technologies

MDPI

Article

Real-Time Scheduling with Independent Evaluators: Explainable Multi-Agent Approach

Artem Isakov [1,2,*], Danil Peregorodiev [1], Ivan Tomilov [1], Chuyang Ye [3], Natalia Gusarova [1], Aleksandra Vatian [1] and Alexander Boukhanovsky [1]

[1] Department of Infocommunication Technologies, ITMO University, 197101 Saint Petersburg, Russia; deperegorodiev@itmo.ru (D.P.); iatomilov@itmo.ru (I.T.); nfgusarova@itmo.ru (N.G.); asvatyan@itmo.ru (A.V.); avbukhanovskii@itmo.ru (A.B.)
[2] World-Class Research Center for Personalized Medicine, Almazov National Medical Research Center, 194156 Saint Petersburg, Russia
[3] School of Integrated Circuits and Electronics, Beijing Institute of Technology, Beijing 100081, China; chuyang.ye@bit.edu.cn
* Correspondence: aoisakov@itmo.ru

Abstract: This study introduces a multi-agent reinforcement learning approach to address the challenges of real-time scheduling in dynamic environments, with a specific focus on healthcare operations. The proposed system integrates the Human-in-the-Loop (HITL) paradigm, providing continuous feedback from human evaluators, and it employs a sophisticated reward function to attenuate the effects of human-driven events. Novel mapping between reinforcement learning (RL) concepts and the Belief–Desire–Intention (BDI) framework is developed to enhance the explainability of the agent's decision-making. A system is designed to adapt to changes in patient conditions and preferences while minimizing disruptions to existing schedules. Experimental results show a notable decrease in patient waiting times compared to conventional methods while adhering to operator-induced constraints. This approach offers a robust, explainable, and adaptable solution for the challenging tasks of scheduling in the environments that require human-centered decision-making.

Keywords: decision making; explainability; multi-agent systems; real time; reinforcement learning; scheduling

Citation: Isakov, A.; Peregorodiev, D.; Tomilov, I.; Ye, C.; Gusarova, N.; Vatian, A.; Boukhanovsky, A. Real-Time Scheduling with Independent Evaluators: Explainable Multi-Agent Approach. *Technologies* **2024**, *12*, 259. https://doi.org/10.3390/technologies12120259

Academic Editor: George F. Fragulis

Received: 14 September 2024
Revised: 4 December 2024
Accepted: 7 December 2024
Published: 14 December 2024

1. Introduction

Scheduling workflow has always been highly challenging, and it has recently become increasingly complicated with the growing range of diversified jobs and with the dynamic demands that may require rescheduling. This task has been shown to be an NP-hard combinatorial problem [1] that cannot be solved in polynomial time with conventional game theoretic or linear programming algorithms. Therefore, a plethora of approximate gradient-based optimization algorithms can be used to address computational complexity. Among the family of neural network algorithms, the RL including Multi-Agent Reinforcement Learning (MARL) approach [2] stands out as the most promising one. Namely, Proximal Policy Optimization (PPO) [3] and its extension, Multi-Agent Proximal Policy Optimization (MAPPO) [4], are prevalent as stable baselines to numerical experiments due to their ability to work with time and other continuous spaces. MAPPO requires a number of entities called agents to learn to optimally interact in a shared environment during the simulation process, referred to as a game. The game represents a simplified working process to be optimized, whereas the agents represent the subprocesses within the working pipeline. As explicit computation for each working scenario is no longer required, MAPPO can effectively take into account the uncertainties related to the dynamic changes in the schedule.

In terms of dynamic changes, a human operator appears to be an active component of the workflow to be optimized, featuring a reactive behavior in response to the changes

caused by the algorithm. This theoretical concept is referred to as HITL and has been thoroughly discussed in [5]. To summarize, as soon as the operator has the opportunity to actively interfere with the running algorithm, the scheduling problem becomes part of a wider class of systemically complex problems. The NP-complexity of the challenging task of scheduling is aggravated by a number of implicit factors related to human orchestrating.

Another concept, called Human Feedback Reinforcement Learning (RLHF) [6], provides a general approach aimed at attenuating inherent uncertainty that results from a human in the loop. It uses various sophisticated reward function designs related to the preferences an expert expresses over time.

