zone for the empty small boxes.
- **Empty Crate Infeed:** The loading position for introducing empty crates into the system.
- **Crate Outfeed:** The discharge position for retrieving fully packaged crates.
- **Gravity Rack:** The intermediate buffering zone for crates during the filling process; it is divided into top, middle, and bottom tiers.
- **Tool Changer:** The designated area where the robot deposits its current gripper and retrieves the alternative one.

Human Supply Areas These zones represent the areas outside the immediate worktable where the operator retrieves necessary items:

- **Pieces and Small Boxes Source:** The storage area where the operator collects new Type A/B pieces and empty small boxes.
- **Crate Source:** The storage area where the operator retrieves empty crates.
- **Corridor:** The transit path the operator uses to navigate between the left and right sides of the workspace.

4.1.1. Task Definition and Constraints

The logistics process is decomposed into atomic actions that define the capabilities of each agent within the workspace.

Robot Actions The robot executes the primary pick and place operations, moving items between the workbench zones defined in Section 4.1:

1. **Position_Crate:** Picks an empty crate from the Empty Crate Infeed and places it onto the top tier of the Gravity Rack.
2. **Place_Small_Box:** Picks a small box from the Small Boxes Staging Area and places it into a crate positioned within the Gravity Rack.
3. **Pick_Piece_A:** Picks a Type A piece from the Type A Staging Area and places it into a specific small box located in the Small Boxes Staging Area.

4. **Pick_Piece_B**: Picks a Type B piece from the Type B Staging Area and places it into a specific small box located in the Small Boxes Staging Area.
5. **Remove_Crate**: Picks a fully loaded crate from the bottom tier of the Gravity Rack and places it into the Crate Outfeed.
6. **Tool_Change**: Navigates to the Tool Changer to swap its current gripper for the alternative one.

Human Actions The human operator is responsible for logistics (restocking) and finishing tasks, transferring materials from the supply areas to the active workbench areas:

1. **Refill_Type_A**: Retrieves Type A pieces from the Source Area and restocks the Type A Staging Area.
2. **Refill_Type_B**: Retrieves Type B pieces from the Source Area and restocks the Type B Staging Area.
3. **Refill_Small_Boxes**: Retrieves empty small boxes from the Source Area and restocks the stack in the Small Boxes Staging Area.
4. **Refill_Crates**: Retrieves empty crates from the Crate Source and loads them into the Empty Crate Infeed.
5. **Close_Box**: Manually closes the lid of a small box once it is fully loaded.

The logistics line also features an automated gravity rack mechanism. Its stop gates release automatically when necessary, ensuring that crates always descend to the lowest available position on the rack.

Constraints The execution of these actions is limited by the state of the environment. We define the following operational rules and precedence constraints:

1. **Single Type Constraint (Small Boxes)**: A small box must contain only one type of component (exclusively Type A or Type B); mixing types within a single box is not allowed. A small box can be moved into a crate only if it is full and closed.
2. **Homogeneity Constraint (Crates)**: A crate must contain only small boxes of the same type. Furthermore, a crate can be removed from the gravity rack only if it is full.
3. **Capacity Limits**: The worktable, the gravity rack, the small boxes, and the crates have finite capacities. Neither the robot nor the operator can place a new object if

the destination is already at its maximum capacity.

4. **Safety Constraint (SSM):** The robot's maximum velocity depends on the distance between it and the human operator. The closer the human is to the robot, the slower the robot must move, reducing speed until a full stop is reached.

4.2. The Production Cycle

Having defined the atomic actions and physical constraints, a simplified workflow for producing a single crate is presented below to illustrate the basic process. In this elementary cycle, the flow proceeds as follows:

- The robot equips the Crate Gripper and moves an empty crate from the Empty Crate Infeed to the Gravity Rack, which automatically advances it to the bottom position.
- The robot changes tools to begin picking Type A pieces and placing them into a small box, designating it as a Type A box.
- Once a small box is full and manually closed by the human operator, the robot moves it into the crate, effectively setting the crate's overall classification to Type A.
- This process repeats continuously, with the human operator actively restocking new pieces and empty small boxes as needed.
- When the main crate reaches full capacity, the robot switches back to the Crate Gripper and moves the finished crate to the Crate Outfeed.

It is important to note that the staging areas, along with the crate infeed, represent the primary shared workspace. These areas exhibit the highest density of human-robot interaction and are the most critical zones for spatial coordination.

General Operational Scenario While the workflow above illustrates a sequential single type process, the general scenario addressed is more complex. In the full operational setting, multiple crates may occupy the gravity rack simultaneously, potentially requiring different component types. The robot must dynamically alternate between Type A and Type B components, optimizing the filling sequence for multiple crates in parallel while strictly adhering to the homogeneity and precedence constraints.

4.3. Problem Statement and Objective

The primary objective of this thesis is to develop a robotic orchestrator capable of solving a dual-objective optimization problem: maximizing system throughput while simultaneously minimizing interference with the human operator.

To address the limitations of static planning discussed in 2.3, the robot’s actions are scheduled *online*. This enables a dynamic approach capable of adapting to process variability in real time. Furthermore, the human operator retains full autonomy (acting as the ‘Leader’), making independent decisions without adhering to external commands. Consequently, the proposed solution is classified as a dynamic reactive approach, in which the orchestrator determines the optimal action based on the instantaneous state of the system, treating the human as an exogenous input.

Within this framework, human avoidance is treated as a core performance objective rather than merely a reactive safety constraint. A critical physical constraint of this shared environment is the underlying Speed and Separation Monitoring (SSM) safety system, which forces the robot to dynamically reduce its speed, and eventually halt entirely, whenever the human operator breaches the minimum safe separation distance. To maintain process fluidity and avoid these disruptive SSM halts, the orchestrator learns to anticipate the operator’s workspace occupancy, proactively scheduling tasks to maintain a safe spatial distance.

However, due to the constrained layout of the workstation the robot and the human operator must frequently access the exact same staging areas. In such a highly shared environment, maximizing throughput increases the likelihood of spatial overlap. Consequently, the control problem becomes one of balancing two inherently conflicting objectives:

1. **Efficiency:** Minimization of the total cycle time (makespan) required to produce the full crates.
2. **Process Fluidity (Social Safety):** Minimization of close proximity situations to prevent disruptive safety stops, ensuring an uninterrupted and intuitive collaborative workflow.

Formally, the optimization problem is defined as finding the optimal scheduling policy π^* that minimizes a composite objective function $J(\pi)$:

$$\pi^* = \arg \min_{\pi} \mathbb{E}_{\pi} \left[C_{\text{cycle}} + \lambda \sum_{t=0}^T \mathcal{P}_{\text{prox}}(s_t, a_t) \right] \quad (4.1)$$

subject to:

- **Safety Constraints:** The robot's velocity adheres to the Speed and Separation Monitoring (SSM) regime.
- **Precedence Constraints:** An action can only be executed once all prerequisite tasks have been completed (e.g., a small box cannot be placed into a crate before being filled and closed).
- **Resource Constraints:** No action may violate the physical capacity limits of buffers or crates (e.g., the robot cannot place an item if the destination is full).

The objective function terms are mathematically defined as follows:

- C_{cycle} denotes the total makespan required to complete the palletizing order.
- T denotes the total number of decision steps executed by the robot to complete the assigned palletizing order.
- $\mathcal{P}_{\text{prox}}(s_t, a_t)$ is a proximity penalty function that assigns a high cost when action a_t leads to a violation of the predefined safety distance with respect to the human operator, and evaluates to 0 otherwise.
- λ is a scalar weighting parameter that regulates the trade-off between productivity and human avoidance. Larger values of λ yield more conservative policies, whereas smaller values prioritize nominal throughput, potentially triggering SSM safety interventions.

4.4. High Level System Architecture

To solve this optimization problem online, a modular control architecture is employed. This framework allows the robot to perceive the dynamic environment and adapt its schedule in real time, as required by the problem statement.

As illustrated in Figure 4.7, the system comprises three main functional blocks:

- **Perception Layer:** This module is responsible for monitoring the collaborative workspace. It retrieves all relevant information to construct the state vector used by the scheduler. Specifically, it monitors:
 - **The Human State:** It tracks the operator's real time position to calculate the robot human distance and identifies their current activity status (specifically, the action currently being performed).

- **The Process State:** It monitors the status of the logistics line, including the execution progress of the robot's current action and the availability of resources in the bins and crates.
- **Decision Layer (The Scheduler):** This is the core intelligence of the system. It receives the state vector and determines the optimal high level action.
- **Execution Layer:** A low level controller translates the abstract action (e.g., *Pick_Piece_A*) into trajectory planning and motor commands.

