

A Hierarchical Stochastic Decision-Making Framework for Task Scheduling and Safety-Aware Human-Robot Collaboration

Muhammad Adel Yusuf

King Fahd University of Petroleum and Minerals

Ali Nasir

ali.nasir@kfupm.edu.sa

King Fahd University of Petroleum and Minerals

Research Article

Keywords: Human-robot collaboration, Task scheduling, Markov decision process, Partially observable Markov decision process, Safety-aware execution, Human fatigue, Industry 5.0

Posted Date: October 21st, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-11205936/v1>

License: © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Additional Declarations: No competing interests reported.

A Hierarchical Stochastic Decision-Making Framework for Task Scheduling and Safety-Aware Human-Robot Collaboration

Muhammad Adel Yusuf^{fa,*}, Ali Nasir^{a,b}

^aDepartment of Control and Instrumentation Engineering, King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia

^bInterdisciplinary Research Center for Intelligent Manufacturing and Robotics, King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia

Abstract

Task scheduling and safety-oriented execution are commonly treated as separate problems in human-robot collaboration (HRC), limiting rapid and coordinated responses to changing task and human conditions. This paper presents a hierarchical stochastic decision-making framework that shifts policy synthesis offline and supports fast online state-feedback decisions. An upper-level Markov decision process schedules and allocates tasks according to urgency, task nature, estimated execution-duration class, agent availability, and fatigue. A lower-level partially observable Markov decision process maintains a belief over operational human-state variables under a prescribed skeleton-based kinematic observation model and selects task-conditioned execution and human-robot separation actions. An intermediate task coordination module coordinates bidirectional information exchange between the scheduling and safety-oriented execution layers, enabling execution outcomes and updated agent states to inform subsequent scheduling decisions. Numerical evaluation demonstrates priority- and duration-class-aware allocation, fatigue-dependent recovery, admissible concurrent execution, and task-conditioned separation. The same precomputed policies are reused across a heterogeneous 12-task stream comprising four successive scheduling windows, with task identities and agent states preserved without window-specific

*Corresponding author

Email addresses: `muhammad.adel@ejust.edu.eg` (Muhammad Adel Yusuf), `ali.nasir@kfupm.edu.sa` (Ali Nasir)

retuning. Online scheduling requires indexed policy access rather than iterative optimization, with a measured mean lookup time of $3.082 \mu\text{s}$. The results establish the decision-layer and computational feasibility of coordinating fatigue-aware scheduling with belief-dependent, safety-oriented execution at the evaluated scale.

Keywords: Human-robot collaboration, Task scheduling, Markov decision process, Partially observable Markov decision process, Safety-aware execution, Human fatigue, Industry 5.0

1. Introduction

Industry 5.0 promotes human-centered, resilient, and sustainable manufacturing in which humans and robots share workspaces and combine complementary capabilities [1]. Human workers provide adaptability, dexterity, and contextual judgment [2], whereas collaborative robots offer precision, endurance, and repeatability [3]. Effective deployment therefore requires coordinated decisions across planning and execution: task assignments must respond promptly to changes in task and agent states, while robot behavior must adapt to uncertain human and interaction conditions. Achieving these objectives within a unified architecture, without introducing substantial on-line computational delay, remains an important challenge in human-robot collaboration (HRC).

At the planning level, mixed-integer formulations provide a systematic basis for optimizing task allocation, sequencing, precedence relations, execution times, and resource utilization [4]. Collaborative assembly-line models extend this perspective to production and resource constraints [5], while human-centered methods incorporate fatigue, workload, and ergonomic criteria into scheduling decisions [6, 7]. Some approaches improve responsiveness through event-triggered reoptimization when task requirements, execution parameters, or resource availability change [8]. Such mechanisms provide online adaptability, but each replanning event requires another iterative optimization process, whose response time depends on the problem size, solver configuration, and termination conditions.

At the execution level, monitoring and control methods adapt robot motion, speed, or separation according to observed human behavior and workspace conditions [9]. These mechanisms provide essential local responses to changes in human position and motion. Nevertheless, conservative reac-

tions to uncertain human behavior can reduce operational efficiency [10], motivating decision models that account explicitly for uncertainty. Moreover, execution-level safety mechanisms do not generally coordinate task priority, agent assignment, fatigue accumulation, and recovery across successive scheduling decisions. Conversely, scheduling formulations typically represent execution safety through predefined constraints or external protective mechanisms and do not continuously exchange task context and execution outcomes with an uncertainty-aware execution layer.

The benefits of adaptation at the planning and execution levels have been quantified independently in representative HRC studies. Lippi *et al.* [4] reported that online human-multi-robot reallocation reduced the average allocation cost by approximately 65% relative to static allocation over 50 simulation trials. At the execution level, Faroni *et al.* [11] reported reductions of approximately 19-25% in robot execution time and increases of 15-23% in the average safety-related speed-scaling factor relative to random task selection in physical pick-and-packaging experiments. Although these results were obtained for different problem formulations and cannot be compared directly, they indicate complementary benefits from adaptive task allocation and safety-aware execution. However, neither study investigates their closed-loop integration within a hierarchical architecture that communicates task context and agent-state information across the planning and execution levels.

Decision-theoretic methods provide a systematic basis for reasoning under partial observability. Partially observable Markov decision processes (POMDPs) have been used to represent partially observable collaborative tasks [12], latent human trust [13], hidden interaction states learned from demonstrations [14], and supervisory control under probabilistic requirements [15]. These studies demonstrate the value of belief-dependent decisions, but the representative literature does not fully address a closed-loop architecture in which a fatigue- and availability-aware scheduler supplies task semantics to a partially observable, safety-oriented execution layer and receives execution and agent-state feedback for subsequent scheduling decisions. This limitation motivates the integration of state-responsive scheduling, task-conditioned safety adaptation, and computationally efficient online policy evaluation.

This paper addresses this gap through a hierarchical architecture with complementary decision responsibilities. The upper-level Markov decision process (MDP) schedules and allocates tasks according to their priority, nature, estimated execution-duration class, and the availability and fatigue

states of the human and robot. The lower-level POMDP maintains a belief over operational engagement- and agitation-related variables using a prescribed skeleton-based kinematic observation model. This belief, together with the task context and the discrete human-robot separation state, supports the selection of safety-oriented execution actions. An intermediate Task Coordination Module (TCM) transfers task semantics and resource assignments to the execution layer, coordinates compatible concurrent assignments, tracks task progress, and returns completion, availability, and fatigue information to the scheduler. The resulting bidirectional interface allows execution outcomes and updated agent states to influence subsequent scheduling decisions.

A central feature of the proposed architecture is the separation of computationally intensive policy synthesis from deployed decision making. The scheduling and underlying execution-state policies are generated offline and stored. During online scheduling, the current state is mapped directly to the precomputed policy without invoking an iterative mixed-integer linear programming (MILP) or metaheuristic solver. At the execution level, the POMDP updates its belief from the current observation, and an Action Selection Heuristic (ASH) aggregates the recommendations of the precomputed execution-state policy to select an admissible action. Within the modeled state and action domains, this architecture removes iterative optimization from the online scheduling loop and enables rapid responses to changes in task status, availability, fatigue, recovery, and partially observable human conditions. The contribution therefore lies not in the individual use of MDPs or POMDPs, but in coordinating their scheduling and execution responsibilities through an explicit task- semantic and agent-state interface.

Contributions: The main contributions of this work are as follows:

- A hierarchical MDP-POMDP architecture that couples task-priority-, availability-, and fatigue-aware scheduling with belief-dependent, safety-oriented execution under partial observability.
- An event-driven TCM that maintains persistent task status and coordinates task semantics, resource compatibility, admissible concurrency, agent-state information, and execution feedback across the scheduling and execution layers.
- An offline-synthesis/online-evaluation strategy that removes iterative optimization from the deployed scheduling loop within the modeled

state and action domains. The measured mean scheduling-policy lookup time is $3.082 \mu\text{s}$; this value characterizes indexed policy access rather than complete sensing-to-action latency.

- A numerical closed-loop evaluation demonstrating priority-aware allocation, fatigue-dependent recovery, compatible concurrent execution, and task-conditioned separation under prescribed observation uncertainty. Policy reuse is further demonstrated across four heterogeneous scheduling windows comprising 12 persistent tasks, without window-specific policy retuning and with agent states and task identities preserved across window boundaries.

The remainder of this paper is organized as follows. Section 2 reviews productivity-oriented planning with human-centric factors, safety-aware execution and adaptive autonomy, and POMDP-based decision making under uncertainty, followed by a comparative positioning of the proposed framework. Section 3 formulates the coupled scheduling and safety-oriented execution problem. Section 4 presents the scheduling MDP and safety-aware execution POMDP. Section 5 describes the solution procedures, ASH, TCM, and computational complexity and scalability. Section 6 presents the numerical configuration, representative closed-loop evaluation, extended workload evaluation, and computational performance. Finally, Section 7 summarizes the principal findings and outlines directions for future work.

2. Related Work

Research relevant to the proposed framework spans three complementary areas: productivity-oriented planning, safety-aware execution, and decision-making under partial observability. These areas are analyzed below according to their optimization scope, representation of human factors, treatment of uncertainty, and connection between scheduling and execution.

2.1. Productivity-Oriented Planning With Human-Centric Factors

Mixed-integer formulations explicitly represent task assignments, precedence relations, execution times, and resource constraints [4]. Their principal advantage is the systematic evaluation of feasible schedules and, for tractable instances, the availability of optimality guarantees. Collaborative assembly-line models extend these formulations to production and budget constraints

[5]. However, their computational burden generally increases with the number of tasks, agents, and decision variables. Genetic approaches provide greater search flexibility for collaborative-system configuration [16], while particle-swarm methods generate approximate solutions for multi-robot allocation problems [17]. Thus, exact formulations provide stronger guarantees, whereas metaheuristics trade guaranteed optimality for computational flexibility.

Human-centric planning extends productivity objectives by incorporating operator condition. Dynamic fatigue estimates can trigger task reallocation to reduce overexertion [6], whereas personalized ergonomic criteria can modify task sequences according to worker characteristics [7]. The underlying assessments may be subjective, behavioral, biomechanical, or physiological. Subjective measures capture perceived workload or comfort but are intermittent and susceptible to reporting bias [18]. Behavioral measures are readily observable but may be confounded by task difficulty or system performance. Biomechanical measures characterize posture and physical exposure, whereas physiological signals support continuous monitoring but require calibration and reliable acquisition. Multimodal frameworks combine complementary measurements to improve workload estimation [19], while visual ergonomic indicators can be fused with cardiovascular signals for online strain detection [20]. Although these approaches strengthen human-aware planning, the resulting estimates are generally treated as observable indices or planning-level variables rather than beliefs used by an execution-level safety policy.

2.2. Safety-Aware Execution and Adaptive Autonomy

Execution-level safety commonly relies on continuous monitoring and reactive robot behavior [9]. Device-free localization estimates human presence without requiring wearable sensors [21], while vision-based control adapts robot motion using observed human activity [22]. Posture-aware optimization incorporates human configuration into motion decisions [23], and learning-based collision avoidance improves responsiveness to changing workspace conditions [24]. These methods provide direct spatial protection but primarily act on observable position, posture, or proximity information. They do not generally maintain beliefs over latent human conditions.

Adaptive-autonomy methods extend reactive safety by changing robot behavior according to interaction context. Task-progress information can support mode switching [25], human feedback can guide autonomy selection [26], and target prediction with fuzzy inference can support dynamic role

assignment [27]. These approaches improve local responsiveness but remain primarily focused on the active interaction rather than jointly managing task sequencing, agent availability, and fatigue over multiple decision epochs.

Recent safety-aware scheduling provides a closer connection between planning and execution. Faroni *et al.* learn the relationship between system state and safety-induced robot-speed reductions and use this relationship for online task selection [11]. This approach quantitatively demonstrates that safety-aware task selection can improve execution efficiency. However, it performs short-horizon selection using observable system variables and a learned slowdown model; it does not maintain a belief over latent human states or coordinate a separate fatigue-aware scheduler with a POMDP execution policy.

2.3. POMDP-Based Decision-Making Under Uncertainty

POMDPs maintain probability distributions over states that cannot be observed directly, providing a principled alternative to rules based only on current measurements [12]. Trust-aware decision-making represents human trust as a predefined latent state for long-horizon interaction [13]. In contrast, Bayesian nonparametric learning infers hidden interaction structures from demonstration data [14], reducing manual model specification but increasing dependence on representative training data. POMDP supervisory control combines belief-based decisions with probabilistic specifications and backup actions [15], thereby emphasizing formal performance requirements rather than task scheduling.