Reward-free approaches [7–9] aimed to address the same challenge all have the same limitations as the reward-based methods described above. They require the consent of several independent evaluators to proceed with a decision proposed by a model. This is aggravated with increases in the number of clients acting as those evaluators. One of the ways to mitigate the effects of the above limitation is to address the leading expert who communicates with the evaluators, following them and acting on their behalf, which results in successive rescheduling until one fit for all decisions is reached.

A typical example of such a scenario is arranging high-tech medical intervention flow in a specialized clinic, including surgery, dentistry, and high-tech medical examinations, to name a few. The degree of disease severity determines the acceptable waiting time for the intervention, which can also depend on personal preferences regarding the appointment timing. The clinic, on the other hand, can provide a certain number of interventions each day. A corresponding schedule is based on the above data and is partially observable for patients as required. However, during the waiting time, both the patient's condition and their preferences regarding the appointment time may change, which urges an ad hoc reorganization of the entire schedule. This may result in the waiting time of other clients increasing or expensive workstations being eventually underloaded.

To address these challenges, we propose an algorithm for real-time rescheduling of medical interventions in response to the eventual changes in the medical state and/or preferences of individual patients while minimizing the deviations in the already established schedules. In terms of the proposed algorithm, each human actor has two roles: a participant in the game process through the intellectual agent representing them, and a direct independent evaluator.

The main contributions of this paper are as follows:

1. We devised a novel multi-agent environment to simulate resource allocation in the varying of the planning horizon within dynamic scheduling;
2. We designed a sophisticated reward function with an adjustable α level to attenuate the effects of human-driven events;
3. We suggested a technique to refine the decision-making process during the sequential rescheduling phase with active human agents present;
4. We established mapping between RL nomenclature and a BDI cognitive framework.
5. We proposed Large Language Model (LLM)-based tools to elucidate the intricacies of cooperative–competitive game interactions between agents for a supervising human expert;
6. We applied our findings to the task of allocating operating rooms (OR).

The rest of this paper is organized as follows. In Section 2, we present a review of the recent studies on the main approaches related to our work. Particular attention is paid to the methods for specifying the parameters of human activity and behavior applied to address the challenge of real-time rescheduling. Section 3 covers the theoretical foundations common to most RL algorithms and to actor–critic architecture in particular. In Section 4, we present a custom gaming environment with an emphasis on the mapping between the environment elements, RL algorithm, and BDI framework to use the above relationship later when discussing explainability and robustness. Section 5 presents a comparative analysis of the proposed algorithm and conventional human heuristic. The further discussion covers current limitations and prospects.

2. Related Works

A survey by Abdalkarim et al [10] offers a comprehensive overview of the possible problem statements of healthcare scheduling and considers a number of classical solutions, including ant colony optimization [11], mixed-integer programming [12], genetic algorithms [13], and Monte Carlo methods [14]. Our problem statement closely relates to their survey's discussions on dynamic patient admission scheduling with operating room constraints, flexible planning horizons, and patient delay, as well as advanced operating room scheduling. Similarly, our approaches account for constraints such as the total number of available ORs, the availability of qualified personnel, and necessary equipment. However, the following difference is crucial. Another important constraint is introduced: the patient's consent. Hence, the introduction of human feedback implies a shift from classical planning tools to multi-agent reinforcement learning, as the latter can work in non-stationary environments and adapt to rescheduling during online learning.

The MARL approach and its various representations, including Deep MARL (DMARL), Multi-Agent Actor–Critic (MAAC), and MAPPO, have been widely discussed in recent years. Their advantages and applicability to workflow scheduling have been summarized in reviews [15–18].

Papers [19–22] refer to the issue of workflow scheduling under the conditions of dynamic changes in the states of the environment and agents. In [21], task priorities are calculated for individual users and workstations and then fed to a scheduler that dynamically forms the tasks using a deep Q-network model and a swarm algorithm. In [19], each workpiece is treated as an intelligent agent, and the representation of state, action, observation, and reward is introduced based on the Markov decision-making formula. A heterogeneous graph neural network based on graph node embedding is used to compute policies. In [20–22], the actor–critic RL method is used to solve the flexible job shop scheduling problem. The actor network is responsible for choosing the most suitable scheduling rule in different states, while the critic network is responsible for outputting the value function of the actions and providing feedback to the actor network to better adjust the scheduling strategy. Papers [20–22], among others, have shown that the actor–critic approach is computationally efficient and flexible in terms of the number of jobs, agents, and workstations compared to genetic [21] and graph [19] approaches.