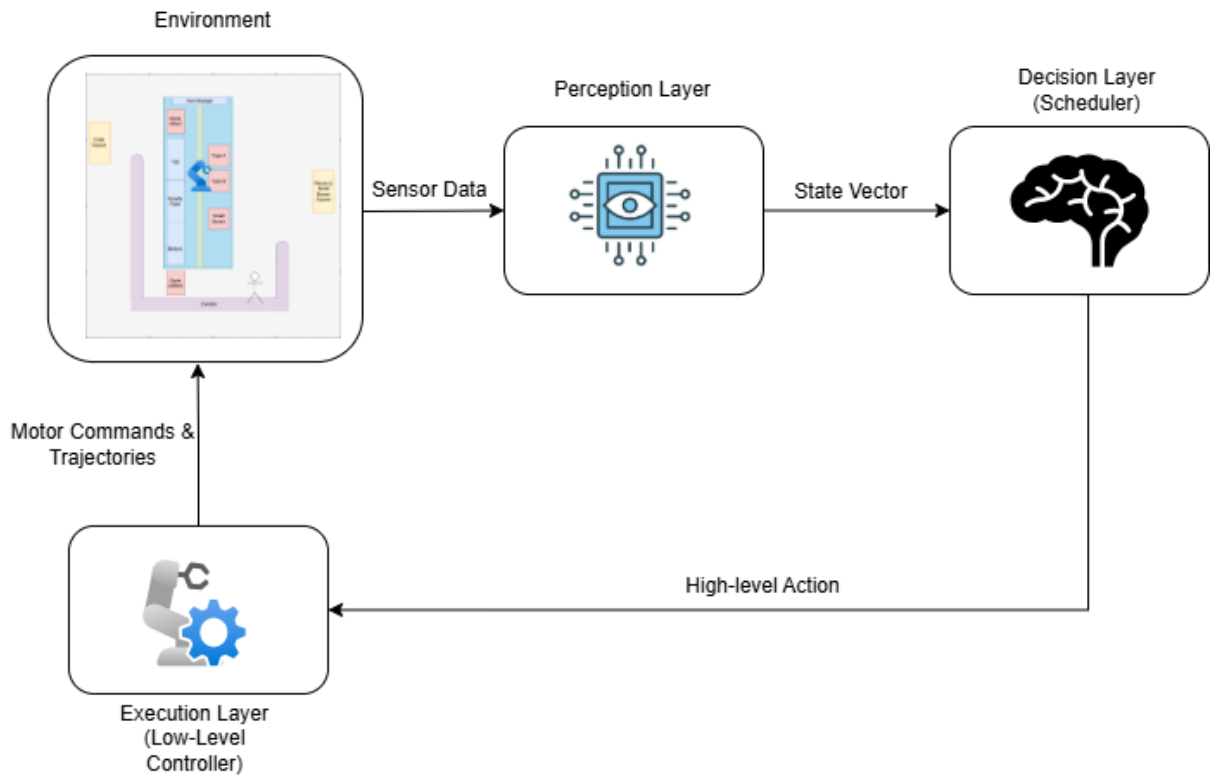


Figure 4.7: The proposed high level control architecture, linking perception, decision making, and execution.

5 | Methodology and Implementation

The preceding chapter established the physical architecture of the collaborative workstation and defined the primary optimization challenge: maximizing system throughput while simultaneously minimizing interference with the human operator.

This chapter presents the proposed solution methodology, which leverages Reinforcement Learning (RL) to derive the optimal control policy. Implementing this approach requires two fundamental steps. First, the operational environment is formalized as a Markov Decision Process (MDP), thereby providing the mathematical framework required for a Reinforcement Learning algorithm, specifically, Masked Proximal Policy Optimization (PPO) to learn effectively. Second, the practical training pipeline is developed by integrating the high fidelity Siemens Tecnomatix simulation environment with a Python based implementation of Masked PPO using the Stable Baselines3 (SB3) library.

5.1. Formal Modeling of the Collaborative System

As detailed in Section 4.1, the collaborative workstation involves two agents: the human, who acts as an autonomous, independent agent, and the robot, which is the controllable agent governed by our policy. Therefore, modeling this system requires two key components: a behavioral model of the human and a formal definition of the underlying Markov Decision Process (MDP) representing the environment.

5.1.1. Human Modeling

To model human behavior, we implemented a Finite State Machine (FSM). The states of the FSM represent the human's current action or goal. Upon completing a specific task, the transition to the next goal is determined by giving equal probability to the possible next actions he can perform. This stochastic approach reflects real world variability, ensuring that the human agent does not deterministically choose the same subsequent

action under identical conditions.

Specifically, each FSM state corresponds to a distinct "goal zone" that the human intends to reach. Referencing the workspace layout defined in Section 4.1, the FSM comprises the following states:

- **Idle and Safety States:**

- **Idle:** The operator is momentarily inactive within the workspace. This state models natural pauses or "thinking time" that occur between operations.
- **MoveToSafety:** The operator retreats to a designated safe area away from the robot's reach. This models the scenario where the human has no pending tasks and must clear the active workspace.

- **Picking States:** The operator moves to the storage areas to retrieve components.

- **PickTypeA and PickTypeB:** Retrieval of Type A and Type B pieces from their designated storage locations.
- **PickSmallBoxes:** Retrieval of empty small boxes from the small box storage area.
- **PickCrates:** Retrieval of empty crates from the crate storage area.

- **Placing States:** The operator moves to specific work areas to deposit retrieved items.

- **PlaceLineA and PlaceLineB:** The human moves to place Type A and Type B components into the Type A and Type B Staging Areas.
- **PlaceSmallBoxes:** The human places new small boxes in the Small Boxes Staging Area.
- **PlaceCrates:** The human deposits the retrieved crates into the Empty Crate Infeed.

- **Other Actions:**

- **CloseSmallBoxes:** The operator closes the lids of the full small boxes located in the Small Boxes Staging Area.

It is important to note that the human agent cannot select every action at every decision step; the availability of actions is constrained by the state of the workstation. For example, a placement action cannot be executed if the Type A Staging Area is already

at maximum capacity. Conversely, the picking zones are modeled as having an infinite supply of components, meaning picking actions are never constrained by stock shortages.

To illustrate this probabilistic decision making process, consider the following scenario. Suppose the human has just finished placing a crate in the Empty Crate Infeed, filling it to capacity. Immediately after, the human transitions to the `Idle` state, representing a momentary thinking pause. Assume the current environmental state is as follows: the Type A Staging Area has available space, the Type B Staging Area is full, and the Small Boxes Staging Area is full but contains small boxes whose lids can be closed. Given these constraints, the FSM assigns a 50% probability that the human will choose `PickTypeA` and a 50% probability that they will choose `CloseSmallBoxes`. If the human stochastically selects `PickTypeA`, they will navigate across the corridor to the storage area, retrieve the component, and subsequently place it in the Type A Staging Area. Upon completing this task, the human returns to the `Idle` state. Assuming the robot has not altered the environment in the meantime, the only remaining valid action is `CloseSmallBoxes`. Once all useful operations are exhausted and no further actions are available, the FSM forces a transition to the `MoveToSafety` state, directing the human away from the active worktable.

A final note regarding the human model is that the time required to complete a given goal depends primarily on the time it takes the operator to reach a specific target zone, which is ultimately dictated by their movement velocity. For simplicity, this study assumes a constant human velocity. While generalizing to a stochastic velocity profile would yield probabilistic goal completion times and more accurately represent a real world scenario, it would also complicate the evaluation of the Reinforcement Learning (RL) policy, making it difficult to isolate and verify the correctness of the agent’s learned behavior.

Table 5.1 summarizes the key parameters governing the simulated human agent, including the constant movement velocity, the duration of the idle state, and representative transit times between key operational zones.

Table 5.1: Summary of Human Agent Simulation Parameters

Parameter / Action	Description	Value
Human Velocity	Constant movement speed	0.7 m/s
Idle Time	Duration of the cognitive pause	1.5 s
Transit: Storage to Infeed	Movement from Empty Crate storage to the Infeed area	3.5 s
Transit: Storage to Staging	Movement from Type A Storage to Type A Staging Area	6.0 s
Transit: Infeed to Storage	Movement from Empty Crate Infeed to Type A Storage	14.5 s

5.1.2. MDP Formulation

The collaborative environment described in Chapter 4 is mathematically formalized as a Markov Decision Process (MDP), defined by the tuple $\langle S, A, R, P, \gamma \rangle$. In this framework, the agent observes the state $s_t \in S$, selects an action $a_t \in A$, receives a reward $r_t \in R$, and transitions to a new state s_{t+1} according to the environment dynamics P .

In the following, we detail the specific formulation of the Action Space (A), Observation Space (S), and Reward Function (R).

Action Space (A)

The action space is discrete and consists of $|A| = 9$ distinct actions $(a_0, a_1, \dots, a_8)$. These actions are designed to take into account the operations that the robot must perform while adhering to the constraints defined previously in Section 4.1.1.

Additionally, a new *wait* operation has been included.

They are categorized as follows:

1. **Component Transfer Actions (a_0, a_1):** These correspond to the `Pick_Piece_A` and `Pick_Piece_B` tasks. The agent explicitly chooses which component type to retrieve and place into the small boxes.
 - a_0 : Transfer Piece A into the chosen available compatible small box (i.e., empty or containing only Type A pieces).
 - a_1 : Transfer Piece B into the chosen available compatible small box (i.e., empty or containing only Type B pieces).
2. **Small Box Logistics (a_2, a_3):** These correspond to the `Place_Small_Box` tasks.
 - a_2 : Transfer a Type A small box into the first available compatible crate (i.e., empty or containing only Type A small boxes), checking from the bottom to the top of the gravity rack.
 - a_3 : Transfer a Type B small box into the first available compatible crate (i.e., empty or containing only Type B small boxes), checking from the bottom to the top of the gravity rack.
3. **Crate Handling (a_4, a_5):** These represent the `Position_Crate` and `Remove_Crate` tasks.
 - a_4 : Move an empty crate to the top position of the gravity rack.