These approaches demonstrate the advantages of belief-state reasoning under sensing noise and latent human behavior. Nevertheless, their objectives concern interaction modeling, trust adaptation, hidden-state learning, or supervisory verification. They do not explicitly condition safety-oriented execution actions on task assignments from a fatigue- and availability-aware scheduler or return task-completion and agent-status information to such a scheduler.

2.4. Comparative Positioning

Across these research areas, the main limitation is not the absence of effective scheduling, human-factor assessment, uncertainty modeling, or safety control individually, but their limited coordination across decision levels.

Scheduling methods emphasize task allocation and productivity but commonly use observable human-state estimates and simplified safety representations. Safety-aware execution methods provide local adaptation but do not usually manage task-level fatigue and availability. POMDP methods handle latent-state uncertainty but are generally formulated independently of production scheduling. The proposed framework assigns these responsibilities to communicating components: the MDP manages task ordering, assignment, availability, and fatigue; the POMDP manages belief-based safety adaptation; and the TCM transfers task semantics to the execution level and returns task-completion and agent-status information.

Table 1 compares representative studies against eight explicitly defined capabilities, grouped into four functional pairs. The first pair characterizes scheduling behavior: *task planning/allocation* requires an explicit task-selection, sequencing, or assignment decision, while *online adaptation* requires that this decision be shown to change during execution in response to updated system information, rather than being fixed at the planning stage. The second pair concerns agent-state awareness during execution: *fatigue/workload dynamics* requires an explicit fatigue or workload variable within the decision model, and *mode switching/adjustable autonomy* requires a context-dependent change in robot behavior or autonomy level. The third pair addresses the treatment of uncertainty: *execution-level safety adaptation* requires an online mechanism that reduces interaction risk during task execution, *belief-state uncertainty* requires that decisions be computed over a probability distribution of partially observable states, and *latent human-state inference* requires the estimation of an unobservable human condition from available observations. The final criterion, *cross-layer scheduling-safety coordination*, requires an explicit and bidirectional connection between task-level planning and safety-oriented execution, rather than the independent specification of either. Each capability is classified as explicitly addressed (✓) when it is formulated or evaluated directly in the cited study, partially addressed (△) when it is treated indirectly or with limited scope, and not reported (-) when it falls outside the study’s stated formulation.

Table 1 shows a systematic division of capabilities: scheduling-oriented methods provide task allocation and online adaptation, with some incorporating fatigue or mode changes, but none maintain belief states for execution-level safety decisions, while the POMDP-based studies show the reverse pattern, representing uncertainty and latent human variables without connecting their execution policies to fatigue- and availability-aware scheduling.

Table 1: Capability-based comparison with representative HRC studies.

Capability	[25]	[4]	[6]	[11]	[14]	[13]	[15]	Proposed
Task planning / allocation	✓	✓	✓	✓	-	-	-	✓
Online task-plan adaptation	✓	✓	✓	✓	-	-	-	✓
Fatigue / workload dynamics	-	△	✓	-	-	-	-	✓
Mode switching / adjustable autonomy	✓	-	-	-	-	-	△	✓
Execution-level safety adaptation	✓	-	-	✓	-	-	✓	✓
Belief-state uncertainty	-	-	-	-	✓	✓	✓	✓
Latent human-state inference	-	-	△	-	✓	✓	△	✓
Cross-layer scheduling-safety coordination	△	-	-	△	-	-	-	✓

Legend: ✓ = explicitly addressed; △ = partially or indirectly addressed; - = not reported or outside the stated scope.

The partial ratings reflect capabilities present only in constrained form: fatigue and mode-switching terms that are workload- or verification-driven rather than dynamically modeled [4, 15], latent-state treatment based on behavioral proxies rather than formal inference [6, 15], and scheduling-safety links that are one-directional rather than closed-loop [11, 25]. The proposed framework satisfies the final criterion by construction rather than by degree: the MDP supplies task-level scheduling and agent-state management, the POMDP supplies belief-based safety adaptation, and the TCM provides the bidirectional exchange of task semantics and execution status between them — a closed-loop coordination absent, even partially, from every other representative study.

3. Problem Description

This work considers an industrial HRC system in which a human and a robot execute a set of tasks with different priorities, estimated execution-duration classes, and agent requirements. Each task may be assigned to the human, the robot, either agent, or both agents for joint execution. Scheduling decisions must account for the current availability and fatigue states of the agents, including the need for human rest or robot cooling and recovery. During task execution, the robot must additionally adapt its behavior and separation distance in response to partially observable human conditions and

kinematic observations.

The resulting decision problem has two interconnected components. The first is a productivity-oriented scheduling problem that determines the task sequence and the appropriate task-agent assignment. The second is a safety-oriented execution problem that determines how the assigned task should be performed under uncertainty regarding the human state. The proposed framework addresses these components through two stochastic decision-making levels connected by the TCM.

At the upper level, the scheduling MDP schedules and allocates tasks according to their priority, nature, estimated execution-duration class, and the current availability and fatigue states of the agents. At each decision epoch, the MDP selects a task-assignment, rest, recovery, transition, or idle action according to its state-dependent policy. The scheduling objective is to prioritize urgent tasks, promote appropriate task-agent assignments, reduce unnecessary idle time, and manage the workload and recovery requirements of both agents.

At the lower level, the execution POMDP governs safety-oriented task execution. The POMDP maintains a belief over latent human variables, including motivation/engagement and aggression/agitation, using probabilistic observations derived from skeletal positions and motion features. Based on the current belief state and the semantics of the scheduled task, the POMDP selects execution actions that regulate robot behavior and the human-robot separation distance. Larger separation margins are favored during solo-agent tasks to reduce unnecessary proximity and modeled interaction risk, whereas closer coordinated operation is permitted when joint task execution requires physical collaboration.

The TCM provides the operational interface between the two decision levels. When the MDP issues a task-assignment action, the TCM retrieves the corresponding task attributes-priority level, estimated execution-duration class, and task nature-and transfers them to the POMDP. During and after execution, the TCM monitors task completion and the availability and fatigue states of the agents. When an agent reaches its predefined fatigue or recovery threshold, the updated state is communicated to the MDP, which selects the appropriate rest, cooling, recovery, or subsequent assignment action from its state-dependent policy. This information exchange maintains consistency between task-level scheduling and safety-oriented execution without combining the two models into a single intractable state space.

As illustrated in Fig. 1, the Task Set module provides the task attributes

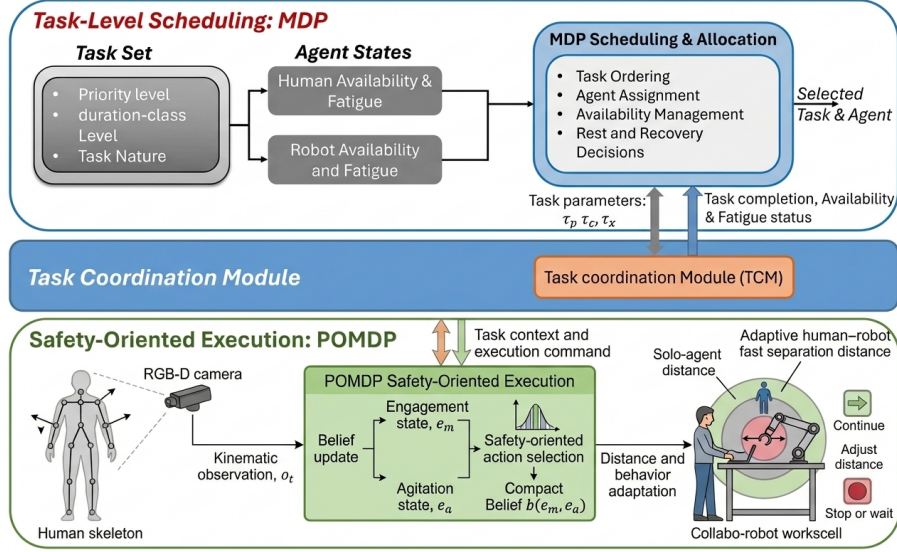


Figure 1: Hierarchical decision architecture for task scheduling and safety-oriented execution.

to the MDP scheduling layer. The TCM then transfers the selected task context (τ_p, τ_c, τ_x) , comprising the priority level, estimated execution-duration class, and task nature, respectively, to the POMDP, which selects safety-oriented execution actions using this context and the belief over the latent human state. Following execution, task-completion, availability, and fatigue-status information is returned through the TCM to support the next scheduling or recovery decision. The principal notation used throughout the framework is summarized in Table 2; symbols used only within individual transition or cost equations are defined locally.

Table 2: Principal notation used in the hierarchical framework.

Symbol	Definition
$\mathcal{T}$	Set of tasks considered by the scheduling framework.
N_T	Number of tasks in $\mathcal{T}$.
τ_j	State description of task slot j .
$\tau_{p,j}$	Priority level of task slot j .
$\tau_{c,j}$	Estimated execution-duration class of task slot j ; 0 denotes a nonpending task, and larger positive values denote longer-duration classes.
$\tau_{x,j}$	Nature of task slot j : robot-only, human-only, either-agent, or joint.
b_a, h_a	Availability states of the robot and human, respectively.
b_f, h_f	Fatigue or recovery states of the robot and human, respectively.
$\mathcal{S}_M, \mathcal{D}_M$	Scheduling-MDP state space and action set, respectively.
$T_M(s' s, u)$	Probability of reaching state s' after applying scheduling action u in state s .
$C_M(s, u)$	Immediate cost of selecting scheduling action u in state s .
γ	Discount factor used in sequential decision-making.
e_m, e_a	Latent human motivation/engagement and aggression/agitation states, respectively.
o_t	Discrete kinematic observation obtained at decision epoch t .
$b_t(s)$	Belief probability assigned to hidden state s at epoch t .
δ_t	Discrete human-robot separation-distance state at epoch t .
π_M, π_F^*	Scheduling-MDP policy and fully observable execution-state policy, respectively.
TCM	Module coordinating information exchange between the MDP and POMDP levels.

4. Hierarchical Decision Models

4.1. Scheduling MDP

The scheduling layer is formulated as a discounted Markov decision process (MDP) that determines task assignments and recovery actions from the current task and agent conditions, minimizing the expected cumulative discounted cost of task execution, incompatible assignments, agent idleness, and recovery requirements. Policy generation is performed offline, while online action selection uses indexed policy lookup; the associated computational requirements are reported in Section 5.5.

Formally,

$$\mathcal{M} = (\mathcal{S}_M, \mathcal{D}_M, T_M, C_M, \gamma), \quad (1)$$

where $\mathcal{S}_M$ is the scheduling state space, $\mathcal{D}_M$ the scheduling action space, $T_M(s' | s, u)$ the probability of reaching s' from s under action u , $C_M(s, u)$ the immediate cost, and $\gamma \in [0, 1)$ the discount factor. Although the formulation admits stochastic transitions, the implementation evaluated in this study uses a deterministic transition map, in which each state-action pair has a unique successor.

4.1.1. State Space

The state space contains task-status, agent-availability, and agent-fatigue variables. Let m denote the number of tasks represented simultaneously by the scheduler, treated as a bounded scheduling window rather than the total number of tasks in the system. The state space is defined as

$$\mathcal{S}_M = \{s_1^M, s_2^M, \dots, s_{N_M}^M\}, \quad (2a)$$

$$s_i^M = (\boldsymbol{\tau}_i, \mathbf{a}_i, \mathbf{f}_i), \quad i = 1, \dots, N_M, \quad (2b)$$

$$\boldsymbol{\tau}_i = (\tau_{i,1}, \tau_{i,2}, \dots, \tau_{i,m}), \quad (2c)$$

$$\tau_{i,j} = (\tau_{p_{i,j}}, \tau_{c_{i,j}}, \tau_{x_{i,j}}), \quad j = 1, \dots, m, \quad (2d)$$

$$\tau_{p_{i,j}} \in \{0, 1, \dots, \rho\}, \quad \tau_{c_{i,j}} \in \{0, 1, \dots, \sigma\}, \quad (2e)$$

$$\tau_{x_{i,j}} \in \{0, 1, 2, 3\}, \quad (2f)$$

$$\mathbf{a}_i = (b_{a_i}, h_{a_i}), \quad (2g)$$

$$b_{a_i}, h_{a_i} \in \{0, 1, 2\}, \quad (2h)$$

$$\mathbf{f}_i = (b_{f_i}, h_{f_i}), \quad (2i)$$

$$b_{f_i} \in \{0, 1, \dots, \epsilon_b\}, \quad h_{f_i} \in \{0, 1, \dots, \epsilon_h\}, \quad (2j)$$

$$N_M = [4(\rho + 1)(\sigma + 1)]^m n_{b_a} n_{h_a} (\epsilon_b + 1)(\epsilon_h + 1), \quad (2k)$$

where $n_{b_a} = n_{h_a} = 3$ are the cardinalities of the robot- and human-availability domains, and ρ, σ are assumed common across all m task slots for notational simplicity. Equation (2k) gives the cardinality of the fully enumerated state space and is consistent with the complexity analysis in Section 5.5.