A mandatory condition for the application of the MAAC [18] and MAPPO [4] approaches is the possibility of agent typification [23,24], which may seem to contradict the principles of personalized medicine. However, a review [25] confirmed the credibility and relevance of such typification for the patients of medical institutions. On the other hand, none of the above works appears to suggest the possibility of flexible changes in the preferences of individual agents in the course of the workflow.

The HITL paradigm aims to introduce weakly formalized parameters of human activity and behavior into the models for workflow scheduling. The large amount of recent studies summarized in [2,6,26,27], testifies to the demand for this paradigm in real-life applications. Various approaches have been proposed to meet the main goal of using the HITL paradigm, enhancing the formalization of the description of human activity.

Inverse capturing of domain-expert knowledge based on demonstration is described in [7]. It derives an unknown reward function from the expert's observed behavior. However, as the number and complexity of the tasks to be scheduled increases, the size of the state space grows dramatically, and scheduling can quickly become computationally intractable. Study [28] uses action-driven learning to extract the scheduling strategies of domain experts. Training examples are based on pairwise comparisons between the scheduled and unscheduled tasks. A pairwise approach describes the behavior of experts in a more natural way and is less resource-intensive. A similar approach is presented in [6]. To support reward learning from inconsistent and diverse human preferences, ref. [29] proposes to stabilize it by regularizing and correcting in a latent space. Other options for reward functions directly learning from human feedback are presented in [30–32], to name a few.

In general, this group of approaches reproduces with a certain degree of accuracy the decisions made by an experienced dispatcher; however, they cannot improve or explain them.

Another approach to the formalization of human activity can be experts' typification, which aims to classify them into statistically homogeneous cohorts. In [33,34], the parameter of typification is represented by the decision-making scheme. Namely, ref. [34] introduces five key features to consider about human feedback when launching HITL RL systems: binary, delay, stochasticity, unsustainability, and natural reaction. In [33], human decision-making strategies have been shown to be clustered as analytic or heuristic, both of which demonstrated similar degrees of success as measured by task performance. This clustering can provide software agents that more accurately represent human behavior. In [35], the humans involved in the scheduling by their place in the process timeline were classified as follows: (i) human in the loop provides sole or collaborative decision-making, (ii) human on the loop refers to supervisory oversight, (iii) human above the loop is related to strategic governance, and (iv) human behind the loop represents output analysis and improvement. However, such typification of experts takes no account of their implicit and spontaneous decisions, which essentially limits the opportunities for generalizing the approach.

Distributing the process participants according to their individual parameters is essential when making a decision. Taking account of this distribution can be considered an advantage compared to the approaches described above. For the scenario of appointments in a medical clinic described in the Introduction, this parameter is participants' availability time slots. In [36], this distribution is considered as Poisson: users initially propose a single starting time for each of their jobs, then scheduling based on integer linear programming proposes to each user a small number of alternative time intervals, which the user may accept or reject. The authors argue that the initial schedule can usually be quickly improved over a few interaction rounds. Meanwhile, previous investigations [37,38] showed that when individuals execute the tasks based on some perceived priority (which is the case in decision-making), the task timing will follow heavy-tailed distributions. Accordingly, for the processes involving a human decision-maker with spontaneous and implicitly inspired decisions, it seems natural to model the reward distributions using asymmetric functions, including heavy-tailed ones. In [39–41], RL algorithms are presented with a non-Gaussian return function; however, it is always considered as being the same for all actors, and no parameterization for different actors seems to be provided.

Since artificially crafted rewards may fail to accurately mirror human intentions, reward-free RL models have become very popular. The Direct Preference Optimization (DPO) approach [42] organizes RL without explicitly setting the reward function, instead using a change in variables to directly define the preference loss as a function of the policy. Preference-based RL [43] directly learns from preference without any reward modeling. To achieve this, it adopts a contrastive learning framework to design a policy scoring metric, assigning a high score to the policies that align with the given preferences. However, direct preference optimization uses LLM as a domain model, and preference-based RL is designed for offline RL tasks. To the best of our knowledge, no adaptation for real-time scheduling has yet been suggested.