- a_5 : Remove a full crate from the bottom position of the gravity rack.
4. **Tool Configuration** (a_6, a_7): The `Tool_Change` operation is resolved into two distinct setup actions:
- a_6 : Set Gripper for Pieces.
 - a_7 : Set Gripper for Crates.
5. **Wait** (a_8):
- a_8 : **Wait**: The agent deliberately holds its current position and takes no physical action. Although not a physical production task, it serves a dual purpose. First, it ensures operational feasibility when no other valid actions are available to the robot (e.g., waiting for the human operator to complete a prerequisite task). Second, as established in Section 4.3, it is essential for satisfying the goal of minimizing interference with the human operator. By strategically choosing to wait, the robot holds its position until the human clears the shared workspace.

Note that these action definitions alone do not inherently satisfy all the task constraints. While actions a_0, a_1, a_2 , and a_3 are explicitly designed to satisfy part of the Single Type and Homogeneity constraints by preventing the mixing of Type A and Type B items, they do not take into account capacity limits and the readiness of the crates and small boxes that enable them to be moved. These, instead, will be satisfied through the implementation of action masking, as seen later on in 5.1.2.

Table 5.2 summarizes the nominal execution times required for the robot to complete each task defined within the action space. These durations represent the baseline operational times under optimal conditions, assuming no dynamic velocity scaling is triggered by the Speed and Separation Monitoring (SSM) system due to human proximity.

Table 5.2: Nominal Execution Times for Robotic Actions

Action	Task Description	Nominal Time (s)
a_0	Transfer Component A to Small Box	25.7
a_1	Transfer Component B to Small Box	25.7
a_2	Transfer Small Box (Type A) to Crate	23.5
a_3	Transfer Small Box (Type B) to Crate	23.5
a_4	Crate Infeed (Move Empty Crate to Rack)	22.0
a_5	Crate Outfeed (Remove Full Crate)	25.0
a_6	Tool Change: Set Pieces Gripper	20.0
a_7	Tool Change: Set Crate Gripper	20.0
a_8	Wait Operation	20.0

Observation Space (S)

The rationale behind the definition of the observation space is to provide all necessary information to ensure the environment satisfies the Markov property of the MDP. It must provide a comprehensive “snapshot” of the system state at any given decision step. The entire state vector is normalized within the range $[0, 1]$ to facilitate neural network training and improve convergence stability.

1. Robot State (s_{robot}) This subset describes the kinematic and configuration status of the manipulator:

- **TCP Position (3 vars):** The normalized (x, y, z) coordinates of the Tool Center Point relative to the robot base.
- **Rail Position (1 var):** The normalized linear position of the robot on the slider (7th axis).
- **Gripper Status (3 vars):** A one hot encoding indicating the currently mounted tool:
 - $[1, 0, 0]$: Gripper for Pieces (Type A/B).
 - $[0, 1, 0]$: Gripper for Crates/Boxes.
 - $[0, 0, 1]$: No Gripper / Empty.

2. Line & Process State (s_{line}) This subset captures the logistics status, quantifying available resources and the progress of the order. All counts and capacities are normalized by their respective maximum limits:

- **Bin Levels (6 vars):** The fill level (percentage) of:
 - Type A pieces in the Type A Staging Area.
 - Type B pieces in the Type B Staging Area.
 - Small boxes in the Small Boxes Staging Area.
 - Full and closed small boxes in the Small Boxes Staging Area.
 - Crates in the Empty Crate Infeed.
 - Crates in the Full Crate Outfeed.
- **Small Box Status ($N \times 6$ vars):** For each potential small box location in the zone 3, we encode:

- Presence (1 var): Is a small box present?
- Type (3 vars): One hot encoding for *Type A*, *Type B*, or *Unassigned*.
- Capacity (1 var): Normalized number of available spots left in the box.
- Status (1 var): Indicates if the box is closed and ready for transport (0 = Open, 1 = Closed).
- **Crate Status (3×5 vars):** For each of the three crate positions (High/Input, Medium, Low/Output), we encode:
 - Presence (1 var): Indicates if a crate is physically present (0 = No, 1 = Yes).
 - Type (3 vars): One hot encoding for *Type A*, *Type B*, or *Mixed/Empty*.
 - Capacity (1 var): Normalized number of available spots left for small boxes within the crate.

3. Human State (s_{human}) This subset defines the operator’s spatial relationship to the robot and encodes their current task intent:

- **Relative Position (2 vars):** The vector (x, y) pointing from the robot’s Tool Center Point (TCP) to the human, normalized by the maximum workspace distance. This provides the agent with crucial directional awareness.
- **Absolute Distance (1 var):** The normalized scalar Euclidean distance between the robot and the human, denoted as $d(R, H)$.
- **Human Intent / Goal (8 vars):** A one hot encoding of the human’s current target zone or activity:
 - *Working Zones:* ZonePieceA, ZonePieceB, ZoneSmallBoxes, ZoneCrates.
 - *Picking Zones:* PickZone1, PickZone2.
 - *Passive States:* StandingStill, SafeZone.

Human Goal Encoding: It is important to note that these encoded goal states do not map strictly 1:1 to the internal FSM states defined earlier. This abstraction is intentional, designed to provide the RL agent with the minimal sufficient information required for decision making. For example, referencing the layout in Section 4.1, the storage locations for Type A pieces, Type B pieces, and small boxes are all clustered within the same general picking area. From the perspective of the robot’s control policy, the most critical information is predicting the human’s spatial trajectory to avoid interference, rather than

explicitly knowing which specific component the human is retrieving. By clustering these specific FSM tasks into a generalized spatial intent (e.g., `PickZone1`), we effectively reduce the dimensionality of the observation space without sacrificing the essential spatial awareness required for safe collaboration.

Table 5.3: State Space Observation Features and Dimensions

State Subset	Feature	Configuration / Real Values	Num. Vars
s_{robot}	TCP Position	(x, y, z) relative to carriage base dist max: 1.2m	3
	Rail Position	Normalized 7th axis joint position dist max: 3.0m	1
	Gripper Status	One hot encoding (Pieces, Crates, Empty)	3
s_{line}	Bin Levels	Max capacities: 6 (Type A/B), 4 (Boxes), 3 (Crates In/Out)	6
	Small Box Status	$N = 4$ locations $\times$ 6 variables per box	24
	Crate Status	3 rack positions $\times$ 5 variables per crate	15
s_{human}	Relative Position	Normalized (x, y) vector dist max: 3.2m	2
	Absolute Distance	Normalized scalar distance $d(R, H)$ dist max: 3.2m	1
	Human Intent	One hot encoding (8 specific zones/states)	8
Total State Space Dimension ($ S $):			63

Action Masking Strategy

To ensure the agent strictly adheres to the logical constraints, we employ Action Masking. At each decision step t , a binary validity mask $M(s_t)$ is computed based on the current state of the environment. This mask filters out impossible actions reducing the exploration space and accelerating convergence.

The validity logic is derived directly from the physical preconditions of the tasks, as defined in 4.1. An action a_i is valid ($M_i = 1$) only if all the following conditions are met:

1. **Component Transfer** (a_0, a_1): Transferring a raw component (Type A or B) into a small box requires three simultaneous conditions:
 - **Tool:** The "Pieces Gripper" must be currently mounted on the robot.
 - **Source:** The respective robot picking zones (Type A or B) must contain at

least one item.

- **Destination:** There must be a small box in the zone 3 with available capacity for that specific component type.
2. **Small Box to Crate Transfer (a_2, a_3):** Moving a small box to the crate in the gravity rack zone 6 requires:
 - **Tool:** The "Pieces Gripper" must be mounted.
 - **Source:** A completed small box of the requested type (A or B) must be present on the rack, completed means closed and full.
 - **Destination:** There must be a compatible crate in the working area (High, Medium, or Low position) with a free slot designated for that box type.
 3. **Crate Infeed Operation (a_4):** Moving an empty crate from zone 4 to the top of the gravity rack of zone 6:
 - **Tool:** The "Crate Gripper" must be mounted.
 - **Source:** An empty crate must be available at the input buffer.
 - **Destination:** The High Position (Gravity Rack Input) must be empty.
 4. **Crate Outfeed Operation (a_5):** Removing a finished crate requires:
 - **Tool:** The "Crate Gripper" must be mounted.
 - **Source:** The crate at the Low Position must be full.
 - **Destination:** The Output Buffer must have free space to receive the crate.
 5. **Tool Change Operations (a_6, a_7):** To prevent redundant moves, a tool change is only permitted if the target tool is **not** currently mounted.
 - a_6 (Set Pieces Gripper) is valid $\iff$ Current Tool $\neq$ Pieces Gripper.
 - a_7 (Set Crate Gripper) is valid $\iff$ Current Tool $\neq$ Crate Gripper.
 6. **Wait Operation (a_8):** The wait action is always valid.

Reward Function ($\mathcal{R}$)

The design of the reward function is a critical factor in guiding the RL agent toward the desired collaborative behavior. As defined in Section 4.3, the theoretical objective is to minimize the overall cycle time while simultaneously reducing the frequency of human

robot safety activations. To achieve this practically, we formulated a composite reward signal consisting of four distinct components:

$$r_t = r_{\text{efficiency}} + r_{\text{safety}} + r_{\text{progress}} + r_{\text{penalty}} \quad (5.1)$$

1. Efficiency (Time Penalty) To align with the objective of minimizing the makespan (C_{cycle}), the agent receives a negative reward proportional to the duration of the chosen action. This acts as a living cost, motivating the agent to complete tasks as efficiently as possible:

$$r_{\text{efficiency}} = w_{\text{time}} \cdot t_{\text{duration}} \quad (5.2)$$

where w_{time} is a constant scaling factor and t_{duration} is time elapsed between two decision steps.