For state s_i , the vector $\boldsymbol{\tau}_i$ contains m task slots indexed by j . In the evaluated formulation, each slot corresponds to one task in the current scheduling window; therefore, j identifies the task without being included as an additional state variable. The tuple $\tau_{i,j}$ contains the priority level $\tau_{p_{i,j}}$, estimated execution-duration class $\tau_{c_{i,j}}$, and task nature $\tau_{x_{i,j}}$ of task j . Positive priority levels range from 1 to ρ , with larger values indicating greater scheduling urgency. A value of $\tau_{p_{i,j}} = 0$ indicates that the task is no longer awaiting assignment and contributes no priority-related holding cost.

The duration variable is an ordinal representation of the relative processing-time requirement rather than a physical time measurement. Specifically, $\tau_{c_{i,j}} = 0$ indicates that the task is no longer pending, whereas the positive values $1, \dots, \sigma$ represent increasing execution-duration classes. For the two positive classes used in the numerical evaluation, $\tau_{c_{i,j}} = 1$ denotes a short-duration task and $\tau_{c_{i,j}} = 2$ denotes a long-duration task. The mapping from physical execution-time estimates to these classes would be calibrated for the relevant application.

Let $\hat{d}_j$ denote the estimated physical execution time of task j . The scheduling representation is obtained through an application-dependent quantization function

$$\tau_{c,j} = Q_d(\hat{d}_j) \in \{1, \dots, \sigma\}, \quad (3)$$

for pending tasks. The value $\tau_{c,j} = 0$ is reserved for tasks that are not pending and therefore does not belong to the set of active duration classes.

The task-nature variable specifies the eligible executing agent. Specifically, $\tau_{x_{i,j}} = 0$ denotes a robot-only task, $\tau_{x_{i,j}} = 1$ denotes a human-only task, $\tau_{x_{i,j}} = 2$ denotes a flexible task executable by either agent, and $\tau_{x_{i,j}} = 3$ denotes a joint human-robot task.

The availability vector $\mathbf{a}_i = (b_{a_i}, h_{a_i})$ describes the current robot and human status. For either agent, a value of 0 denotes task execution, 1 denotes availability for a new assignment, and 2 denotes a recovery state, corresponding to human rest or robot cooling/recharging.

The fatigue vector $\mathbf{f}_i = (b_{f_i}, h_{f_i})$ represents accumulated robot operational load and human fatigue. The corresponding domains contain $\epsilon_b + 1$ and $\epsilon_h + 1$ ordered levels, respectively, with increasing values indicating in-

creasing fatigue or operational load. Reaching the applicable maximum level requires a recovery action under the admissibility conditions of the model. Fatigue increases during committed task execution and decreases during rest or cooling according to the transition rules defined in Section 4.1.4. In the numerical model, one committed-work transition increases the corresponding fatigue state by one level, whereas one recovery transition decreases it by one level. These updates define normalized state dynamics and should not be interpreted as a calibrated one-to-one relationship between physical execution time and recovery time. Application-specific accumulation and recovery rates would require calibration using task and agent measurements.

Task-assignment actions identify both the selected task slot j and the assigned execution mode-robot, human, or joint-subject to the task-nature constraints. Rest, recall, idle, and recovery-transition actions operate directly on the corresponding agent states and therefore do not require a task-slot index. The specific action set used in the numerical study is defined in Section 4.1.3.

4.1.2. Human Fatigue State

The human-fatigue state introduced in the MDP state space belongs to the ordered set

$$h_f \in \mathcal{H}_f = \{0, 1, \dots, \epsilon_h\}, \quad (4)$$

where ϵ_h is the maximum discrete fatigue level and $L = \epsilon_h + 1$ is the selected number of fatigue levels. In the numerical study, the transition model updates h_f according to the sequence of committed-work and scheduled-recovery transitions. For physical deployment, physiological and ergonomic measurements would be processed and mapped to the same state representation at each scheduling epoch. The number of levels, fusion parameters, and classification thresholds would be selected according to the available sensing modalities and calibrated for the relevant workers and tasks. This sensing-to-state interface incorporates measurement-derived fatigue information without changing the scheduling formulation; sensor-system development and experimental validation remain outside the scope of the present decision-layer study.

Prior research indicates that fatigue-related information can be estimated by combining cardiovascular measurements, such as heart rate (HR), acquired using wearable devices based on photoplethysmography (PPG) or electrocardiography (ECG), with vision-based ergonomic indicators derived

from red-green-blue (RGB) or RGB-depth (RGB-D) skeletal observations [20]. A practical implementation would synchronize the measurements, attenuate motion artifacts and skeletal jitter, and normalize the signals relative to worker-specific baselines. An illustrative continuous fatigue indicator is

$$\hat{F}_h(t) = w_{HR}\widetilde{HR}(t) + w_e\tilde{r}_e(t - \Delta), \quad (5)$$

where $\widetilde{HR}(t)$ and $\tilde{r}_e(t)$ are the normalized cardiovascular and ergonomic indicators, respectively; Δ represents a possible delay between physical strain and cardiovascular response; and $w_{HR}, w_e \geq 0$, with $w_{HR} + w_e = 1$, are fusion weights.

The continuous estimate can be mapped to L discrete fatigue levels using an ordered set of thresholds:

$$h_f(t) = f_\ell \quad \text{if} \quad \theta_\ell < \hat{F}_h(t) \leq \theta_{\ell+1}, \quad \ell = 0, \dots, L-1, \quad (6)$$

where $\theta_0 = -\infty$, $\theta_L = +\infty$, and $\theta_1 < \dots < \theta_{L-1}$. A coarser resolution (small L) favors tractability and robustness to sensor noise, whereas a finer resolution (larger L) allows more graduated scheduling responses at increased state-space cost, per Eq. (45). The specific resolution adopted in the numerical case study is reported with the case-study parameters in Section 6.

Measurement uncertainty becomes decision-relevant when it changes the assigned fatigue level, potentially causing premature or delayed recovery decisions. Cardiovascular measurements may be affected by motion artifacts and non-fatigue influences, such as stress or transient exertion, whereas vision-based ergonomic indicators may be degraded by occlusion, viewpoint, and skeletal-estimation errors. Temporal filtering, threshold hysteresis, minimum dwell times, confidence-based rejection of unreliable samples, and conservative fallback strategies during prolonged signal loss can mitigate these effects. Evaluation of these sensor-level mechanisms remains an important component of future physical validation; the present results assess the scheduling response to the fatigue state supplied to the MDP.

4.1.3. Decision Variables

In the proposed hierarchical framework, the higher-level MDP governs task scheduling and centralized decision-making for the HRC system. The system controller operates through both the human and the robot, issuing instructions based on the optimal policy derived from the MDP. These instructions determine whether an agent should execute a task, enter a rest

period, or remain idle, depending on task characteristics, agent availability, fatigue levels, and other state variables.

Based on the defined state space, the set of available decision variables is given as:

$$\mathcal{D}_M = \{ rb_j, rh_j, rbh_j, rbR, rbW, rhR, rhW, idle, rbhR, rbhW, rbhT \} \quad (7)$$

Unlike elements in the state space, these decision elements represent discrete actions performed by the robot, the human, or both. Specifically, rb_j assigns task slot j to the robot, rh_j assigns task slot j to the human, and rbh_j assigns task slot j jointly to both agents, where $j \in \{1, \dots, m\}$. The variables rbR and rbW command the robot to enter recovery or return to work, respectively, while rhR and rhW perform the corresponding functions for the human. The actions $rbhR$ and $rbhW$ apply these commands to both agents. The action $idle$ leaves the system inactive, and $rbhT$ places both agents in the transition state while they await the next assignment.

4.1.4. State-Transition Mapping

The scheduling model uses a deterministic transition map. Given $s_t^M \in \mathcal{S}_M$ and $u_t \in \mathcal{D}_M$, the successor state is

$$s_{t+1}^M = f_M(s_t^M, u_t). \quad (8)$$

Accordingly, the transition kernel is

$$T_M(s^{M'} | s^M, u) = \begin{cases} 1, & s^{M'} = f_M(s^M, u), \\ 0, & \text{otherwise.} \end{cases} \quad (9)$$

The deterministic model is used for offline policy synthesis, while measurement-derived states may update the online policy input.

Task Transition. Let χ_R , χ_H , and χ_{HR} denote the realized robot, human, and joint execution modes, respectively. For the task-nature encoding adopted in this study,

$$(\chi_R, \chi_H, \chi_{HR}) = (0, 1, 3). \quad (10)$$

When task j is assigned, it is removed from the pending scheduling set and transferred to the execution layer:

$$(\tau_{p,j}^{t+1}, \tau_{c,j}^{t+1}, \tau_{x,j}^{t+1}) = \begin{cases} (0, 0, \chi_R), & u_t = rb_j, \\ (0, 0, \chi_H), & u_t = rh_j, \\ (0, 0, \chi_{HR}), & u_t = rbh_j, \\ (\tau_{p,j}^t, \tau_{c,j}^t, \tau_{x,j}^t), & \text{otherwise.} \end{cases} \quad (11)$$

Thus, $\tau_{p,j} = \tau_{c,j} = 0$ indicates that task j has been removed from the pending scheduling set following assignment, whereas $\tau_{x,j}$ records its realized execution mode. In this transition, $\tau_{c,j} = 0$ is a scheduling-status sentinel and does not indicate instantaneous physical completion. Before these task attributes are cleared from the scheduling state, the TCM transfers the selected task context to the execution layer, which subsequently tracks activation, progress, and completion.

Agent-State Transitions. Let H and R denote the human and robot, respectively, and define

$$(a_H, f_H, \epsilon_H) = (h_a, h_f, \epsilon_h), \quad (a_R, f_R, \epsilon_R) = (b_a, b_f, \epsilon_b), \quad (12)$$

where a_i , f_i , and ϵ_i are the availability, fatigue, and maximum fatigue level of agent $i \in \{H, R\}$.

For each agent, define the assignment, recovery, and recall action sets:

$$\mathcal{U}_{R,A} = \{rb_j, rbh_j \mid j = 1, \dots, m\}, \quad \mathcal{U}_{H,A} = \{rh_j, rbh_j \mid j = 1, \dots, m\}, \quad (13a)$$

$$\mathcal{U}_{R,R} = \{rbR, rbhR\}, \quad \mathcal{U}_{H,R} = \{rhR, rbhR\}, \quad (13b)$$

$$\mathcal{U}_{R,W} = \{rbW, rbhW\}, \quad \mathcal{U}_{H,W} = \{rhW, rbhW\}. \quad (13c)$$

Here, the subscripts A , R , and W identify assignment, recovery, and recall actions, respectively. Throughout the formulation, $\mathbb{I}[\mathcal{P}]$ denotes the indicator function, equal to one when proposition $\mathcal{P}$ is true and zero otherwise.

Using 0, 1, and 2 to represent committed, available, and recovering states, respectively, the availability transition is

$$a_i^{t+1} = \begin{cases} 0, & u_t \in \mathcal{U}_{i,A}, \\ 2, & u_t \in \mathcal{U}_{i,R}, \\ \mathbb{I}[a_i^t > 0], & u_t \in \mathcal{U}_{i,W}, \\ 1, & u_t = rbhT, \\ a_i^t, & \text{otherwise,} \end{cases} \quad i \in \{H, R\}. \quad (14)$$

For an arbitrary maximum fatigue level ϵ_i , define

$$[x]_0^{\epsilon_i} \triangleq \min\{\max\{x, 0\}, \epsilon_i\}. \quad (15)$$

The nominal fatigue transition is

$$f_i^{t+1} = \begin{cases} [f_i^t + 1]_0^{\epsilon_i}, & u_t \in \mathcal{U}_{i,A}, \\ [f_i^t - 1]_0^{\epsilon_i}, & u_t \in \mathcal{U}_{i,R}, \\ f_i^t, & u_t \in \mathcal{U}_{i,W}, \\ [f_i^t + \mathbb{I}[a_i^t = 0] - \mathbb{I}[a_i^t = 2]]_0^{\epsilon_i}, & \text{otherwise,} \end{cases} \quad i \in \{H, R\}. \quad (16)$$

Thus, fatigue increases during commitment, remains unchanged while the agent is available, and decreases during recovery. Equation (16) defines the nominal discrete dynamics used for policy synthesis. In physical deployment, measurement-derived human-fatigue estimates may update h_f through the sensing-to-state interface described in Section 4.1.2.