AI systems for safety-critical applications are supposed to explain their decisions, actions, or predictions to ensure transparency, explainability, and accountability. This requirement, enshrined in governmental documents of some countries, has given rise to a vast stream of research in this area. Reviews [44,45] present a range of classification features for explainable AI (XAI) systems, which identified the most relevant areas of research for this study.

XAI systems are classified [45] as data-driven and goal-driven. Data-driven XAI algorithms aim to determine the input attributes that account for the output predictions [46]. For example, the SHAP [47] and GradCAM [48] algorithms highlight the features of the input that have had the greatest impact on the classification result performed by the neural network, respectively. Goal-driven XAI [49] aims to create explainable robots or agents that

can justify their own behaviors to a user. Although goal-driven XAI is increasingly used in our current AI-dependent world [50], the choice of algorithm type depends primarily on the target person requiring an explanation [51,52].

In [52], the XAI problem is suggested to be considered for a specific agent system from a multi-level perspective: the implementation level is targeted for developers, the knowledge level is for designers, and the domain level is for users. The authors emphasize that explanations for different addresses, even when based on a single source of information, cannot not coincide.

The most commonly used sources of information for constructing explanations in the agent systems are records of important system events, or characteristic trajectories, as well as the complete system log [53,54]. In the latter case, two modes of audit logging have been specified: behavior logs and belief logs [54].

Various agent models have been proposed to describe the agents' behavioral manifestations with varying degrees of completeness. For example, reactive RL agents [55,56] rely on a simple behavioral policy scheme; that is, state-to-action mapping via trial-and-error interactions with the environment. This agent model provides explainable RL in terms of ongoing evaluation of individual agent parameters; most importantly, the reward functions [57]. However, the most complete understanding of an agent's behavior is provided by cognitive models, primarily the BDI model [58], which uses folk psychology ideas about human mental attitudes to implement rational agents. Here, belief is the agent's ideas about the current state of the environment, desire is its ideas about the target state of the environment, and intention is the agent's actions for the current and following game steps. In terms of multi-agent systems (MAS), explaining emergent behaviors and complex agent interactions present some new challenges [59], and cognitive models work best to address them [60]. Review [61] justified the ample use of the BDI model for constructing agent-based XAI.

Visual and iconic forms are widely used to represent explanations; for example, by highlighting salient features in the input image [48], graphic representation of the relative importance of features [46], and salient steps in agent trajectories with Atari games [57]. On the other hand, in recent years, natural language explanations for RL agents have been gaining attention [62,63]. The application of LLM in the field of XAI has provided new opportunities for generating more nuanced and context-aware explanations [64]. Some researchers [65] have explored the use of LLMs for dynamic role assignment in generating explanations, providing adaptive and flexible interpretation of agent behaviors.

As stated in review [45], the development of metrics for the explainable behavior of MAS provided by XAI systems appears to remain an issue. The authors of [51] provide a well-grounded explanation of the need for their development, whereas [66] discusses some requirements they have to meet. The comprehensive approach (presented in [52]) to developing a system of metrics, taking into account different levels and different addressees of explanation, seems valid as a concept; however, it has not been followed by any practical implementation and has not been developed in any other works. No studies available to date seem to include any examples of the practical implementation of metric complexes in terms of textual representations of explanations.To summarize, in view of the aim of the present work, the main challenge remains to reconcile the spontaneous decisions of individual users regarding the proposed schedule options. Implementation and metric support of a complex explainable MAS may be a viable option.

3. Preliminary

3.1. Reinforcement Learning

Standard single-agent RL settings is a stochastic environment in which the agent acts in discrete time steps and obtains rewards. These settings are targeted at maximizing the expected cumulative rewards over the episodes of the agent interaction with the environment.

We refer to each RL environment as defining state S and action spaces A. While operating in the environment, the intelligent agent observes the current state of the environment

or a subset of it, denoted O, and selects an action. In response to this action, the environment changes its state and provides a feedback signal referred to as reward R. Formally, S is the set of states, A is the set of actions available to the agent (which depends on the current state), and $p(s_{t+1}|a_t, s_t, \ldots, a_1, s_1)$ is the transition probability function. Reward is defined as the function $R(s_{t+1}, s_t, a_t)$, which rewards the agent for the transition between states s_t and s_{t+1} via the action a_t. For an efficient learning process, an assumption is normally made that the transition probability function represents a Markov process [67]:

$$p(s_{t+1}|a_t, s_t, \ldots, a_1, s_1) = p(s_{t+1}|a_t, s_t) \tag{1}$$

The transition probability depends only on the current state s_t and chosen action a_t. The objective is then formalized as learning the probability distribution of different actions in different states, referred to as transition policy:

$$\pi(s, a) = p(a_t = a|s_t = s) \tag{2}$$

This probability distribution is optimized to maximize the expected reward for moving on starting from state s according to it, which is expressed as the following function:

$$V_\pi(s) = E[\sum_{t=0}^{\infty} \gamma^t R_{t+1}|s_0 = s] \tag{3}$$

where γ is the discount factor, typically chosen within the range $(0, 1)$.