2. Social Safety (Collision Penalty) This component of the reward function penalizes human robot safety activations:

$$r_{\text{safety}} = \begin{cases} p_{\text{collision}} & \text{if } d(R, H) < d_{\text{safe}} \text{ (Episode Termination)} \\ 0 & \text{otherwise} \end{cases} \quad (5.3)$$

3. Progress & Exploration (Dense Rewards) To prevent agent stagnation, we introduce dense positive reinforcement (r_{progress}) for each productive operational step. This guides the agent incrementally toward the primary objective:

- **Useful Action Bonus (r_{action}):** Awarded for every successful pick and place operation. Here, successful implies the action was completed without triggering a human robot safety intervention.
- **Crate Completion Bonus (r_{crate}):** Awarded upon the successful removal of a full crate from the workspace.
- **Job Completion (r_{job}):** A large terminal reward granted once the objective of removing three completed crates is fulfilled.

4. Logical Constraints (Action Shaping) Finally, specific penalties (r_{penalty}) are applied to discourage behaviors that are possible but inefficient:

- **Tool Change Penalty (p_{tool}):** Changing the gripper takes time but does not advance the task. This penalty stops the agent from constantly switching tools and

encourages it to batch similar operations together.

- **Excessive Waiting Penalty (r_{wait}):** This penalty represents the λ factor from Equation 4.1. It adds an extra cost to the *Wait* action, on top of the standard time penalty:

$$r_{\text{wait}} = w_{\text{wait}} \cdot t_{\text{wait}} \quad (5.4)$$

A lower penalty creates a more conservative policy, where the robot prefers to wait and keep its distance.

Table 5.4 summarizes the specific numerical values assigned to the reward components and penalties. These parameters were tuned to balance execution speed, safety, and goal completion.

Table 5.4: Reward Function Hyperparameters

Symbol	Description	Value
w_{time}	Efficiency time penalty weight	-0.01
$p_{\text{collision}}$	Safety violation penalty (Terminal)	-10.0
r_{action}	Useful action bonus (Pick/Place)	+0.75
r_{crate}	Crate completion bonus	+2.0
r_{job}	Global job completion bonus	+15.0
p_{tool}	Tool change penalty	-0.45
w_{wait}	Excessive wait penalty weight	-0.02

Decision Events

The decision making process is triggered by specific discrete events within the environment:

1. **Action Completion (Atomic Actions):** All physical manipulation tasks (e.g., *Pick Piece*, *Place Box*, *Tool Change*) are treated as atomic operations. Once the agent selects a task, the low level controller executes the full trajectory. The agent cannot interrupt or alter its decision mid execution. The next decision step is triggered only when the robot reports that the physical movement is completely finished.
2. **Conditional Interrupt (Wait Action):** Forcing the *Wait* action to stop the robot for a fixed amount of time would be inefficient. Instead, this action terminates the moment the system detects that the human has moved safely away from the robot.

This early stop lets the agent immediately look at the environment and make a new choice. This keeps the system fast and efficient: as soon as the human leaves the shared space, the robot knows the area is clear and can start a new task without standing idle.

Table 5.5: SSM Safety Zone Thresholds and Wait Action Interrupt Triggers

Safety Zone / Event	Distance Threshold	System Response / Trigger
Danger Zone	$d < 0.5$ m	Protective Stop
Reduced Speed Zone	$0.5 \leq d < 1.5$ m	Velocity Scaling
Nominal Speed Zone	$d \geq 1.5$ m	Full Speed Execution
Conditional Interrupt	$d > 1.7$ m	Wait Action Exit

5.2. Implementation and Training Framework

The practical implementation of the proposed solution relies on integrating two distinct software environments: a Python based Deep Learning framework and a high-fidelity industrial simulation platform built with C# and .NET.

5.2.1. Algorithm and Policy Implementation

The control policy is parameterized using a deep neural network, with parameters optimized via the Maskable PPO algorithm from the `sb3-contrib` package [22], an extension of the Stable Baselines3 library [21]. The implementation is written in Python, where the Markov Decision Process (MDP) environment is managed using the Gymnasium library.

5.2.2. The Simulation Environment (Tecnomatix)

Because training an RL agent involves thousands of trial and error interactions, training directly on a physical robot is impossible due to time and safety constraints. Consequently, a high fidelity digital twin developed in Siemens Tecnomatix is utilized. This environment serves as the ground truth for the MDP, providing the agent with state observations and rewards based on the simulated industrial process.

We selected *Siemens Tecnomatix Process Simulate* as the simulation platform. As an industry standard, it is capable of accurately simulating robotic operations, discrete automation, human movements, and collision physics.

While Tecnomatix supports highly detailed human avatars for ergonomic analysis, rendering them is computationally expensive and unnecessary for collision avoidance training. For the purpose of RL training, the human operator is modeled as a simplified kinematic volume (represented as a geometric primitive or cube in the visualizer). This abstraction captures the essential information spatial occupancy and the distance $d(R, H)$ while significantly reducing the computational load, allowing for faster training epochs. Despite this visual simplification, the human’s behavior remains complex, driven by the Probabilistic FSM described in Section 5.1.1.

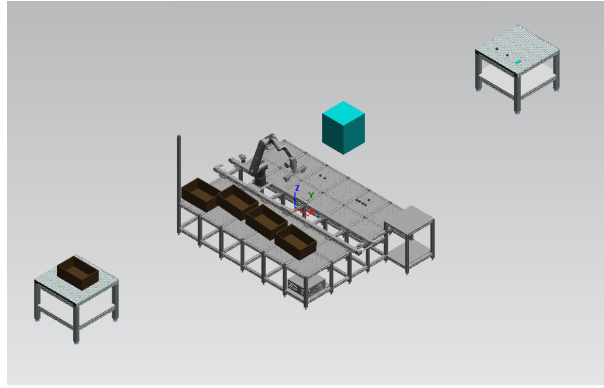


Figure 5.1: The Tecnomatix Process Simulate environment used for training. The cube represents the human operator

Implementing the logic of the environment required extensive use of C# to access the Tecnomatix APIs. This allowed us to take complete programmatic control over the simulator’s features.

Finally, we set up a TCP/IP communication bridge so the simulator and the RL agent could exchange information. At each step, the Python script sends the chosen action to the Tecnomatix environment, and the Tecnomatix simulator returns the resulting observation and reward.



Figure 5.2: Architecture of the communication bridge: The Python script (Agent) communicates with the C# Simulation Controller via TCP/IP.

6 | Results and Discussion

This chapter evaluates the performance and emergent behaviors of the trained Reinforcement Learning agent. Transitioning from the theoretical methodology established in the previous chapter, we first detail the final hyperparameter configuration used to train the Masked Proximal Policy Optimization (PPO) algorithm. We then analyze the agent’s learning curves to verify convergence and stability. Following this, a quantitative comparative analysis is presented, evaluating the proactive RL orchestrator against a reactive baseline to highlight the trade-offs between cycle time and overall process fluidity. Finally, we qualitatively evaluate the learned policy by observing the agent within the simulation. This demonstrates that the robot can successfully complete its production tasks while dynamically adapting its movements, through strategic wait actions and opposite side tasking, to ensure an uninterrupted, ergonomic workflow for the human operator.

6.1. Training Settings

Achieving a high-performing policy requires not only a well-defined Markov Decision Process (MDP) but also the careful selection of algorithmic hyperparameters. Table 6.1 details the configuration used to train the Masked PPO algorithm.

Because the action mask strictly bounds the agent’s exploration space by excluding invalid actions, the need for entropy driven exploration is reduced. Consequently, a low entropy coefficient (0.02) was selected.

Furthermore, the neural network architecture employs two hidden layers of 256 units each for both the actor (policy) and critic (value) networks. Even though the state and action spaces are small, the environment’s complexity and randomness require a network with sufficient capacity to accurately estimate the value function. We use the Tanh activation function, following the standard Stable Baselines3 setup.

Table 6.1: Masked PPO Hyperparameter Configuration

Parameter	Value	Description
Learning Rate (α)	2.5×10^{-4}	Step size for the optimizer to adjust model weights
Number of Steps	512	Environment steps collected per environment before an update
Batch Size	64	Number of samples used in each mini-batch during the update
Number of Epochs	10	Number of optimizer passes over the collected batch of data
Discount Factor (γ)	0.995	Importance weight of future rewards
Clip Range	0.2	Surrogate clipping threshold restricting deviation from the old policy
Entropy Coefficient	0.02	Weight of the entropy term to regulate exploration vs. exploitation
Total Timesteps	750×10^3	Total number of environment steps executed during the training run

6.1.1. Learning Curves

The learning curves illustrate the Masked PPO agent’s performance during training. It is important to note that while the plateauing of these curves indicates the agent has converged to a stable policy, it does not guarantee a global optimum; rather, it suggests a local optimum has been reached.

Figure 6.1 presents both the mean cumulative reward per episode and the mean length of those episodes. During the initial phases of training, the agent’s performance is penalized due to its inability to select actions that proactively avoid the human operator. This is evidenced by the combination of low cumulative rewards and short episode lengths; any robot action that breaches the predefined human safety distance results in immediate episode termination.