4.1.5. Cost Function

The scheduling cost balances four objectives: task urgency, execution-duration-class preference, fatigue-aware assignment, and robot preference for flexible tasks. It is defined as

$$C_M(s^M, u) = \alpha_1 c_1(s^M) + \alpha_2 c_2(s^M) + \alpha_3 c_3(s^M, u) + \alpha_4 c_4(s^M, u), \quad (17)$$

where $\alpha_1, \dots, \alpha_4 \geq 0$ are the corresponding weighting coefficients. Their numerical values are reported in Table 3. The relative weighting of task urgency and execution-duration class is determined by α_1/α_2 ; hence, the formulation represents a design tradeoff rather than a strict lexicographic ordering.

Let

$$\mathcal{J}(s^M) = \{j \in \{1, \dots, m\} \mid \tau_{c,j} > 0\} \quad (18)$$

To accommodate arbitrary finite discretizations, let $p^{(1)}, \dots, p^{(n_p)}$ denote the active priority levels ordered from lowest to highest urgency, and let $c^{(1)}, \dots, c^{(n_c)}$ denote the positive execution-duration classes ordered from shortest to longest. Similarly, $f_i^{(0)}, \dots, f_i^{(n_{f_i}-1)}$ are the fatigue levels of agent i , ordered from lowest to highest. Here, n_p , n_c , and n_{f_i} are the numbers of active priority classes, positive duration classes, and agent-fatigue levels, respectively.

Their normalized scores are

$$\phi_p(p^{(q)}) = \frac{q}{n_p}, \quad q = 1, \dots, n_p, \quad (19a)$$

$$\phi_c(c^{(q)}) = \frac{n_c - q + 1}{n_c}, \quad q = 1, \dots, n_c, \quad (19b)$$

$$\phi_{f_i}(f_i^{(q)}) = \frac{q}{n_{f_i} - 1}, \quad q = 0, \dots, n_{f_i} - 1. \quad (19c)$$

The priority score of an inactive task is defined as zero. These rank-based mappings make the cost independent of the numerical labels assigned to the discrete states.

Task-Priority Cost. The priority component is

$$c_1(s^M) = \sum_{j \in \mathcal{J}(s^M)} \phi_p(\tau_{p,j}). \quad (20)$$

A more urgent pending task produces a larger holding cost. Assigning that task removes its contribution from subsequent scheduling states and thereby promotes its earlier selection.

Duration-Class Holding Cost. The duration component is

$$c_2(s^M) = \sum_{j \in \mathcal{J}(s^M)} \phi_c(\tau_{c,j}). \quad (21)$$

Because $\phi_c(\cdot)$ decreases as the execution-duration class increases, a pending task in a shorter-duration class produces a larger holding cost. Since assigning the task removes this contribution from subsequent scheduling states, this construction promotes earlier selection of shorter tasks when the effects of urgency and other scheduling criteria are comparable. The relative influence of this preference is controlled by α_2 .

Fatigue-Aware Assignment Cost. The fatigue component is

$$\begin{aligned} c_3(s^M, u) = \sum_{j \in \mathcal{J}(s^M)} & \left(\mathbb{I}[u = rb_j] \phi_{f_R}(b_f) \right. \\ & + \mathbb{I}[u = rh_j] \phi_{f_H}(h_f) \\ & \left. + \mathbb{I}[u = rbh_j] [\phi_{f_R}(b_f) + \phi_{f_H}(h_f)] \right). \end{aligned} \quad (22)$$

This term discourages additional allocation to agents with greater fatigue and favors less-fatigued feasible assignments when the remaining scheduling conditions are comparable.

Robot-Preference Cost. Let

$$\mathcal{A}(\tau_{x,j}) \subseteq 2^{\{H,R\}} \quad (23)$$

denote the resource combinations permitted by the nature of task j , where $2^{\{H,R\}}$ is the power set of the human-robot resource set. For the task-nature encoding adopted in this study,

$$\begin{aligned} \mathcal{A}(0) &= \{\{R\}\}, & \mathcal{A}(1) &= \{\{H\}\}, \\ \mathcal{A}(2) &= \{\{R\}, \{H\}\}, & \mathcal{A}(3) &= \{\{H, R\}\}. \end{aligned} \quad (24)$$

The robot-preference component is

$$\begin{aligned} c_4(s^M, u) &= \sum_{j \in \mathcal{J}(s^M)} \mathbb{I}[u = rh_j] \mathbb{I}[\{H\} \in \mathcal{A}(\tau_{x,j})] \\ &\quad \times \mathbb{I}[\{R\} \in \mathcal{A}(\tau_{x,j})] \mathbb{I}[b_a = 1] \mathbb{I}[b_f < \epsilon_b]. \end{aligned} \quad (25)$$

This term expresses a workload-allocation preference rather than a feasibility requirement. It penalizes human assignment only when the task also permits robot execution and the robot is available below its fatigue limit. It therefore reduces avoidable human task exposure without assuming that robot execution is universally preferable.

Assignment Feasibility. Let $\mathcal{D}_M(s^M) \subseteq \mathcal{D}_M$ denote the actions admissible in state s^M , and define the resource requirement of each assignment action as

$$\mathcal{R}(rb_j) = \{R\}, \quad \mathcal{R}(rh_j) = \{H\}, \quad \mathcal{R}(rbh_j) = \{H, R\}. \quad (26)$$

An assignment action for task j belongs to $\mathcal{D}_M(s^M)$ only if

$$\mathcal{R}(u) \in \mathcal{A}(\tau_{x,j}), \quad a_i = 1, \quad f_i < \epsilon_i, \quad \forall i \in \mathcal{R}(u). \quad (27)$$

Thus, an assignment is permitted only when it is compatible with the task nature and every required agent is available below its maximum fatigue level. Recovery, recall, idle, and availability-transition actions are admitted according to the agent-state conditions specified by the transition model.

4.2. Safety-Aware Execution POMDP

The lower-level execution model is adopted from our previous work [28], where its state space, decision variables, transition probabilities, cost function, and numerical evaluation are presented in detail. The internal POMDP formulation is retained without modification in the present study. Its new role is to operate as the safety-execution layer of the hierarchical architecture, receiving the selected task context from the scheduling MDP through the TCM and returning task-execution information to the coordination process.

The POMDP state is summarized as

$$\mathbf{s}_P = (\boldsymbol{\tau}, \delta, \mathbf{e}, \mathbf{h}, \mathbf{b}), \quad (28)$$

where $\boldsymbol{\tau} = (\tau_p, \tau_c, \tau_x, \tau_y)$ describes the assigned task through its priority level, execution-duration class, task nature, and commitment status, respectively. The variable δ is the discrete human-robot separation state, and $\mathbf{e} = (e_m, e_a)$ contains the partially observable human-related variables. The vectors $\mathbf{h} = (h_p, h_c)$ and $\mathbf{b} = (b_p, b_c)$ describe the current human and robot activities, with h_c and b_c representing discrete remaining-work states after commitment. These activity-progress states are abstract model levels rather than measurements of physical elapsed time. In the hierarchical framework, the task attributes (τ_p, τ_c, τ_x) are supplied by the TCM, whereas the POMDP selects the corresponding safety-oriented execution action.

4.2.1. Latent Behavioral State Space

The POMDP represents motivation-related engagement e_m and aggression-related agitation e_a as operational variables used to condition execution and safety-distance decisions. They are not interpreted as direct psychological diagnoses. Their ordered domains are

$$\begin{aligned} e_m &\in \mathcal{E}_m = \{0, 1, \dots, \epsilon_m\}, \\ e_a &\in \mathcal{E}_a = \{0, 1, \dots, \epsilon_a\}, \end{aligned} \quad (29)$$

where ϵ_m and ϵ_a denote the maximum represented levels. Hence, the joint latent-state space is

$$\mathcal{E} = \mathcal{E}_m \times \mathcal{E}_a, \quad |\mathcal{E}| = (\epsilon_m + 1)(\epsilon_a + 1). \quad (30)$$

Increasing indices represent increasing behavioral intensity, and the POMDP maintains a belief distribution over $\mathcal{E}$ because these variables are not directly observable. The latent-state resolution used in the numerical evaluation is specified in Section 6.1.

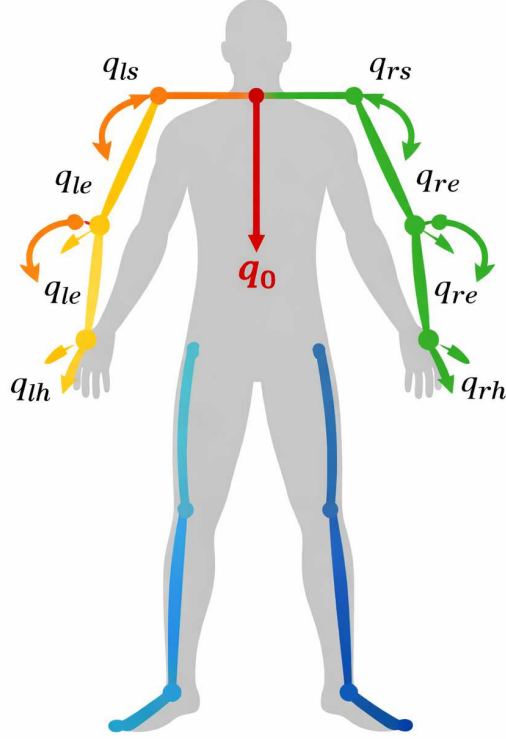


Figure 2: Upper-body joints used by the POMDP observation interface.

4.2.2. Skeleton-Based Observation Interface

The POMDP uses human skeletal position and motion variables as probabilistic evidence for updating the belief over $\mathcal{E}$. Body landmarks and skeletal posture have been used to represent bodily expressions of affect [29]. Graph-based representations preserve spatial relationships among the selected joints [30], while temporal skeletal features provide motion information that is unavailable from static posture alone [31]. Rapid and forceful bodily movements have also been associated with anger-related expression [32]. These findings support the use of skeletal kinematics as uncertain behavioral evidence; they do not imply deterministic identification of engagement or agitation.

As illustrated in Fig. 2, the observation interface uses a torso reference q_0 together with the left and right shoulders, elbows, and hands.

For a physical implementation, let $\mathbf{p}_j(t) \in \mathbb{R}^3$ denote the measured position of joint j and let $\mathbf{p}_0(t)$ denote the torso reference. The torso-relative

position and corresponding velocity are obtained as

$$\begin{aligned}\tilde{\mathbf{p}}_j(t) &= \frac{\mathbf{p}_j(t) - \mathbf{p}_0(t)}{\ell(t)}, \\ \mathbf{v}_j(t) &= \frac{\tilde{\mathbf{p}}_j(t) - \tilde{\mathbf{p}}_j(t-1)}{\Delta t},\end{aligned}\tag{31}$$

where $\ell(t)$ is a body-scale reference, such as shoulder width. Torso centering and scale normalization reduce variations caused by the worker’s location, body size, and camera distance. Temporal filtering can be applied before differentiation to reduce landmark jitter.

For M selected joints, motion activation over a window of W epochs can be represented by

$$I_v(t) = \frac{1}{WM} \sum_{\kappa=t-W+1}^t \sum_{j=1}^M \|\mathbf{v}_j(\kappa)\|,\tag{32}$$

where M is the number of selected joints. Following normalization, $\hat{I}_v(t)$ is mapped to the finite observation alphabet $\mathcal{O} = \{o_1, \dots, o_{L_o}\}$ according to

$$o_t = o_k \quad \text{if} \quad \theta_{k-1} < \hat{I}_v(t) \leq \theta_k, \quad k = 1, \dots, L_o,\tag{33}$$

where $0 = \theta_0 < \theta_1 < \dots < \theta_{L_o} = 1$.

The observation resolution L_o and the associated thresholds are application-dependent and can be selected according to the required behavioral resolution, available sensing accuracy, and computational constraints. Each observation provides probabilistic evidence about the joint latent state; it does not uniquely determine either e_m or e_a . This uncertainty is represented through the observation likelihood $Z_P(o_k \mid e_m, e_a)$ and the resulting belief distribution over $\mathcal{E}$.

4.2.3. Observation Likelihood and Calibration

The observation likelihood

$$Z_P(o_k \mid e_m, e_a) = \Pr(o_t = o_k \mid e_m, e_a)\tag{34}$$

relates each discrete motion observation to the joint latent state. In the present numerical study, the observation-conditioned belief distributions used by ASH are prescribed in (51) rather than estimated from human-participant data. The reported results therefore evaluate the hierarchical

decision process and belief-dependent action selection under the specified observation uncertainty; they do not evaluate the accuracy of a physical behavioral-state classifier.