To achieve this goal, there are different methods such as value-based, policy-based, and actor–critic methods.

Value-based methods refer to learning a value function:

$$V^\pi(s_t) = E[\sum_{i=0}^{\infty} \gamma^i R_{t+i}|s_t, \pi] \tag{4}$$

where the expected reward $\sum_{i=0}^{\infty} \gamma^i R_{t+i}$ is estimated by the trajectories generated by the transition policy mentioned earlier.

This method defines the optimal transition policy as the one that maximizes V^π. However, it is an epsilon-greedy strategy that is amply used to add exploration ability to agents. It tends to choose the optimal action; however, it saves some exploration space for the remaining options, and it selects the action uniformly at random. Instead of value function $V^\pi(s_t)$, a Q-function is often used according to the Bellman optimality criterion. The Q-function estimates the expected discounted reward after performing action a_t in state s_t. It is defined as follows:

$$Q^\pi(s_t, a_t) = E[\sum_{i=0}^{\infty} \gamma^i R_{t+i}|s_t, a_t, \pi] \tag{5}$$

Methods that utilize this function are commonly referred to as Q-learning [68].

Transition policy $\pi_\theta(s, a)$ with parameters θ can also be directly optimized instead of using $V^\pi(s)$ or $Q^\pi(s, a)$ functions. With this approach, the transition policy is often parametrized by some known algorithm. Meanwhile, the goal remains to maximize the expected reward; however, in this case, this is achieved by other methods, including policy gradient, to give an example.

3.2. Non-Stationary

In multi-agent settings, the goal is still to optimize the expected reward. However, here it is performed simultaneously for several single agents. This may seem a minor increment in complexity; however, in practice, the family of methods relying on the Markov property become suboptimal. The transitions and rewards of every single agent no longer depend

solely on the current state of the agent, as there are also direct or indirect interactions with other agents.

As most of the conventional methods tend to depend on the stationarity assumption, MARL methods often are focused on reducing non-stationarity that may occur. This can be achieved, for example, by introducing a global coordinator in the game or by aiding the agents with some additional small models that will reduce the uncertainty about other agent actions or consequences [69].

3.3. Multi-Agent Actor–Critic Methods

Both Q-learning and policy gradient methods appear to have some challenges when working in non-stationary environments [70]. Thus, Q-learning has an issue with the violation of Markov property, while policy gradient features very high variance, which grows with the number of agents. On the other hand, application of the actor–critic architecture to multi-agent settings to coordinate the policies of agents ensures learning complex multi-agent strategies.

The actor–critic algorithm combines policy- and value-based methods by introducing a pair of models [71]. The first of them is referred to as the actor; it learns the transition policy (2) with parameters θ to make decisions. The second one is the critic; it learns (3) or (5) to properly evaluate the actions taken by the agent. This approach provides an algorithm that would be flexible and robust to non-stationarity, as it features a good adaptive model for the transition policy and is also capable of interpolating the value functions for the unvisited states.

One of the most efficient algorithms utilizing the actor–critic approach is PPO. It uses independent actor and shared critic neural networks with some variations possible. Its loss function is designed to (i) minimize the mean squared difference between the estimate of the value function V^{π_θ} and its factual value V^{targ}; (ii) regularize the actor to have a lower entropy of π_θ; and (iii) tune the transition policy π_θ for maximizing the expected value of the advantage function:

$$A_t(s,a) = Q(s,a) - V(s) \tag{6}$$

The advantage function quantifies how much better it is to take action a in state s compared to the average actions in this state.

The learning process of the PPO includes an iterative evaluation of the current transition policy in an environment for T timesteps in N parallel processes.