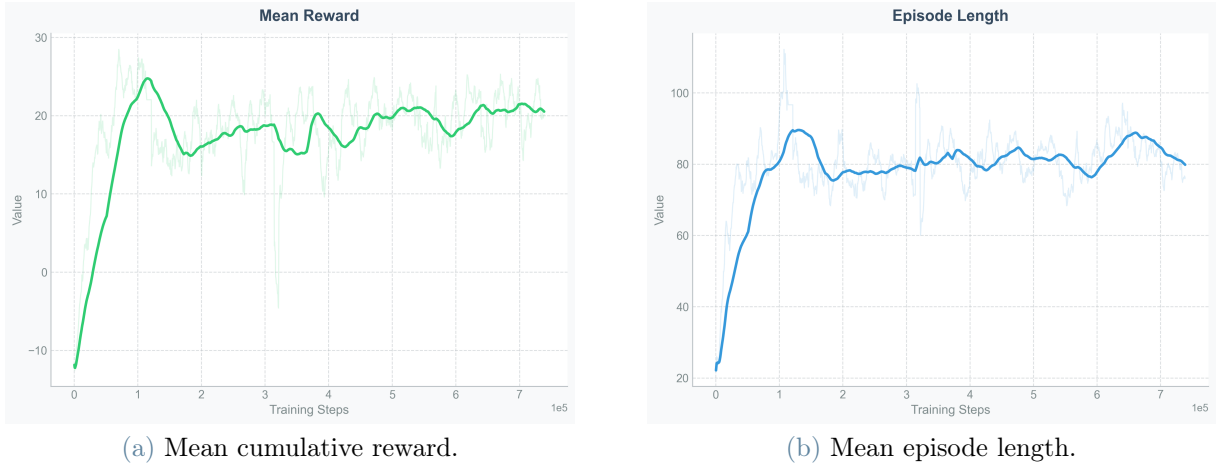


Figure 6.1: Training performance metrics displaying the mean cumulative reward (a) and mean episode length (b) over time.

6.2. Policy Evaluation and Qualitative Analysis of Emergent Behaviors

To evaluate if the learned policy works in practice, we visually inspected the agent’s behavior in the simulation. This confirmed that the numerical convergence in our learning curves actually translates to logical task execution. Specifically, we focused on how the robot adapts its movements when a human operator enters and exits the shared workspace.

6.2.1. End-to-End Task Execution

Figure 6.2 illustrates a chronological sequence of three simulation states, demonstrating the agent’s ability to successfully navigate a complete production cycle from start to finish.

The sequence begins with the initial workstation conditions (a). As the episode progresses to an intermediate stage (b), the environment populates with components, and the human operator is observed restocking the staging areas while the robot performs its operations. Finally, in (c), the robot successfully removes the third fully packed crate from the gravity rack, representing the terminal state and triggering a reset of the environment.

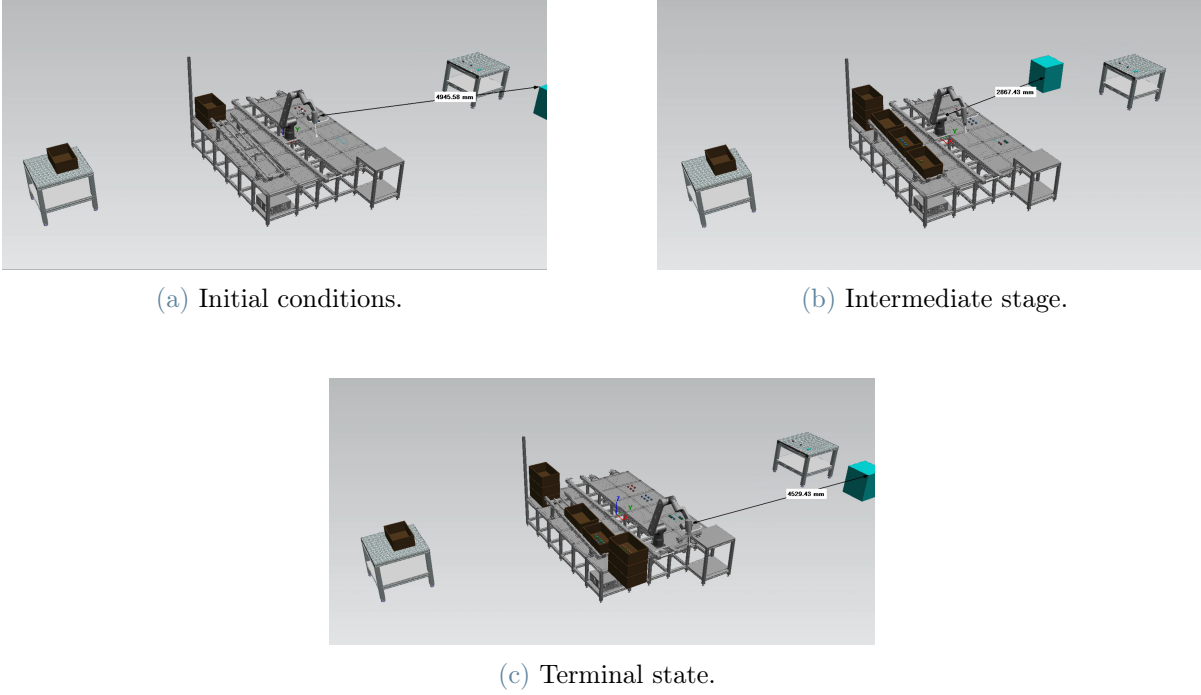


Figure 6.2: Chronological sequence of a complete production cycle. The images depict the transition from the (a) initial state, through an (b) intermediate assembly stage, to (c) the final reset condition.

6.2.2. Cycle Time and Efficiency

A core objective of this study was to minimize the overall cycle time (makespan) of the workstation. Because the Reinforcement Learning agent receives a constant negative penalty for every time step it takes to complete the order, it is mathematically driven to finish its tasks as quickly as possible.

The simulation shows that the trained policy effectively minimizes unnecessary downtime. Instead of simply freezing and waiting whenever the human is nearby, the robot leverages Opposite Side Tasking (Section 6.2.4) to find safe, productive operations on the opposite side of the table. This approach keeps both the human and the robot working simultaneously, cutting down on the idle time typically seen in standard setups where the machine merely stops and waits for the human to leave.

While it is difficult to mathematically guarantee that this cycle time is the absolute global minimum without a deterministic baseline solver for comparison, the stable, minimized episode lengths seen in the training curves (Figure 6.1) demonstrate that the agent converged on a highly optimized policy for this specific layout.

6.2.3. Proactive vs. Reactive: A Comparative Analysis

To quantitatively contextualize this efficiency and demonstrate the value of the proposed approach, the proactive RL orchestrator was evaluated against a greedy baseline. This baseline was trained without the proximity penalty in the reward function, effectively representing a traditional reactive system that prioritizes absolute spatial speed and relies entirely on the underlying Speed and Separation Monitoring (SSM) system to prevent collisions.

The comparison, summarized in Table 6.2, highlights the fundamental trade-off between execution speed and process fluidity in a highly constrained shared workspace. Because the robot and the human operator frequently need to access the exact same zones, the greedy baseline achieves a faster overall cycle time by aggressively taking the most direct spatial paths and simply waiting out the resulting safety stops. However, this comes at a severe cost to fluidity, triggering an average of nearly 20 disruptive SSM safety halts per episode.

In contrast, the proposed proactive RL policy intentionally trades a small amount of speed to virtually eliminate these sudden mid-air stops. Through its strategic use of the *Wait* action and *Opposite Side Tasking*, the proactive policy anticipates human movement and avoids conflicts before they occur. Consequently, while it requires slightly more time to complete the order (26.7 minutes versus 24.3 minutes), this trade-off is decisively justified. By preventing continuous deceleration and stop cycles, the orchestrator provides a much more predictable, ergonomic environment for the human operator.

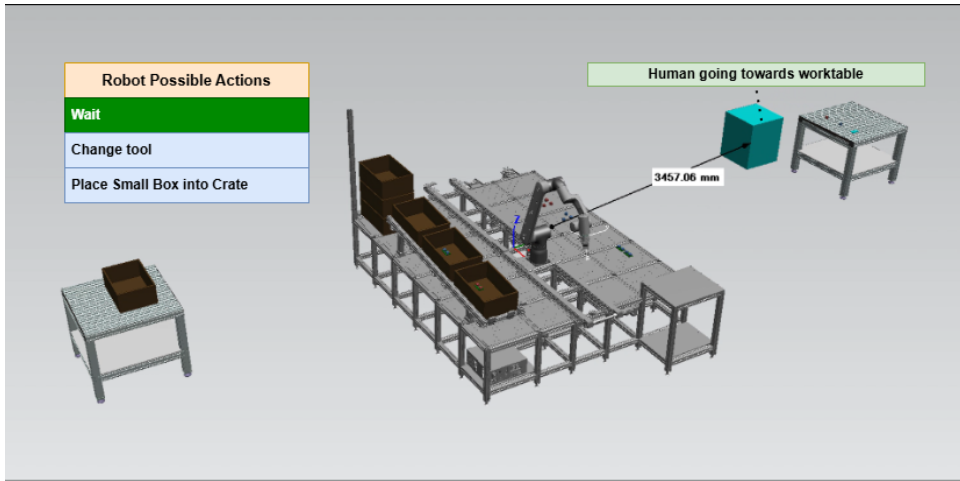
Table 6.2: Mean performance comparison between the proactive RL policy and the greedy baseline.