For physical deployment, the likelihood values can be calibrated from synchronized skeletal measurements and behavioral reference labels. Given labeled samples, a smoothed empirical estimate is

$$\hat{Z}_P(o_k | e_m, e_a) = \frac{N_{(e_m, e_a), k} + \lambda}{N_{(e_m, e_a)} + \lambda L_o}, \quad (35)$$

where $N_{(e_m, e_a), k}$ is the number of occurrences of observation o_k under latent state (e_m, e_a) , $N_{(e_m, e_a)}$ is the number of samples associated with that state, and $\lambda > 0$ is a smoothing parameter. To support transfer across industrial settings, the observation-model parameters should be calibrated using representative data that capture task-dependent motion, occlusion, viewpoint variation, and inter-worker variability.

5. Solution Methodology

This section presents the decision-making methodology employed in the proposed framework, integrating the MDP and the POMDP for ensuring efficient and safe HRC in industrial setting.

5.1. MDP

As defined in the previous section, the scheduling MDP is represented by the five-tuple

$$\mathcal{M} = (\mathcal{S}_M, \mathcal{D}_M, T_M, C_M, \gamma), \quad (36)$$

where the symbols retain the definitions introduced in Eq. (1).

The optimal scheduling value function is computed using value iteration, which repeatedly applies Bellman's equation:

$$V_M^{(k+1)}(s) = \min_{u \in \mathcal{D}_M} \left\{ C_M(s, u) + \gamma \sum_{s' \in \mathcal{S}_M} T_M(s' | s, u) V_M^{(k)}(s') \right\} \quad (37)$$

The iteration continues until the convergence condition is satisfied:

$$\|V_M^{(k+1)} - V_M^{(k)}\|_2 < \eta \quad (38)$$

where η is a small positive threshold ensuring numerical stability.

After convergence, the resulting value function $V_M^*(s)$ is used to derive the scheduling policy:

$$\pi_M^*(s) = \arg \min_{u \in \mathcal{D}_M} \left\{ C_M(s, u) + \gamma \sum_{s' \in \mathcal{S}_M} T_M(s' | s, u) V_M^*(s') \right\} \quad (39)$$

The value-iteration algorithm thus computes $V_M^*(s)$ and the corresponding policy $\pi_M^*(s)$, mapping each scheduling state to a cost-minimizing action.

5.2. POMDP

While the MDP framework effectively handles systems with fully observable states, it is insufficient when certain system states—such as human emotional or cognitive conditions—are hidden. In order to model this partial observability, the framework is extended into a POMDP. A POMDP combines the MDP formulation with a hidden Markov model (HMM) that links unobservable states to measurable observations.

A POMDP introduces an observation space and an observation model to infer hidden states probabilistically. The process begins with an initial belief state $b_0(s)$, which represents a probability distribution over the execution states. Based on the current belief, an execution action $u_t \in \mathcal{D}_P$ is selected using the ASH rule. After executing the action, the system receives observation o_{t+1} , and the belief is updated using Bayesian inference:

$$b_{t+1}(s') = \frac{Z_P(o_{t+1} | s', u_t) \sum_{s \in \mathcal{S}_P} T_P(s' | s, u_t) b_t(s)}{\sum_{\bar{s} \in \mathcal{S}_P} Z_P(o_{t+1} | \bar{s}, u_t) \sum_{s \in \mathcal{S}_P} T_P(\bar{s} | s, u_t) b_t(s)} \quad (40)$$

where $Z_P(o_{t+1} | s', u_t)$ denotes the likelihood of receiving observation o_{t+1} after action u_t when the execution state is s' . This iterative process—comprising prediction, observation, and belief update—enables continuous refinement of decisions under uncertainty.

5.3. Action Selection Heuristic

The Action Selection Heuristic (ASH) is adopted as the online action-selection mechanism of the POMDP execution layer. Before implementing the ASH, we convert the proposed POMDP into the corresponding MDP by assuming all states to be observable. Then we solve the MDP to get the optimal policy under full state observation (i.e., fully observable state policy). ASH reuses the fully observable state policy and aggregates its action recommendations according to the current belief. In the present framework,

the TCM supplies the task context, the POMDP updates the belief over the partially observable behavioral states, and ASH selects the corresponding execution action.

Let $\pi_F^*(s)$ denote the offline policy obtained for the underlying fully observable execution-state model. At decision epoch t , let $\{s_j, w_j\}_{j=1}^{N_b}$ denote a weighted representation of the current belief, where $w_j \geq 0$ and $\sum_{j=1}^{N_b} w_j = 1$. For an exactly represented discrete belief, $w_j = b_t(s_j)$; for a particle representation, w_j is the normalized particle weight. The score assigned to action $u \in \mathcal{D}_P$ is

$$\text{Score}_t(u) = \sum_{j=1}^{N_b} w_j \mathbb{I}[\pi_F^*(s_j) = u], \quad (41)$$

where $\mathbb{I}[\cdot]$ is the indicator function. The selected action is

$$u_t^{\text{ASH}} \in \arg \max_{u \in \mathcal{D}_P} \text{Score}_t(u). \quad (42)$$

If several actions receive the same score, the tie is resolved in favor of the action supported by the largest individual belief weight. Thus, ASH selects the action recommended by the greatest cumulative probability mass rather than assuming that the most likely state is known with certainty.

ASH differs from standard POMDP solution methods in both objective and computational structure. Point-based value iteration approximates the belief-space value function by performing Bellman backups over a selected set of reachable beliefs [33]. It therefore optimizes actions directly in belief space, but its offline computational and storage requirements increase with the number of belief points, actions, and observations. Monte Carlo tree-search methods, such as partially observable Monte Carlo planning (POMCP), instead perform online look-ahead simulations from the current belief [34]. They provide anytime planning and can accommodate large state spaces, but their decision latency depends on the rollout budget and planning horizon.

In contrast, ASH performs neither belief-space value-function backups nor online tree search. Once π_F^* has been generated, action selection requires one policy lookup and one score accumulation for each supported belief state, giving an online complexity of $\mathcal{O}(N_b)$ and working storage of $\mathcal{O}(|\mathcal{D}_P|)$. This provides predictable and low online computational demand for the discrete execution model. The trade-off is that ASH does not provide the POMDP optimality or approximation guarantees associated with belief-space solution methods. It is therefore used here as a practical policy-reuse mechanism for

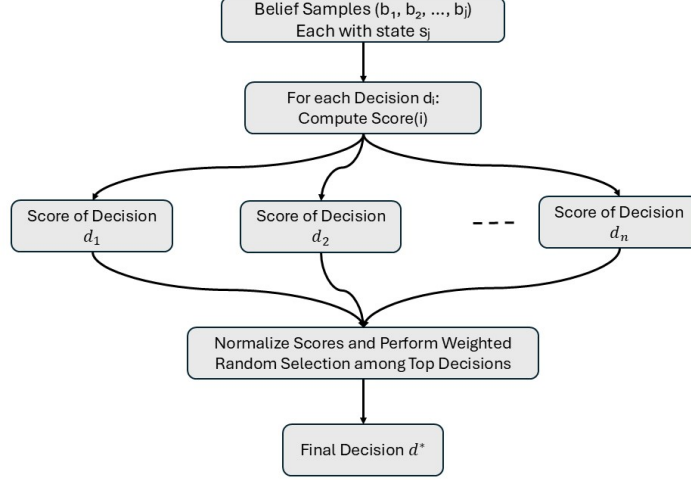


Figure 3: Framework of the proposed Action Selection Heuristic (ASH).

belief-dependent execution, rather than as a replacement for general POMDP solvers.

5.4. Task Coordination Module

The Task Coordination Module (TCM) implements the event-driven interface between the scheduling MDP and the safety-execution POMDP. It does not introduce a third optimization problem or modify the actions selected by either decision layer. Instead, it (i) interprets MDP actions, (ii) transfers the corresponding task context to the POMDP, (iii) coordinates the execution of compatible assignments, and (iv) returns completion, availability, and fatigue information to the scheduler.

Let $\mathcal{Q}_t$ denote the set of assignments issued by the MDP but not yet activated, and let $\mathcal{A}_t$ denote the set of active assignments. For task j , the information transferred by the TCM is

$$\mathbf{c}_j = (j, \tau_{p,j}, \tau_{c,j}, \tau_{x,j}, \mathcal{R}_j), \quad (43)$$

where $\tau_{p,j}$, $\tau_{c,j}$, and $\tau_{x,j}$ denote the task-priority level, estimated execution-duration class, and task nature, respectively, and $\mathcal{R}_j \subseteq \{H, R\}$ is the human, robot, or joint resource set selected by the scheduling policy.

Two assignments j and k are eligible for concurrent execution when

$$\text{Compat}(j, k) = \mathbb{I}[\mathcal{R}_j \cap \mathcal{R}_k = \emptyset] \mathbb{I}[\tau_{p,j} = \tau_{p,k}], \quad (44)$$

where $\mathbb{I}[\cdot]$ is the indicator function. Thus, concurrent execution is permitted only for equal-priority assignments that use different available agents. A joint task occupies both resources and cannot overlap with another assignment. Assignments that do not satisfy Eq. (44) remain in $\mathcal{Q}_t$ until their required resources become available.

Algorithm 1 summarizes the coordination procedure. At each scheduling epoch, the TCM receives the MDP action. A task-assignment action creates a task context and places it in the execution queue, whereas rest, cooling, recall, transition, and idle actions are dispatched directly to the corresponding agent. Compatible queued assignments are activated when their required resources are available. During execution, the TCM constructs the POMDP context from the active assignments and agent states. The POMDP then updates its belief from the current observation, and ASH selects the execution action. When a task is completed, the TCM removes it from the active set and communicates the updated completion, availability, and fatigue states to the MDP before the next scheduling decision.

The TCM therefore preserves the separation of decision responsibilities: the MDP selects task-assignment and recovery actions, the POMDP-ASH layer selects belief-dependent execution actions, and the TCM manages their timing, information exchange, and resource compatibility.

5.5. Computational Complexity and Scalability

The proposed framework separates offline policy generation from online decision making. The scheduling policy π_M^* and the underlying fully observable execution-state policy π_F^* are generated offline and stored. During online operation, the scheduling layer retrieves an action from π_M^* , whereas the execution layer updates its belief and applies the Action Selection Heuristic (ASH) to π_F^* .

For the scheduling MDP, let m denote the number of task slots represented simultaneously. Let n_p , n_c , and n_x denote the cardinalities of the task-priority, execution-duration-class, and task-nature variables, respectively. The scheduling state-space cardinality is

$$|\mathcal{S}_M| = (n_p n_c n_x)^m n_{h_a} n_{b_a} n_{h_f} n_{b_f}, \quad (45)$$

where n_{h_a} , n_{b_a} , n_{h_f} , and n_{b_f} are the cardinalities of the human and robot availability and fatigue states.

Algorithm 1 Task Coordination Module

Require: Scheduling policy π_M , execution state policy π_F^* , initial scheduling state $\mathbf{s}_{M,0}$, initial belief $\mathbf{b}_0$, and task set $\mathcal{T}$

Ensure: Coordinated scheduling and execution actions

```
1:  $t \leftarrow 0$ ,  $\mathcal{Q}_0 \leftarrow \emptyset$ ,  $\mathcal{A}_0 \leftarrow \emptyset$ 
2: while an unfinished, queued, active, or recovery operation exists do
3:   Observe the current scheduling state  $\mathbf{s}_{M,t}$ 
4:    $d_{M,t} \leftarrow \pi_M(\mathbf{s}_{M,t})$ 
5:   if  $d_{M,t}$  assigns task  $j$  to resource set  $\mathcal{R}_j$  then
6:     Construct  $\mathbf{c}_j$  using Eq. (43)
7:      $\mathcal{Q}_t \leftarrow \mathcal{Q}_t \cup \{j\}$ 
8:   else if  $d_{M,t}$  is a recovery, recall, transition, or idle action then
9:     Dispatch  $d_{M,t}$  to the corresponding agent
10:  end if
11:  for each task  $j \in \mathcal{Q}_t$  do
12:    if the agents in  $\mathcal{R}_j$  are available and  $j$  is compatible with  $\mathcal{A}_t$  then
13:      Transfer  $\mathbf{c}_j$  to the execution layer
14:       $\mathcal{A}_t \leftarrow \mathcal{A}_t \cup \{j\}$ 
15:       $\mathcal{Q}_t \leftarrow \mathcal{Q}_t \setminus \{j\}$ 
16:    end if
17:  end for
18:  if  $\mathcal{A}_t \neq \emptyset$  then
19:    Acquire observation  $o_t$  and update belief  $\mathbf{b}_t$ 
20:    Select  $d_{P,t}$  using ASH
21:    Dispatch  $d_{P,t}$  to the execution layer
22:  end if
23:  Receive task-completion and agent-state information
24:  for each completed task  $j \in \mathcal{A}_t$  do
25:     $\mathcal{A}_t \leftarrow \mathcal{A}_t \setminus \{j\}$ 
26:    Return completion, availability, and fatigue updates to the scheduling state
27:  end for
28:   $t \leftarrow t + 1$ 
29: end while
```

The implementation stores an indexed successor for each state-action pair instead of constructing a dense transition-probability tensor. Consequently,

K_M value-iteration updates require

$$\mathcal{O}(K_M |\mathcal{S}_M| |\mathcal{D}_M|) \quad (46)$$

operations.