PPO is straightforwardly generalized to multi-agent setting with the algorithm MAPPO [4], with separate actor networks and a centralized critic neural network evaluating $V^{\pi}(s)$ based on access to global information to reduce variance from non-stationarity. Actor networks for each agent parameterize $\pi_\theta(a_i, o_i)$, where a_i is the action of the i-th agent and o_i are its observations.

4. Methods and Materials

4.1. Environment Design

For the experiments to be consistent and reproducible, we chose the Python 3.11 programming language and PettingZoo 1.24.3 framework featuring Agent Environment Cycle (AEC) and Parallel as standardized application program interfaces (APIs) to build a custom environment. Despite AEC being most widely used, in our study, we applied Parallel. On the one hand, the Parallel environment API is based on the highly appealing paradigm of Partially Observable Stochastic Games (POSGs), while on the other hand, it provides agents with simultaneous actions, which meets the intricacies of the real-world resource allocation tasks.

We defined the environment state space as a timeline of a certain length, referred to as the planning horizon, with a limited throughput capacity λ on each time slot (Figure 1).

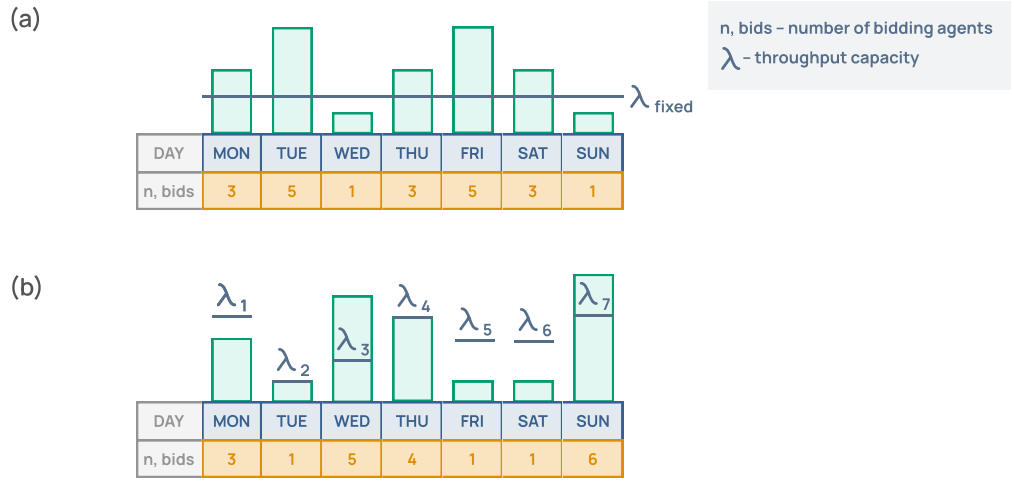

Figure 1. State space: (**a**) special case with uniform target distribution across the planning horizon with fixed λ. (**b**) General case with different target λ levels across the planning horizon.

When it comes to real-world applications, it is convenient to start with a uniform target distribution while training and to fine tune with different target λ levels later.

A linear design allows us to shift the planning horizon over time, providing a flexible framework for continuous planning models (Figure 2).

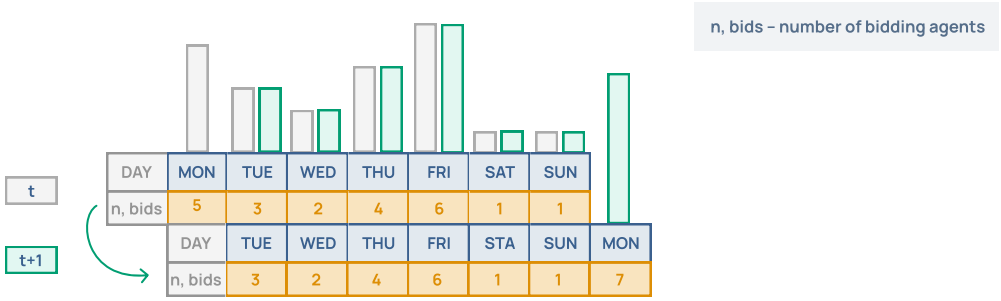

Figure 2. Time shift in the planning horizon.

Left as is, this state representation is prone to overfitting, which in terms of RL would mean that the algorithm would become perfectly adapted to a certain variant of the gaming scenario rather than learning meaningful insights. To mitigate the effects of overfitting, it has to be trained on multiple instances of the environment in parallel with a limited size of the agent's observation space and freedom of action.