	Time (min)	Safety Halts
Greedy	24.3	19.8
Proactive	26.7	0.5

6.2.4. Human Avoidance and Spatial Coordination

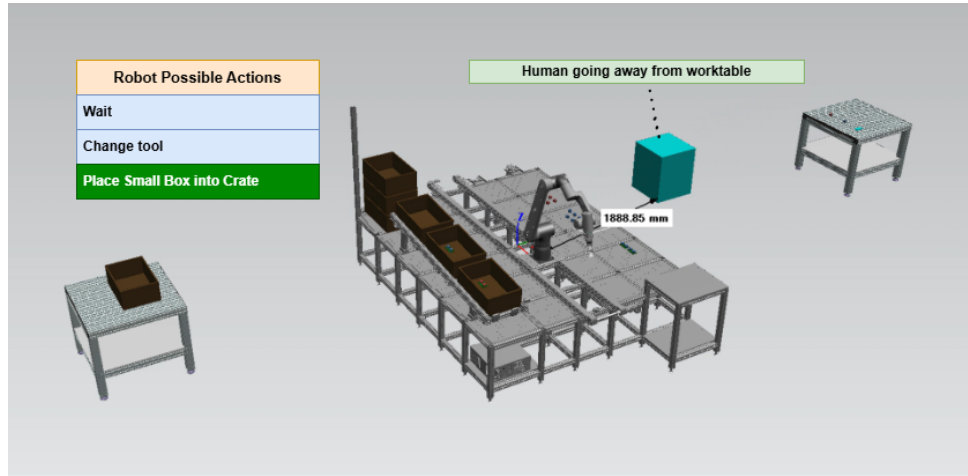
Another primary goal was for the agent to proactively avoid collisions by anticipating human movements and remaining clear of their operating zones. The agent successfully learned this behavior, avoiding safety stops by employing the following two strategies:

- **The Wait Action (a_8):** The robot pauses its physical movement, holding a safe position to allow the human operator to complete their task and clear the zone without triggering a safety stop. This behavior is illustrated in Figure 6.3. In (a), the human approaches the line, prompting the robot to select the wait action. In (b), the human has finished placing new Type A pieces and is moving away from the worktable, at which point the robot selects the action to move a small box into a crate. Finally, in (c), the robot performs the action while the human is safely outside the shared workspace.

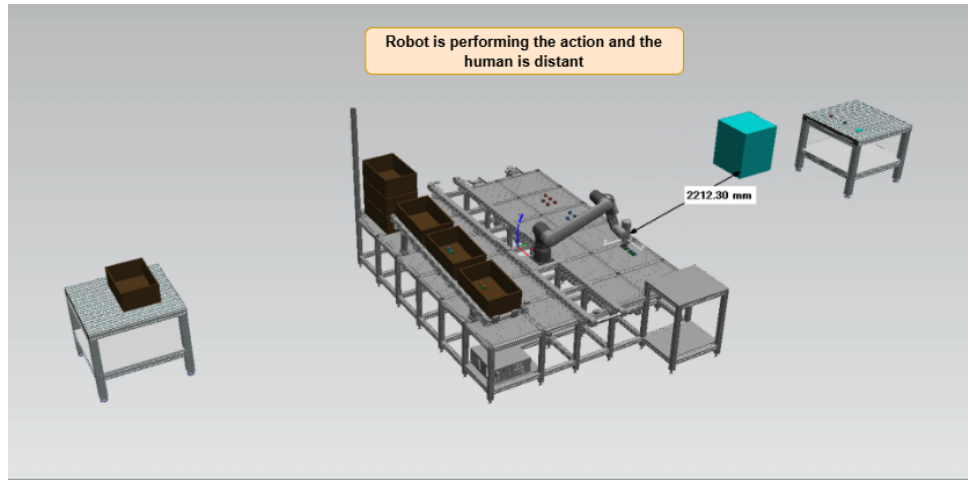


(a) Robot waits as human approaches.

Figure 6.3: Sequence demonstrating the Wait Action (a_8) (Part 1 of 2).



(b) Human retreats; robot selects action.



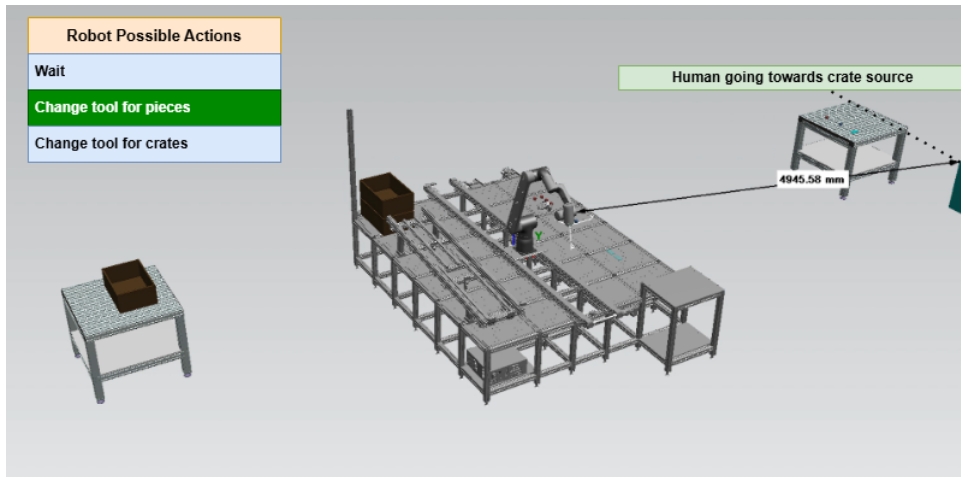
(c) Robot performs action safely.

Figure 6.3: Sequence demonstrating the Wait Action (a_8) (Continued). The robot anticipates the human's arrival (a), waits until the human clears the workspace (b), and then safely resumes its task (c).

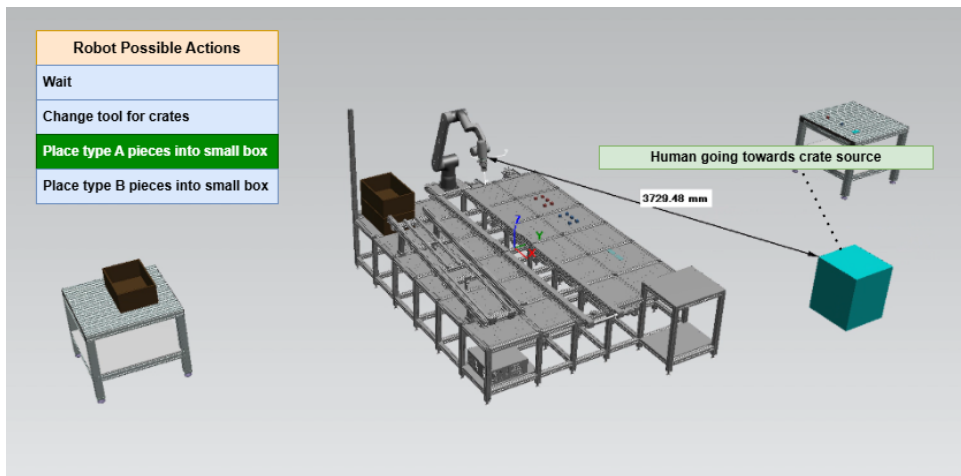
- **Opposite Side Tasking:** When the human moves toward a specific section of the assembly line, the agent proactively selects a task on the opposite side. We illustrate this behavior through two scenarios: one where the human approaches the left side of the worktable, and another where the human operates on the right side.

The first scenario is shown in Figure 6.4. As the human moves toward the left side of the workspace, the robot anticipates their destination and decides to equip the pieces gripper. This specific tool allows the robot to perform tasks exclusively on the right side of the worktable.

By selecting and executing an action on the opposite side, the robot maximizes the physical distance between itself and the human operator, ensuring continuous operation without triggering a safety stop.

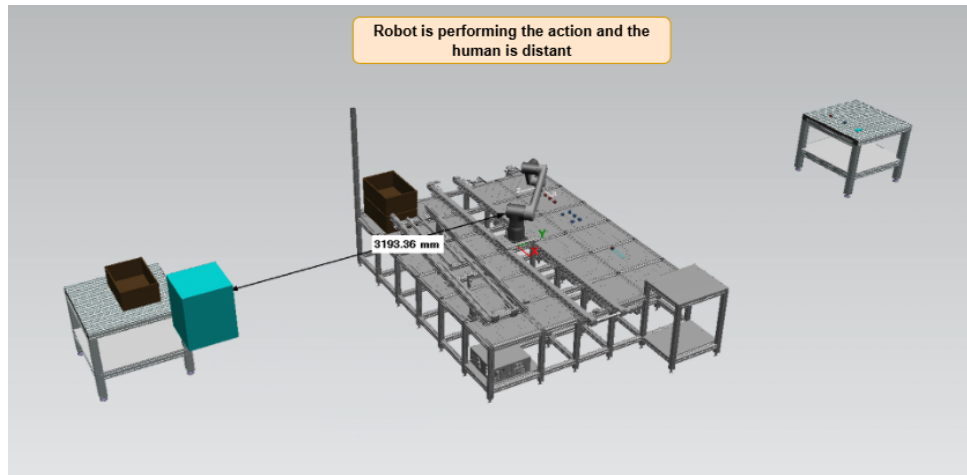


(a) Robot equips the pieces gripper as human approaches the left.



(b) Robot selects a task on the opposite (right) side of the table.