The underlying discrete state space of the execution layer has cardinality

$$|\mathcal{S}_P| = (\rho + 1)^3 (\sigma + 1)^3 4^3 3^2. \quad (47)$$

Generating its fully observable state policy through K_P value-iteration updates therefore requires

$$\mathcal{O}(K_P |\mathcal{S}_P| |\mathcal{D}_P|). \quad (48)$$

After π_F^* has been generated, ASH aggregates the policy recommendations associated with the N_b states supporting the current belief. Its online action-selection complexity is $\mathcal{O}(N_b)$, with working storage $\mathcal{O}(|\mathcal{D}_P|)$.

Solving the scheduling and execution layers separately avoids explicitly constructing a monolithic state space with cardinality

$$|\mathcal{S}_M \times \mathcal{S}_P| = |\mathcal{S}_M| |\mathcal{S}_P|. \quad (49)$$

This decomposition preserves the distinct decision roles of the two layers while avoiding enumeration of their complete Cartesian product.

Equation (45) nevertheless shows that the scheduling state space grows exponentially with the number of simultaneously represented task slots. The present implementation therefore employs a bounded three-task scheduling window. Longer task streams are processed through successive windows while preserving task identities and transferring the terminal agent states between windows. Thus, the framework supports longer task streams under a fixed window size but does not eliminate the combinatorial growth associated with increasing m . Larger scheduling windows may be addressed through reachability-based pruning, sparse state generation, rolling-horizon scheduling, factored representations, or approximate dynamic programming.

6. Numerical Evaluation

This section evaluates the proposed hierarchical framework through three complementary analyses. First, a representative trajectory demonstrates the interaction among the scheduling MDP, TCM, and POMDP-ASH execution layer. Second, an extended 12-task experiment evaluates policy reuse and

Table 3: Parameters used in the numerical evaluation

Layer	Parameter	Value
Common	Discount factor, γ	0.95
Scheduling	Concurrent task slots, m	3
Scheduling	Maximum priority level and duration class, (ρ, σ)	(2, 2)
Scheduling	Fatigue limits, (ϵ_h, ϵ_b)	(2, 2)
Scheduling	Cost weights, $(\alpha_1, \dots, \alpha_4)$	(400, 50, 200, 1000)
Execution	Latent-state limits, (ϵ_m, ϵ_a)	(2, 2)
Execution	Observation levels, L_o	3
Execution	Task-completion weights, $\{\alpha_i\}_{i=1}^9$	1
Execution	Safety weights, $\{\beta_i\}_{i=1}^6$	(10, 10, 10, 1, 10, 1)

agent-state continuity across successive scheduling windows. Finally, the offline policy-generation cost and online policy-access time are quantified. These analyses assess the decision-making behavior and computational feasibility of the hierarchical framework under the prescribed observation model; participant-based calibration and validation of the human-state estimation model remain subjects of future work.

6.1. Experimental Configuration

Table 3 lists the parameters used in the numerical evaluation. The scheduling coefficients weight task urgency, execution-duration-class preference, fatigue-aware assignment, and robot-preferred allocation of eligible flexible tasks. Assignment compatibility, agent availability, and maximum-fatigue conditions are enforced through the admissible action set rather than through penalty terms. The execution-layer weights regulate task-completion and safety-oriented behavior. The same parameters and precomputed policies are used in all evaluated task windows.

The scheduling-cost coefficients were specified according to the relative weight assigned to the four normalized objectives and were retained unchanged across all evaluated task windows. The selected values emphasize task urgency and robot allocation of eligible flexible tasks, while execution-duration class and fatigue provide additional tradeoffs among feasible assignments. These weights do not impose a strict lexicographic ordering; compatibility, availability, and maximum-fatigue requirements are enforced independently through the feasible action set.

For the case study, the joint latent behavioral state and observation spaces are

$$\begin{aligned}\mathcal{E}_{\text{case}} &= \{0, 1, 2\} \times \{0, 1, 2\}, \quad |\mathcal{E}_{\text{case}}| = 9, \\ \mathcal{O}_{\text{case}} &= \{o_1, o_2, o_3\},\end{aligned}\tag{50}$$

where o_1 , o_2 , and o_3 represent low, moderate, and high motion activation, respectively. The latent states are ordered lexicographically as $(0, 0), (0, 1), \dots, (2, 2)$. The first observation of an execution episode initializes the belief using

$$\mathbf{B}_{\text{case}} = \begin{bmatrix} 0.80 & 0 & 0 \\ 0.10 & 0.10 & 0 \\ 0 & 0 & 0.05 \\ 0.10 & 0.10 & 0 \\ 0 & 0.70 & 0 \\ 0 & 0.05 & 0.10 \\ 0 & 0 & 0.05 \\ 0 & 0.05 & 0.10 \\ 0 & 0 & 0.70 \end{bmatrix}, \tag{51}$$

where $B_{qk} = \Pr(\mathbf{e}^{(q)} \mid o_k)$ and every column sums to one. Thus, $\mathbf{B}_{\text{case}}$ is an observation-conditioned belief-initialization map, rather than the likelihood matrix $Z_P(o_k \mid \mathbf{e}^{(q)})$. Subsequent beliefs are propagated using the Bayesian recursion defined for the POMDP execution layer. The prescribed map allows the coordination and action-selection mechanisms to be evaluated under controlled observation uncertainty; its calibration from physical sensing data is outside the scope of this numerical study.

The representative scheduling state is initialized as

$$\mathbf{s}_{M,0} = [1, 2, 3, 2, 2, 0, 2, 2, 2, 1, 1, 0, 0], \tag{52}$$

where each of the first three triplets specifies a task's priority level, execution-duration class, and nature. Accordingly, Task 1 is a low-priority joint task, Task 2 is a high-priority robot-only task, and Task 3 is a high-priority flexible task that can be assigned to either agent. All three tasks belong to execution-duration class 2, representing the longer of the two positive duration classes used in the numerical model. The value does not denote two physical time units or imply that the task must be completed in exactly two scheduling epochs. The final four entries indicate that the human and robot are initially available and have zero fatigue.

When task j is selected, the TCM transfers its attributes and the selected agent to the execution layer. The corresponding initial POMDP state is

$$\mathbf{s}_{P,0}^{(j)} = [0, 0, \tau_{p,j}, \tau_{c,j}, \tau_{x,j}, 0, 0, 0, 0, 0, 1], \quad (53)$$

where $\tau_{p,j}$, $\tau_{c,j}$, and $\tau_{x,j}$ denote the task-priority level, execution-duration class, and task nature, respectively. The commitment variable τ_y , represented by the sixth entry, is initially zero because execution has not yet begun.

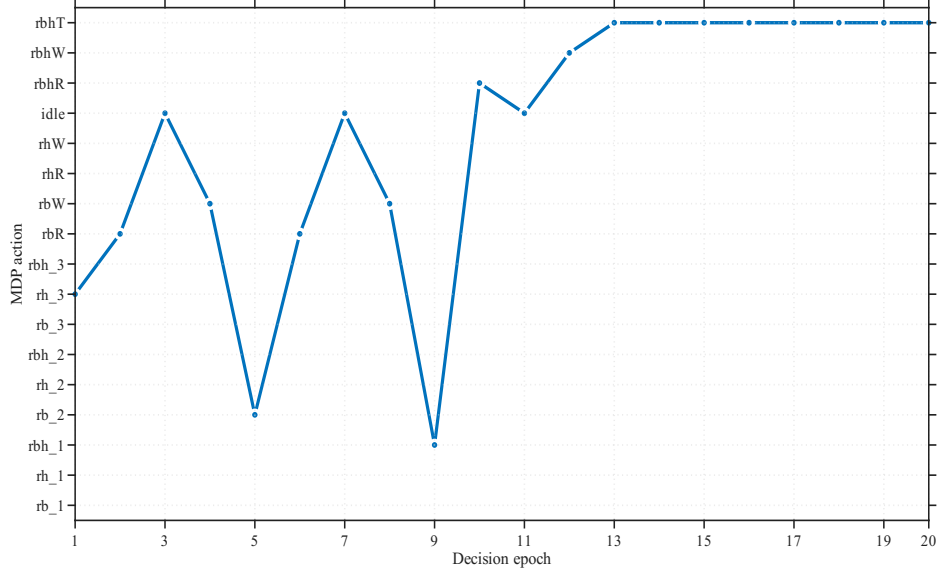
6.2. Representative Closed-Loop Evaluation

Figure 4 presents the scheduling decisions and the resulting agent states. The MDP first selects the two high-priority tasks, assigning the flexible task to the human and the robot-only task to the robot. After completing their respective assignments, both agents reach their fatigue thresholds and enter the recovery intervals. The low-priority joint task is consequently delayed until the human and robot are simultaneously available. The resulting trajectory demonstrates that the scheduling policy accounts for task priority, execution requirements, agent availability, and fatigue-dependent recovery.

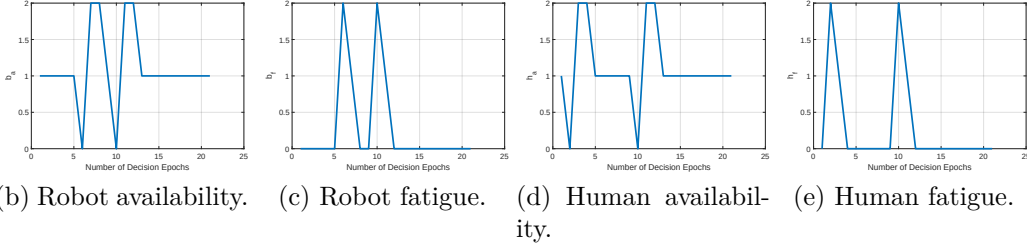
The TCM permits concurrent execution when the selected tasks have compatible precedence relations, require disjoint resources, and satisfy the applicable availability conditions. In the representative scenario, the two high-priority assignments satisfy these conditions and are therefore activated concurrently, although their MDP assignment actions occur at successive decision epochs. Resource occupancy, task completion, and fatigue progression remain separately tracked for each agent, and the updated states are returned to the scheduling layer after execution.

Figure 5 presents the corresponding coordination and execution behavior. The TCM activates the compatible human- and robot-assigned tasks, while the POMDP-ASH layer selects belief-dependent execution and distance-regulation actions. The low-priority collaborative task is admitted only after both agents complete their required recovery intervals. Thus, the MDP determines which task should be performed and by whom, the TCM governs activation and cross-layer information exchange, and the POMDP-ASH layer regulates the execution behavior under partial observability.

The task-conditioned evolution of the separation state is reported in Fig. 6. For the solo Tasks 3 and 2, the policy increases the separation from the initial minimum state and maintains the large-distance state. By contrast, for the collaborative Task 1, the policy selects distance-decrease actions and



(a) Selected scheduling actions.



(b) Robot availability. (c) Robot fatigue. (d) Human availability. (e) Human fatigue.