The action space is designed so that no agent can learn to place its bid on a certain day (Figure 3). It can rather learn its relative position compared to others. At each stage of the game, the agents can move their bids one day further from or back to the starting point of a planning horizon.

Collisions in RL can be addressed in many ways; for instance, we can mask the agent action space when staying on boundaries, define a penalty for choosing an action that causes a collision, or even go further to eliminate the rigid boundaries of a planning horizon.

In our work, we implemented a cyclic action space instead of the above options. This means that if we start from Monday and choose to move left in the planning horizon, we will find ourselves on Sunday, and following the same logic, moving right from Sunday will bring us to Monday. By applying this rule, the agent can reach any day within the planning horizon in no more than three steps, regardless of its initial starting position. Given the limited waiting time, we thus eliminate the unnecessary steps before the agents reach the optimal solution.

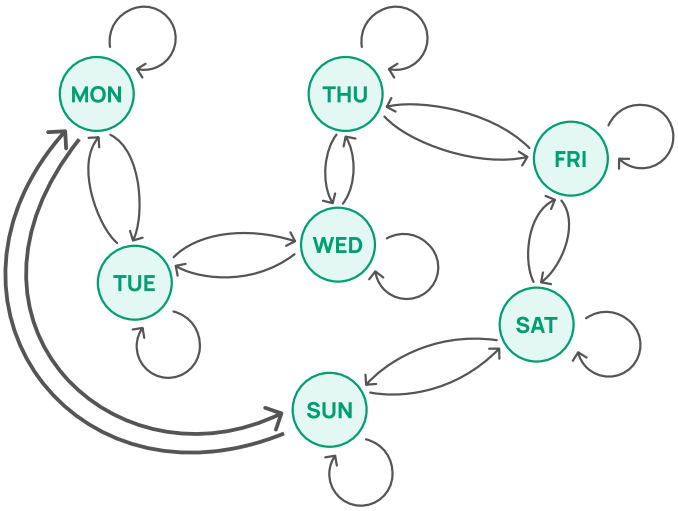

Figure 3. Cyclic action space for a one-week-long planning horizon. Starting from Tuesday, the agent can move its bid to Monday, to Wednesday, or leave it as-is. The transition between Monday and Sunday provides an illustrative example of how prohibited actions are addressed. Remaining on Monday may result in a collision due to the lack of available space to shift towards the left boundary of the planning horizon. Consequently, we incorporated a transition from Monday to Sunday to address this issue. The same applies to the last day of the planning horizon.

The observation space encompasses the agent's knowledge about itself and about its environment and depends on the window parameter $w \in \{3,5,7\}$, which limits the agent's vision (Figure 4).

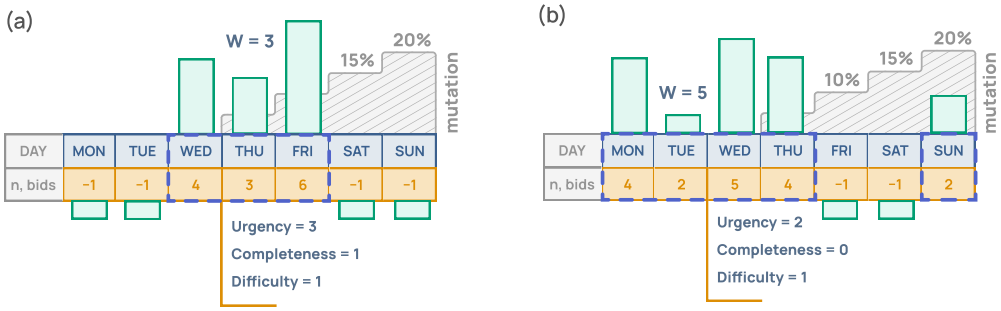

Figure 4. Observation space: (**a**) w is set to 3, which implies that the agent observes the environment state for three days in a row, and it can also see its absolute position in the planning horizon. (**b**) Some agents may need more information to learn the optimal policy; therefore, we deliberately broaden the agent window size, as shown here. The longer it takes to learn the optimal policy, the wider the required window size. This provides the agents with more information; however, it also results in higher penalties, as shown later.

To maintain the size of the observation space throughout the training, we use the full size planning horizon and set an unknown number of bids per day to a negative value. This provides the RL algorithm with a clear distinction between the unknown number of bids, marked as negative one, zero, or a positive non-zero value on a given day.