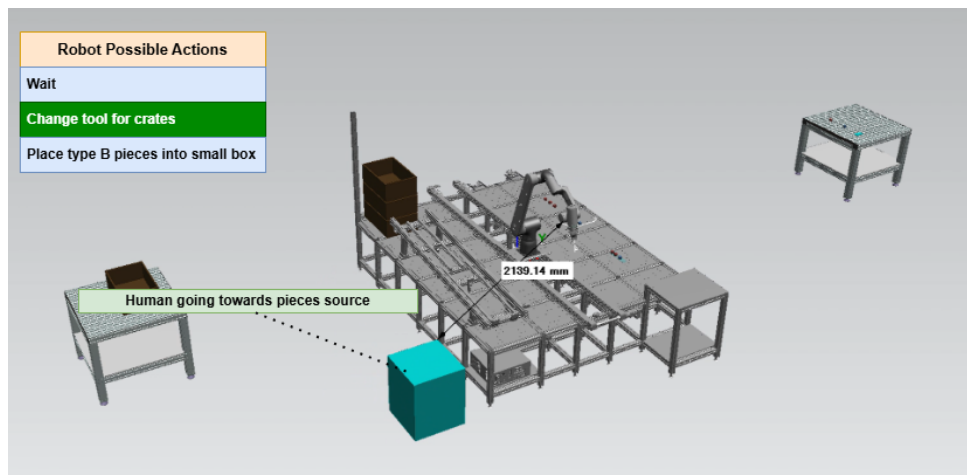
Figure 6.4: Task Zone Separation (Scenario 1) (Part 1 of 2).



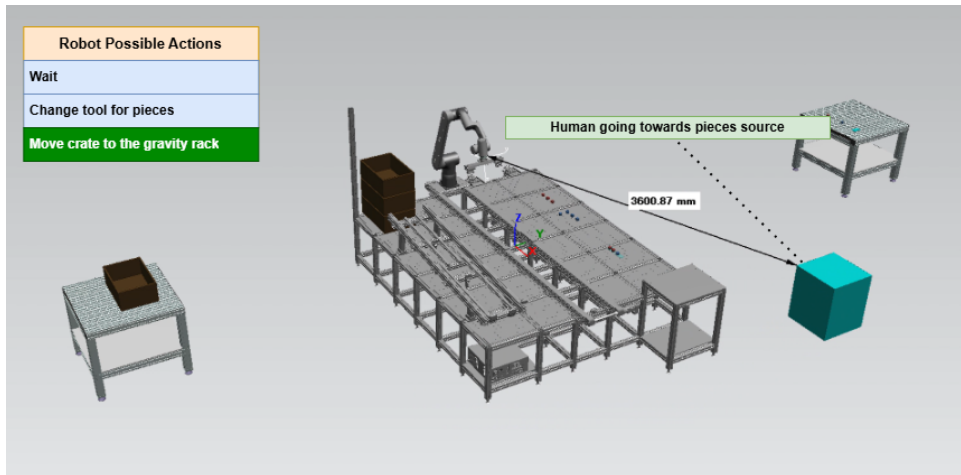
(c) Robot executes the action while maintaining a safe distance.

Figure 6.4: Task Zone Separation (Scenario 1) (Continued). The robot avoids the human's workspace on the left by retrieving a specific tool (a) that allows it to safely operate on the right side of the table (b, c).

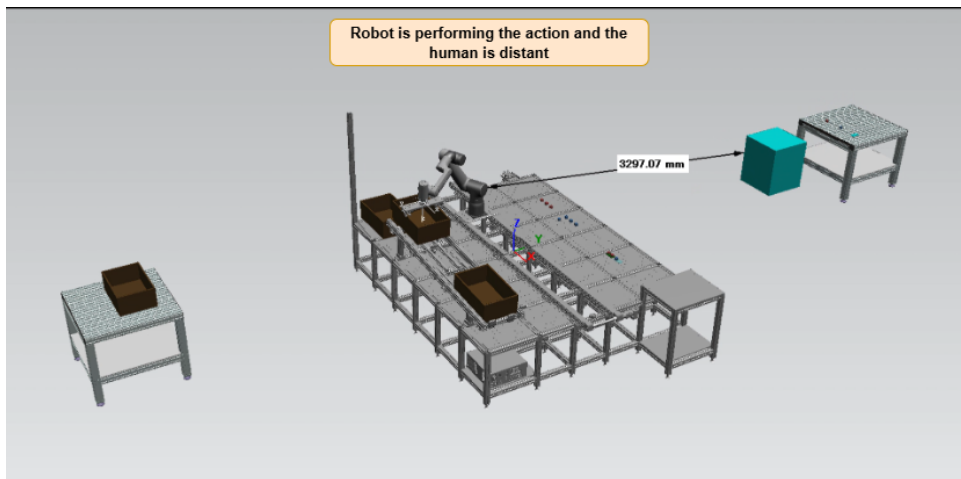
The second scenario is illustrated in Figure 6.5. Conversely, when the human approaches the right side of the workspace, the robot proactively changes its tool to the crate gripper. This decision enables the robot to focus its actions entirely on the left side of the table, maintaining a safe operating distance throughout the task execution.



(a) Robot equips the crate gripper as human approaches the right.



(b) Robot selects a task on the opposite (left) side of the table.



(c) Robot executes the action while maintaining a safe distance.

Figure 6.5: Task Zone Separation (Scenario 2). The robot adapts to the human approaching the right side by equipping the crate gripper (a) and performing its tasks safely on the left side (b, c).

7 | Conclusion and Future Work

7.1. Conclusion

This thesis addressed the problem of dynamic task scheduling and spatial coordination within a collaborative logistics workstation. This research is motivated by the growing interest in collaborative robotics, a field that enhances workstation flexibility and mitigates occupational health risks for human operators. To maximize user acceptance, a highly advantageous approach is to allow the human operator to work completely independently, requiring the robotic system to efficiently adapt to their decisions.

To achieve this, the operational environment was formalized as a Markov Decision Process (MDP), and a Reinforcement Learning orchestrator was developed. By leveraging the Masked Proximal Policy Optimization (PPO) algorithm, the robotic agent was trained within a high-fidelity Siemens Tecnomatix digital twin. The use of action masking ensured strict adherence to physical and logical production constraints, significantly accelerating the algorithm's convergence by not considering invalid actions.

The results demonstrated that the learned policy successfully navigated the trade-off between absolute execution speed and process fluidity. Rather than acting as a primary safety mechanism, the orchestrator functioned with a proactive awareness of the underlying Speed and Separation Monitoring (SSM) system. By intelligently utilizing the Wait action and employing Opposite Side Tasking, the RL agent anticipated human movement and dynamically adapted its schedule to avoid spatial conflicts before they occurred. While this proactive approach required a slight, intentional increase in overall cycle time compared to a greedy baseline, it drastically improved ergonomics by reducing safety halts from an average of nearly 20 per episode down to near zero.

7.2. Limitations and Future Work

While the simulation results prove the theoretical viability and efficiency of the RL-based orchestrator, translating these findings to a real world industrial environment introduces

several challenges. The following sections outline the primary limitations of the current study and propose directions for future research.

7.2.1. Human Intent Recognition and Goal Inference

A primary limitation of the current framework lies in the construction of the human state observation. In the simulation, the human operator is governed by a probabilistic Finite State Machine, and their exact current goal is explicitly provided to the Reinforcement Learning (RL) agent as a known observation vector.

In a practical physical deployment, the human's internal intent cannot be directly queried. Instead, this must be addressed within the perception layer (Figure 4.7) by utilizing the operator's past kinematic trajectories to infer their current goal. Consequently, the state input to the scheduling agent must be modified to incorporate probabilistic state estimates rather than the previously assumed determinism. For instance, when analyzing the human's movement through the corridor in the workspace, a single trajectory might ambiguously suggest either moving to the parts source to retrieve new objects or heading to the assembly line to close the lid of the small boxes.

Finally, to further enhance the robot's capabilities, the system should ideally predict not only the human's current goal but also their potential subsequent goals. Providing the agent with this predictive information would help mitigate the simulation to reality gap during real world deployment.

7.2.2. Experimental Validation and Sim-to-Real Transfer

Finally, the ultimate validation of this methodology requires experimental deployment on a physical workstation. While the Siemens Tecnomatix environment provided a robust and realistic digital twin for offline training, physical reality introduces unmodeled dynamics, including sensor noise, network latency, and, most problematically, the unpredictability of human behavior.

Future work must focus on the Sim to Real transfer of the learned policy. This will involve deploying the algorithm on the physical 7-DoF robotic setup to experimentally validate whether the collision avoidance strategies and the overall cycle time improvements observed in simulation hold true under real world industrial conditions.

Bibliography

- [1] D. Antonelli and G. Bruno. Dynamic distribution of assembly tasks in a collaborative workcell of humans and robots. *FME Transactions*, 47(4):723–730, 2019. doi: 10.5937/fmet1904723A. URL <https://doi.org/10.5937/fmet1904723A>.
- [2] G. Bruno and D. Antonelli. Dynamic task classification and assignment for the management of human-robot collaborative teams in workcells. *The International Journal of Advanced Manufacturing Technology*, 98(9):2415–2427, 2018. doi: 10.1007/s00170-018-2400-4. URL <https://doi.org/10.1007/s00170-018-2400-4>.
- [3] K. Darvish, B. Bruno, E. Simetti, F. Mastrogiovanni, and G. Casalino. Interleaved on-line task planning, simulation, task allocation and motion control for flexible human-robot cooperation. In *2018 27th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 58–65. IEEE, 2018. doi: 10.1109/ROMAN.2018.8525644. URL <https://doi.org/10.1109/ROMAN.2018.8525644>.
- [4] K. Darvish, E. Simetti, F. Mastrogiovanni, and G. Casalino. A hierarchical architecture for human-robot cooperation processes. *IEEE Transactions on Robotics*, 36(5):1541–1559, 2020. doi: 10.1109/TRO.2020.2995874. URL <https://doi.org/10.1109/TRO.2020.2995874>.
- [5] N. Dhanaraj, S. V. Narayan, S. Nikolaidis, and S. K. Gupta. Contingency-aware task assignment and scheduling for human-robot teams. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5765–5771. IEEE, 2023. doi: 10.1109/ICRA48891.2023.10161245.
- [6] G. Evangelou, N. Dimitropoulos, G. Michalos, and S. Makris. An approach for task and action planning in human-robot collaborative cells using ai. *Procedia CIRP*, 97:476–481, 2020. doi: 10.1016/j.procir.2020.08.006. URL <https://doi.org/10.1016/j.procir.2020.08.006>.
- [7] G. Giacomuzzo, M. Terreran, S. Jain, and D. Romeres. Decaf: a discrete-event based collaborative human-robot framework for furniture assembly. In *2024 IEEE/RSJ*