Figure 4: Representative MDP scheduling trajectory. The selected actions coordinate task assignment and agent recovery according to task priority, execution requirements, resource availability, and fatigue state.

approaches the minimum non-collision separation required for close cooperation. The commitment markers identify changes in the task-commitment variable τ_y ; they do not denote physical task completion or independently certify execution authorization. In particular, commitment may occur while the POMDP continues regulating the separation state. No collision state is visited during the evaluated trajectory. The result therefore demonstrates that the POMDP does not apply a single fixed separation requirement: it maintains greater separation during solo operation and closer non-collision separation during collaborative operation. This task-conditioned distance regulation provides explicit trajectory-level evidence of the safety-aware role

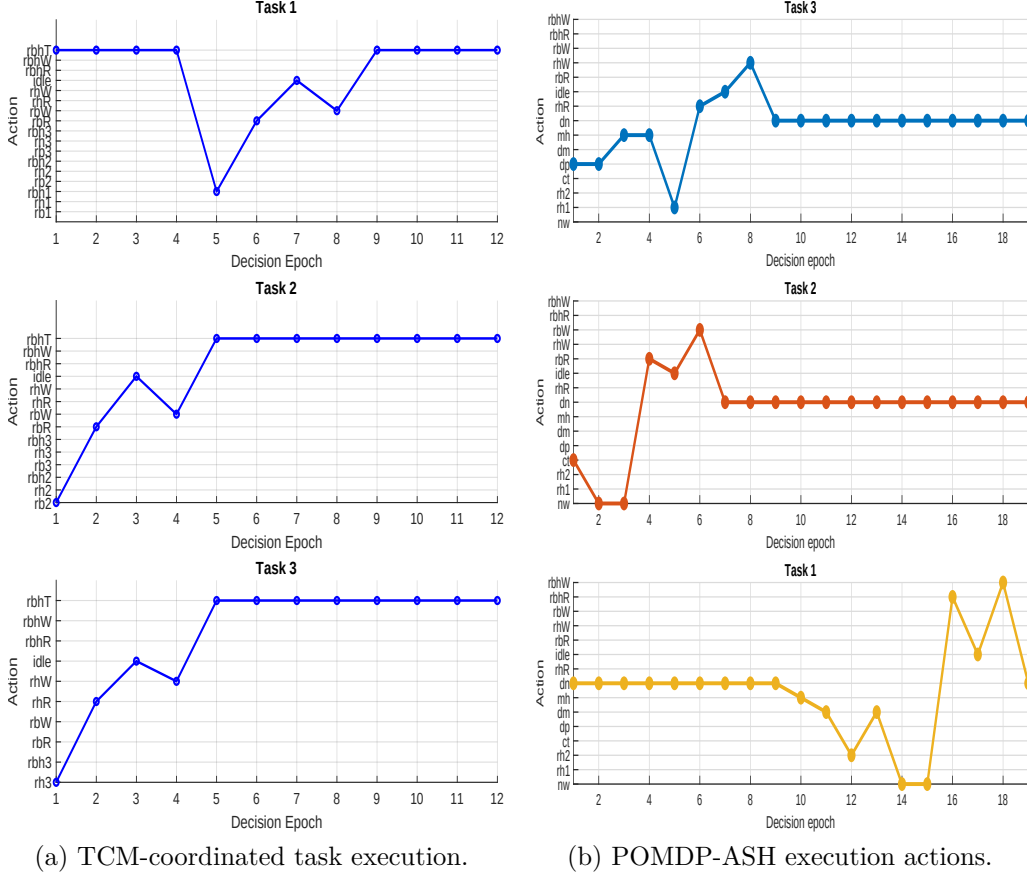


Figure 5: Representative closed-loop behavior of the coordination and execution layers. The TCM activates compatible assignments, while the POMDP-ASH layer selects belief-dependent execution and distance-regulation actions for robot-involved tasks.

of the execution layer.

6.3. Extended Workload Evaluation

To evaluate policy reuse over a longer heterogeneous workload, 12 persistent tasks are introduced through four successive three-task windows. The windows represent mixed, human-intensive, robot-intensive, and collaboration-intensive workloads, as summarized in Table 4. The same MDP policy and model parameters are used throughout without window-specific retuning. A new window is admitted after the current tasks leave the scheduling pool

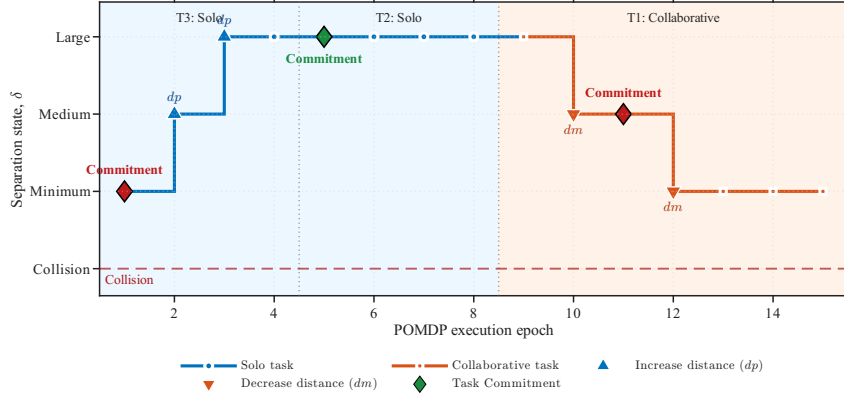


Figure 6: Task-conditioned human-robot separation during the representative closed-loop trajectory.

Table 4: Task windows used in the extended evaluation

Window	Tasks	Workload emphasis
1	$T1-T3$	Mixed
2	$T4-T6$	Human-intensive
3	$T7-T9$	Robot-intensive
4	$T10-T12$	Collaboration-intensive

and both agents return to the available state; the terminal fatigue states are retained at the window boundary.

The reported realization schedules all 12 tasks over 37 MDP decision epochs. Across the complete stream, six tasks involve the robot and nine involve the human, with joint tasks included in both counts. As shown in Fig. 7, the realized allocation changes with the workload composition: human utilization increases in the second window, robot utilization increases in the third, and the joint assignments in the fourth are delayed until both agents are available. This result demonstrates reuse of the same state-feedback policy across heterogeneous task configurations.

Figure 8 confirms that the scheduling windows are not treated as independent experiments. Human and robot fatigue is carried across window boundaries and directly affects subsequent assignment and recovery decisions. The corresponding availability trajectories alternate among working, recovery, and available states, while the cumulative task count reaches 12.

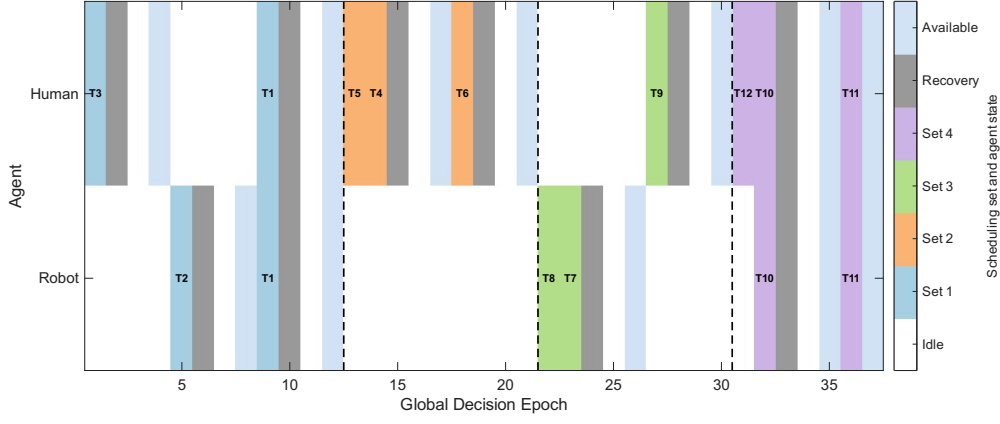


Figure 7: Realized task assignments and agent states across four successive scheduling windows. Task colors identify the workload window, and recovery and availability intervals are shown separately.

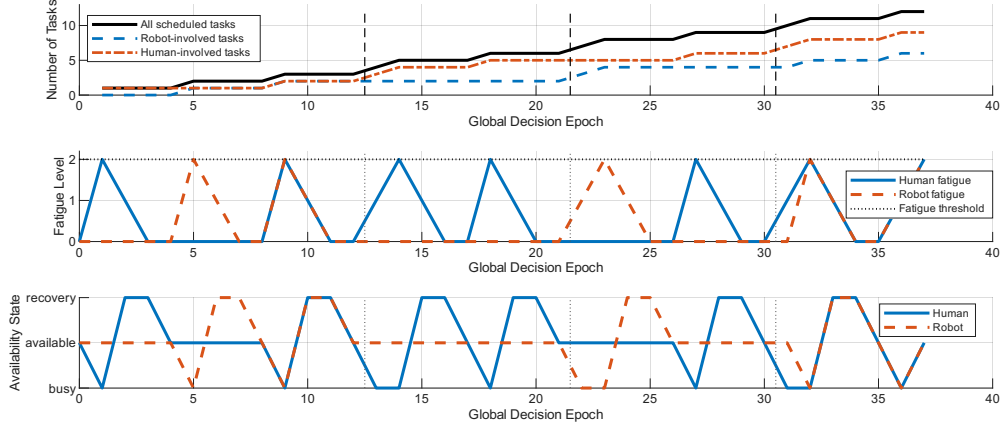


Figure 8: Cumulative task assignments and agent-state evolution across the 12-task workload. Vertical dotted lines indicate transitions between successive scheduling windows.

These results establish workload continuity under the fixed three-task state representation.

The TCM maintains persistent task identifiers independently of the task slots used by the MDP. Each selected task progresses through

$$\text{unassigned} \rightarrow \text{queued} \rightarrow \text{active} \rightarrow \text{completed}. \quad (54)$$

Human-only assignments follow the human-execution path, whereas robot-only and joint assignments are routed to the POMDP-ASH layer. Comple-

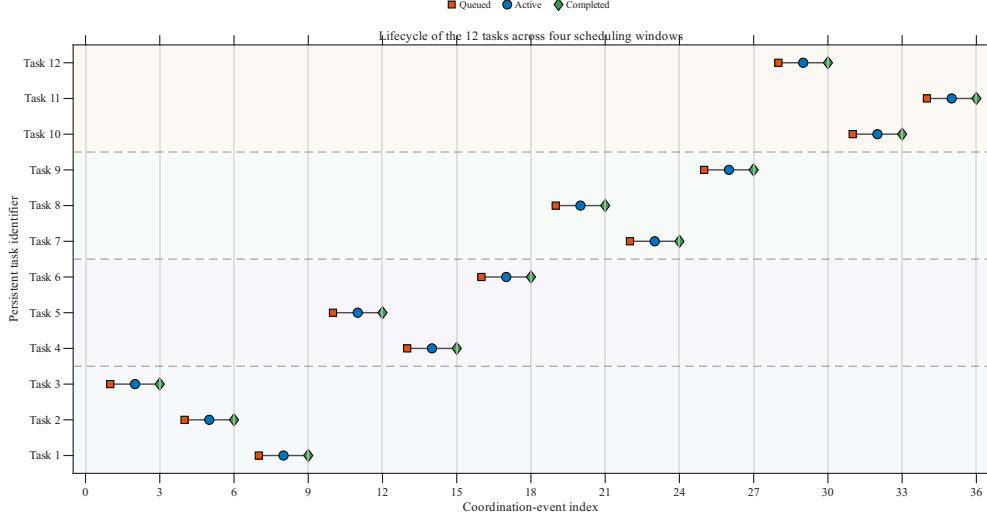


Figure 9: Lifecycle transitions of the 12 persistent tasks. Squares, circles, and diamonds denote queued, active, and completed states, respectively; shaded regions separate successive scheduling windows.

tion of a robot-involved task is recorded only after the terminal execution action dn is reached. Figure 9 verifies that all 12 assignments are registered, activated, and closed by execution feedback. Its horizontal axis represents coordination-event order rather than physical execution time.

The extended experiment demonstrates workload-length extensibility rather than simultaneous optimization of a 12-task state. At most three tasks are represented concurrently, and additional tasks are admitted through successive windows while global task identities and agent states are preserved. Increasing the number of simultaneously represented tasks would enlarge the exact MDP state space and require a separate scalability study.

6.4. Computational Performance

The computational evaluation was conducted in MATLAB R2026a on a 64-bit computer equipped with a 2.40 GHz Intel Core i7-4700HQ processor and 16 GB of random-access memory (RAM). All computations were performed serially without parallel or graphics processing unit (GPU) acceleration. The scheduling and execution-state policies were generated offline using in-place value iteration with the convergence condition.

$$\|V^{k+1} - V^k\|_2 < 10^{-6}. \quad (55)$$

Table 5: Computational profile of the decision layers

Quantity	Scheduling	Execution
States	3,779,136	419,904
State-level actions	17	13
Value-iteration updates	8	385
Final 2-norm residual	0	9.8004×10^{-7}
Offline generation time	866.293 s	2712.983 s
Principal-array storage	1.070 GB	0.094 GB
Mean indexed-lookup time	3.082 μ s	0.053 μ s

Table 5 reports the policy-generation cost, principal-array storage, and indexed-lookup time of the two decision layers. The offline generation times include model construction, value iteration, policy extraction, initialization, data loading, memory allocation, and policy-storage overhead.

The offline generation times are incurred once during policy synthesis and do not represent deployed decision latency. During online scheduling, the current state is mapped directly to the stored policy π_M^* , yielding a mean indexed-lookup time of 3.082 μ s. The corresponding value of 0.053 μ s measures access to the underlying fully observable execution-state policy π_F^* . It does not include observation processing, belief updating, action-admissibility evaluation, or belief-weighted ASH selection. The lookup measurements were obtained using 100,000 randomly sampled state indices.

Table 6 places these results in the context of representative scheduling architectures. The comparison distinguishes complete schedule optimization, event-triggered reoptimization, and online state-feedback policy evaluation because these operations have fundamentally different computational requirements.