- International Conference on Intelligent Robots and Systems (IROS)*, pages 7085–7091, 2024. doi: 10.1109/IROS58592.2024.10802728.
- [8] L. Graesser and W. L. Keng. *Foundations of Deep Reinforcement Learning: Theory and Practice in Python*. Addison-Wesley Data & Analytics Series. Addison-Wesley Professional, 2019. ISBN 978-0135172384.
- [9] International Organization for Standardization. Iso/ts 15066:2016. robots and robotic devices — collaborative robots. Technical specification, International Organization for Standardization, Geneva, Switzerland, 2016.
- [10] A. Jain, S. Mehak, P. Long, J. D. Kelleher, M. Guilfoyle, and M. C. Leva. Evaluating safety and productivity relationship in human–robot collaboration. *Health and Technology*, 2022. doi: 10.3850/981973-0000-00-0_output. URL https://doi.org/10.3850/981973-0000-00-0_output.
- [11] L. Johannsmeier and S. Haddadin. A hierarchical human-robot interaction-planning framework for task allocation in collaborative industrial assembly processes. *IEEE Robotics and Automation Letters*, 2(1):41–48, 2017. doi: 10.1109/LRA.2016.2535907. URL <https://doi.org/10.1109/LRA.2016.2535907>.
- [12] W. Li, Y. Hu, Y. Zhou, and D. T. Pham. Safe human–robot collaboration for industrial settings: a survey. *Journal of Intelligent Manufacturing*, 35(5):1125–1147, 2024. doi: 10.1007/s10845-023-02159-4. URL <https://doi.org/10.1007/s10845-023-02159-4>.
- [13] Y. Y. Liao and K. Ryu. Genetic algorithm-based task allocation in multiple modes of human–robot collaboration systems with two cobots. *The International Journal of Advanced Manufacturing Technology*, 119(1):1045–1062, 2022. doi: 10.1007/s00170-022-08670-x. URL <https://doi.org/10.1007/s00170-022-08670-x>.
- [14] B. Matthias and T. Reisinger. Example application of iso/ts 15066 to a collaborative assembly scenario. In *Proceedings of ISR 2016: 47th International Symposium on Robotics*, pages 1–5, Munich, Germany, 2016. VDE.
- [15] C. Messeri, A. Bicchi, A. M. Zanchettin, and P. Rocco. A dynamic task allocation strategy to mitigate the human physical fatigue in collaborative robotics. *IEEE Robotics and Automation Letters*, 7(2):2178–2185, 2022. doi: 10.1109/LRA.2022.3143520. URL <https://doi.org/10.1109/LRA.2022.3143520>.
- [16] E. Montini, F. Daniele, L. Agbomemewa, N. Bonomi, P. Pedrazzoli, and A. Bettoni. Collaborative robotics: A survey from literature and practitioners perspec-

- tives. *Journal of Intelligent & Robotic Systems*, 110(117), 2024. doi: 10.1007/s10846-024-02141-z. URL <https://doi.org/10.1007/s10846-024-02141-z>.
- [17] R. Müller, M. Vette, and O. Mailahn. Process-oriented task assignment for assembly processes with human-robot interaction. *Procedia CIRP*, 44:210–215, 2016. doi: 10.1016/j.procir.2016.02.080. URL <https://doi.org/10.1016/j.procir.2016.02.080>.
- [18] N. Nikolakis, N. Kousi, G. Michalos, and S. Makris. Dynamic scheduling of shared human-robot manufacturing operations. *Procedia CIRP*, 72:9–14, 2018. doi: 10.1016/j.procir.2018.04.007. URL <https://doi.org/10.1016/j.procir.2018.04.007>.
- [19] C. Petzoldt, M. Harms, and M. Freitag. Review of task allocation for human-robot collaboration in assembly. *International Journal of Computer Integrated Manufacturing*, 36(11):1675–1715, 2023. doi: 10.1080/0951192X.2023.2204467. URL <https://doi.org/10.1080/0951192X.2023.2204467>.
- [20] M. L. Pinedo. *Scheduling: Theory, Algorithms, and Systems*. Springer Cham, 6th edition, 2022. ISBN 978-3-031-05921-6. doi: 10.1007/978-3-031-05921-6. URL <https://doi.org/10.1007/978-3-031-05921-6>.
- [21] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021. URL <http://jmlr.org/papers/v22/21-1311.html>.
- [22] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann. Stable baselines3 contrib documentation - ppo mask. https://sb3-contrib.readthedocs.io/en/master/modules/ppo_mask.html, 2021. Accessed: 2025-02-21.
- [23] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL <http://arxiv.org/abs/1707.06347>.
- [24] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018. URL <http://incompleteideas.net/book/the-book-2nd.html>.
- [25] S. Takata and T. Hirano. Human and robot allocation method for hybrid assembly systems. *CIRP Annals*, 60(1):9–12, 2011. doi: 10.1016/j.cirp.2011.03.128. URL <https://doi.org/10.1016/j.cirp.2011.03.128>.
- [26] C.-Y. Tang, C.-H. Liu, W.-K. Chen, and S. D. You. Implementing action mask in

- proximal policy optimization (ppo) algorithm. *ICT Express*, 6(3):200–203, 2020. doi: 10.1016/j.icte.2020.05.003. URL <https://doi.org/10.1016/j.icte.2020.05.003>.
- [27] L. Wang, S. Liu, H. Liu, and X. V. Wang. Overview of human-robot collaboration in manufacturing. In L. Wang, V. D. Majstorovic, D. Mourtzis, E. Carpanzano, G. Moroni, and L. M. Galantucci, editors, *Proceedings of 5th International Conference on the Industry 4.0 Model for Advanced Manufacturing*, pages 15–58, Cham, 2020. Springer International Publishing. ISBN 978-3-030-46212-3.
- [28] X. V. Wang, A. Seira, and L. Wang. Classification, personalised safety framework and strategy for human-robot collaboration. In *Proceedings of the 48th International Conference on Computers and Industrial Engineering (CIE48)*, Auckland, New Zealand, December 2018.

List of Figures

1.1	The collaborative logistics environment realized in the joint laboratory between Camozzi Automation S.p.A. and Politecnico di Milano.	3
2.1	The diagram illustrates the reallocation of tasks: the robot executes activities with low added value or those easily automated (blue/black blocks), allowing the operator to focus on high value, complex tasks (orange/red blocks).	6
3.1	The Agent-Environment Interaction Loop. At each time step t , the agent receives state S_t and reward R_t from the environment and executes action A_t , which triggers the transition to state S_{t+1} and reward R_{t+1}	18
4.1	The collaborative logistics environment realized in the joint laboratory between Camozzi Automation S.p.A. and Politecnico di Milano.	27
4.2	The two interchangeable tools used during the experiment.	27
4.3	The two types of pieces.	28
4.4	A small box.	28
4.5	The crate.	28
4.6	Top-down view of the workspace zones.	29
4.7	The proposed high level control architecture, linking perception, decision making, and execution.	35
5.1	The Tecnomatix Process Simulate environment used for training. The cube represents the human operator	49
5.2	Architecture of the communication bridge: The Python script (Agent) communicates with the C# Simulation Controller via TCP/IP.	49
6.1	Training performance metrics displaying the mean cumulative reward (a) and mean episode length (b) over time.	53

6.2	Chronological sequence of a complete production cycle. The images depict the transition from the (a) initial state, through an (b) intermediate assembly stage, to (c) the final reset condition.	54
6.3	Sequence demonstrating the Wait Action (a_8) (Part 1 of 2).	56
6.3	Sequence demonstrating the Wait Action (a_8) (Continued). The robot anticipates the human's arrival (a), waits until the human clears the workspace (b), and then safely resumes its task (c).	57
6.4	Task Zone Separation (Scenario 1) (Part 1 of 2).	58
6.4	Task Zone Separation (Scenario 1) (Continued). The robot avoids the human's workspace on the left by retrieving a specific tool (a) that allows it to safely operate on the right side of the table (b, c).	59
6.5	Task Zone Separation (Scenario 2). The robot adapts to the human approaching the right side by equipping the crate gripper (a) and performing its tasks safely on the left side (b, c).	60

List of Tables

2.1	Collaborative robots vs traditional robots (extension of [6])	7
2.2	Features of different human-robot relationships (Adapted from Wang et al. [28]).	9
2.3	Comparison of strengths and weaknesses between Humans and Collaborative Robots.	12
5.1	Summary of Human Agent Simulation Parameters	39
5.2	Nominal Execution Times for Robotic Actions	41
5.3	State Space Observation Features and Dimensions	44
5.4	Reward Function Hyperparameters	47
5.5	SSM Safety Zone Thresholds and Wait Action Interrupt Triggers	48
6.1	Masked PPO Hyperparameter Configuration	52
6.2	Mean performance comparison between the proactive RL policy and the greedy baseline.	55