Nourmohammadi *et al.* [35] construct complete schedules from predefined task times and resource configurations. Their reported central processing unit (CPU) times range from 1.1 to 22.7 s for small instances and from 45.2 to 1067.1 s for medium instances. Their formulation assumes continuous robot availability and excludes maintenance and breakdown. Moreover, the term “adaptive simulated annealing (SA)” refers to adaptation of the neighborhood-selection mechanism during optimization, rather than online adaptation of the deployed schedule.

Event-triggered methods provide a closer comparison with online opera-

Table 6: Comparison of computational architectures and reported execution times for representative scheduling methods.

Method	Offline stage	Online response	Online iterative solve	Online information	Reported time
MILP and metaheuristics [35]	Complete schedule optimization	No online adaptation evaluated	No	Predefined task durations, resources, and execution modes	1.1–22.7 s (small); 45.2–1067.1 s (medium)
Lippi <i>et al.</i> [8]	Initial MILP schedule	Event-triggered task reallocation	Yes	Execution time, workload, preferences, new tasks, and resource availability	Not reported
Wulan [36]	Dynamic-programming policy generation	State-dependent policy evaluation	No	Order arrivals and equipment capacity; non-HRC application	Not reported
Proposed framework	Offline hierarchical policy synthesis	Direct scheduling-policy lookup and belief-dependent action selection	No	Task status, agent availability, fatigue, recovery, and latent human conditions	Scheduling: 3.082 μ s; Execution: 0.053 μ s

Note: Reported values correspond to the computational operations evaluated in the respective studies and therefore do not constitute a controlled runtime comparison. HRC denotes human-robot collaboration.

tion. Lippi *et al.* [8] update execution parameters, human preferences, new tasks, and resource availability during execution. When the existing schedule becomes infeasible or sufficiently suboptimal, however, a new MILP is solved for the remaining tasks. This provides online adaptability, but the response time depends on the computational behavior of the optimization solver; the corresponding online reoptimization time was not reported. Wulan [36] avoids repeated optimization through a dynamic-programming decision policy, although the formulation concerns order arrivals and equipment capacity rather than human-robot collaboration.

The proposed framework similarly transfers its principal computational burden to offline policy synthesis but extends online state-feedback decision making to task status, agent availability, fatigue, recovery, and partially observable human conditions. At each scheduling epoch, the current state is evaluated through π_M^* without invoking an iterative MILP or metaheuristic solver. The POMDP-ASH layer then uses the updated belief and the underlying policy π_F^* to select safety-oriented execution actions. This separation provides predictable microsecond-scale scheduling-policy access and avoids

solver-dependent convergence delays within the online scheduling loop.

The resulting architecture is suitable for rapid response to task and agent-state changes represented within the modeled state and action domains. It also provides a basis for future high-rate HRC planning in which online human-state estimation is coordinated with task scheduling and safety adaptation. Because the published methods differ in their objectives, instance sizes, hardware, and measured computational operations, Table 6 is interpreted as a comparison of computational architectures rather than as a controlled numerical speedup.

7. Conclusion

This paper presented a hierarchical stochastic decision-making framework that couples fatigue- and availability-aware task scheduling with belief-dependent, safety-oriented execution for human-robot collaboration. The scheduling MDP determines task-agent assignments and recovery actions from the task and agent states, while the POMDP-ASH layer regulates task-conditioned execution and human-robot separation under partial observability. The TCM connects these decision layers by preserving task status, evaluating resource compatibility, transferring task semantics, and returning execution and agent-state information to the scheduler. The numerical evaluation demonstrated priority-aware task allocation, fatigue-dependent recovery, admissible concurrent execution, and task-conditioned separation behavior. Greater separation was maintained during solo-agent execution, whereas closer noncollision separation was permitted for joint tasks. The same precomputed policies were reused across four successive scheduling windows comprising 12 persistent tasks, with task identities and agent states preserved across window boundaries. Exact policy generation was computationally feasible at the evaluated scale, and the mean indexed-access times were $3.082\ \mu\text{s}$ for the scheduling policy and $0.053\ \mu\text{s}$ for the underlying execution-state policy. These values characterize policy access rather than complete sensing-to-action latency, but confirm that policy retrieval itself is not a significant online computational burden.

Future work will extend the numerical foundation through calibration of the observation and human-state models using synchronized kinematic, physiological, and behavioral measurements from human participants, followed by evaluation in physical HRC experiments. Complete sensing-to-action latency and separation behavior will be measured under dynamic operating

conditions. Reachability-based pruning, rolling-horizon scheduling, factored state representations, and approximate dynamic programming will also be investigated for larger simultaneous task windows and multi-agent HRC configurations.

References

- [1] H. R. Chi, C. K. Wu, N.-F. Huang, K.-F. Tsang, A. Radwan, A survey of network automation for industrial internet-of-things toward industry 5.0, *IEEE Transactions on Industrial Informatics* 19 (2) (2022) 2065–2077.
- [2] S. Li, P. Zheng, S. Liu, Z. Wang, X. V. Wang, L. Zheng, L. Wang, Proactive human–robot collaboration: Mutual-cognitive, predictable, and self-organising perspectives, *Robotics and Computer-Integrated Manufacturing* 81 (2023) 102510.
- [3] M. L. Nicora, R. Ambrosetti, G. J. Wiens, I. Fassi, Human–robot collaboration in smart manufacturing: Robot reactive behavior intelligence, *Journal of Manufacturing Science and Engineering* 143 (3) (2021) 031009.
- [4] M. Lippi, A. Marino, A mixed-integer linear programming formulation for human multi-robot task allocation, in: 2021 30th IEEE International Conference on Robot & Human Interactive Communication (RO-MAN), IEEE, 2021, pp. 1017–1023.
- [5] O. Battaïa, A. Dolgui, N. Guschinsky, H. Tavakkoli-Moghaddam, A novel milp model for human-robot collaboration in assembly line balancing under budget constraints, *IFAC-PapersOnLine* 59 (10) (2025) 1922–1927.
- [6] B. Yao, X. Li, Z. Ji, K. Xiao, W. Xu, Task reallocation of human-robot collaborative production workshop based on a dynamic human fatigue model, *Computers & Industrial Engineering* 189 (2024) 109855.
- [7] S. Lou, R. Tan, Y. Zhang, M. Zhou, C. Lv, Personalized disassembly sequence planning for a human–robot hybrid disassembly cell, *IEEE Transactions on Industrial Informatics* 20 (9) (2024) 11372–11383.

- [8] M. Lippi, P. Di Lillo, A. Marino, A task allocation framework for human multi-robot collaborative settings, arXiv preprint arXiv:2210.14036 (2022).
- [9] C. S. Franklin, E. G. Dominguez, J. D. Fryman, M. L. Lewandowski, Collaborative robotics: New era of human–robot cooperation in the workplace, *Journal of Safety Research* 74 (2020) 153–160.
- [10] A. Giallanza, G. La Scalia, R. Micale, C. M. La Fata, Occupational health and safety issues in human-robot collaboration: State of the art and open challenges, *Safety science* 169 (2024) 106313.
- [11] M. Faroni, A. Spanò, A. M. Zanchettin, P. Rocco, Learning-based safety-aware task scheduling for efficient human-robot collaboration, *IEEE Robotics and Automation Letters* 11 (2) (2025) 2226–2233.
- [12] N. Gopalan, S. Tellex, Modeling and solving human–robot collaborative tasks using POMDPs, in: *Proceedings of the RSS Workshop on Model Learning for Human–Robot Communication*, 2015, pp. 590–628.
URL <https://cs.brown.edu/people/ngopalan/docs/gopalan15.pdf>
- [13] M. Chen, S. Nikolaidis, H. Soh, D. Hsu, S. Srinivasa, Trust-aware decision making for human-robot collaboration: Model learning and planning, *ACM Transactions on Human-Robot Interaction (THRI)* 9 (2) (2020) 1–23.
- [14] W. Zheng, B. Wu, H. Lin, Pomdp model learning for human robot collaboration, in: *2018 IEEE Conference on Decision and Control (CDC)*, IEEE, 2018, pp. 1156–1161.
- [15] X. Zhang, H. Lin, Performance guaranteed human-robot collaboration with pomdp supervisory control, *Robotics and Computer-Integrated Manufacturing* 57 (2019) 59–72.
- [16] M. Dalle Mura, G. Dini, Designing assembly lines with humans and collaborative robots: A genetic approach, *CIRP Annals* 68 (1) (2019) 1–4.

- [17] C. Wei, Z. Ji, B. Cai, Particle swarm optimization for cooperative multi-robot task allocation: a multi-objective approach, *IEEE Robotics and Automation Letters* 5 (2) (2020) 2530–2537.
- [18] R. Gervasi, K. Aliev, L. Mastrogiacomo, F. Franceschini, User experience and physiological response in human-robot collaboration: a preliminary investigation, *Journal of Intelligent & Robotic Systems* 106 (2) (2022) 36.
- [19] A. Buerkle, H. Matharu, A. Al-Yacoub, N. Lohse, T. Bamber, P. Ferreira, An adaptive human sensor framework for human–robot collaboration, *The International Journal of Advanced Manufacturing Technology* 119 (1) (2022) 1233–1248.
- [20] K. Papoutsakis, G. Papadopoulos, M. Maniadakis, T. Papadopoulos, M. Lourakis, M. Pateraki, I. Varlamis, Detection of physical strain and fatigue in industrial environments using visual and non-visual low-cost sensors, *Technologies* 10 (2) (2022) 42.
- [21] S. Savazzi, V. Rampa, F. Vicentini, M. Giussani, Device-free human sensing and localization in collaborative human–robot workspaces: A case study, *IEEE Sensors Journal* 16 (5) (2015) 1253–1264.
- [22] A. Cherubini, R. Passama, A. Crosnier, A. Lasnier, P. Fraisse, Collaborative manufacturing with physical human–robot interaction, *Robotics and Computer-Integrated Manufacturing* 40 (2016) 1–13.
- [23] K. H. Kim, I. J. Park, J.-H. Choi, S. Rhim, Evaluation of head-collision safety of a 7-dof manipulator according to posture variation, *Multibody system dynamics* 37 (1) (2016) 95–105.
- [24] X. Zhao, T. Fan, Y. Li, Y. Zheng, J. Pan, An efficient and responsive robot motion controller for safe human-robot collaboration, *IEEE Robotics and Automation Letters* 6 (3) (2021) 6068–6075.
- [25] W. Ma, A. Duan, H.-Y. Lee, P. Zheng, D. Navarro-Alarcon, Human-aware reactive task planning of sequential robotic manipulation tasks, *IEEE Transactions on Industrial Informatics* (2025).

- [26] M. K. M. Rabby, A. Karimoddini, M. A. Khan, S. Jiang, A learning-based adjustable autonomy framework for human–robot collaboration, *IEEE Transactions on Industrial Informatics* 18 (9) (2022) 6171–6180.
- [27] C. Wang, J. Zhao, Role dynamic assignment of human–robot collaboration based on target prediction and fuzzy inference, *IEEE Transactions on Industrial Informatics* 20 (1) (2023) 471–481.
- [28] M. A. Yusuf, A. Nasir, Z. H. Khan, Stochastic decision-making framework for human-robot collaboration in industrial applications, manuscript under review at *IEEE Transactions on Human-Machine Systems* (2025).
- [29] E. Martinez-Martin, A. Fernández-Caballero, Improved human emotion recognition from body and hand pose landmarks on the gemep dataset using machine learning, *Expert Systems with Applications* 269 (2025) 126427.
- [30] E. Ghaleb, A. Mertens, S. Asteriadis, G. Weiss, Skeleton-based explainable bodily expressed emotion recognition through graph convolutional networks, in: 2021 16th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2021), IEEE, 2021, pp. 1–8.
- [31] H. Lu, J. Chen, Z. Zhang, R. Liu, R. Zeng, X. Hu, Emotion recognition from skeleton data: A comprehensive survey, *arXiv preprint arXiv:2507.18026* (2025).
- [32] N. Dael, M. Mortillaro, K. R. Scherer, Emotion expression in body action and posture., *Emotion* 12 (5) (2012) 1085.
- [33] J. Pineau, G. Gordon, S. Thrun, et al., Point-based value iteration: An anytime algorithm for pomdps, in: *Ijcai*, Vol. 3, 2003, pp. 1025–1032.
- [34] D. Silver, J. Veness, Monte-carlo planning in large pomdps, *Advances in neural information processing systems* 23 (2010).
- [35] A. Nourmohammadi, M. Fathi, A. H. Ng, Balancing and scheduling assembly lines with human-robot collaboration tasks, *Computers & Operations Research* 140 (2022) 105674.

- [36] Q. Wulan, Order scheduling optimization in manufacturing enterprises based on mdp and dynamic programming, Scientific Reports 13 (1) (2023) 9783.