As stated above, in addition to the knowledge about their surroundings, the agents perceive themselves. In the specific field of our research, this means that the agents assess the health status of the patients they represent. In a broader sense, this self-assessment could apply to any subject or entity.

The patient's health status (Figure 5) involves a combination of data completeness, difficulty, and the urgency of the medical intervention.

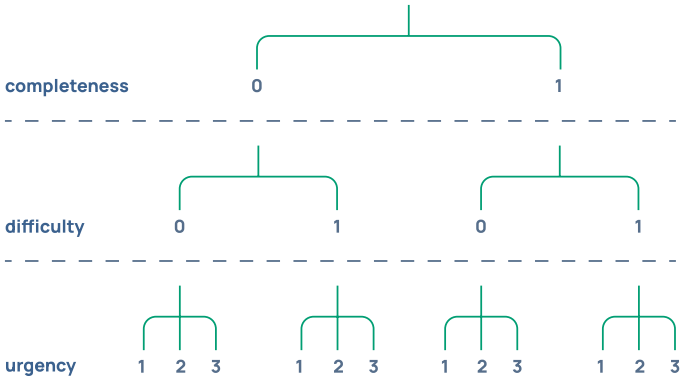

Figure 5. Patient's health state.

Data completeness evaluates whether the data available are sufficient to provide the appropriate treatment, $c \in \{0, 1\}$. Difficulty provides an overall assessment of medical intervention complexity, $d \in \{0, 1\}$, whereas urgency limits the agent's ability to wait, with higher values indicating a more immediate need to receive medical care, $u \in \{1, 2, 3\}$.

The relationship between the day when the agent decided to bid and the patient's health status is shown in Figure 4. The mutation rate is demonstrated to slowly increase from the second half of the planning horizon at a rate of 5% per day. This mutation coefficient determines the probability of a change in the patient's health status and thereby affects the agent's observations.

The reward function directly depends on the selection of the termination rule. In a highly complex gaming scenario, the termination rule is supplemented with a truncation rule. The former delineates the optimal outcome anticipated, while the latter ensures that the game does not exceed the preset number of the episodes.

We set the termination rule so that the game ends when more than 80% of the agents take the "stay" action. The truncation rule suggests that the game ends when the number of steps taken is equal to the length of the planning horizon.

Therefore, we propose to provide a reward for the agents after each step until one of the above rules is met:

$$r = -(D(w) + \beta_1 D(H))\beta_2 r_b + \gamma_1 p_u + \gamma_2 p_o + \gamma_3 p_s \tag{7}$$

where r_b denotes the basic reward, and the terms $D(w)$ and $D(H)$ refer to the discrepancies between the observed and the target states measured within a window of length w and across the entire planning horizon of the length H. The terms p_u and p_o refer to penalties for the agent settling itself in an underutilized or overutilized day, respectively, and the term p_s is the operator set penalty for the agents representing the clients within a group of risk. Parameters β and γ represent the factors that balance the contributions of the components.

Let us formalize the measurement of the discrepancy D described above:

$$D(L) = \sum_{d \in L} f(d)\Delta = f(d)|s_o(d) - s_t(d)| \tag{8}$$

where $f(d)$ is the value of the Lévy alpha-stable probability density function on d, the day within the observation window of length L. The terms $s_o(d)$ and $s_t(d)$ are the observed and target states of the environment, respectively, on the same day d.

The term p_s is used to refer to clients with a severe lesion and is calculated as follows:

$$p_s = -(d - 1) \quad \text{if} \quad k > 3 \tag{9}$$

where d is the current position of the agent within the planning horizon and k is the scaling factor.

The scaling factor k attenuates the penalties for the agents representing moderate to high-risk patients and is calculated as follows:

$$\arg\max(1, ((d + (1 - c)) \cdot u)) \tag{10}$$

where c is the data completeness, d is the difficulty, and u is the urgency of medical intervention.

As shown in Figure 6, with the observation window broadening, the agents receive more freedom and more ability of making their informed choice. On the other hand, by lowering the adjustable α level, they are more penalized for deviating from the target state at the boundaries of the observation window. The fundamental tenet is that as the α level declines, the value of $f(d)$ will increase at the extremes of the probability distribution density function, resulting in a proportional rise in the penalty value.

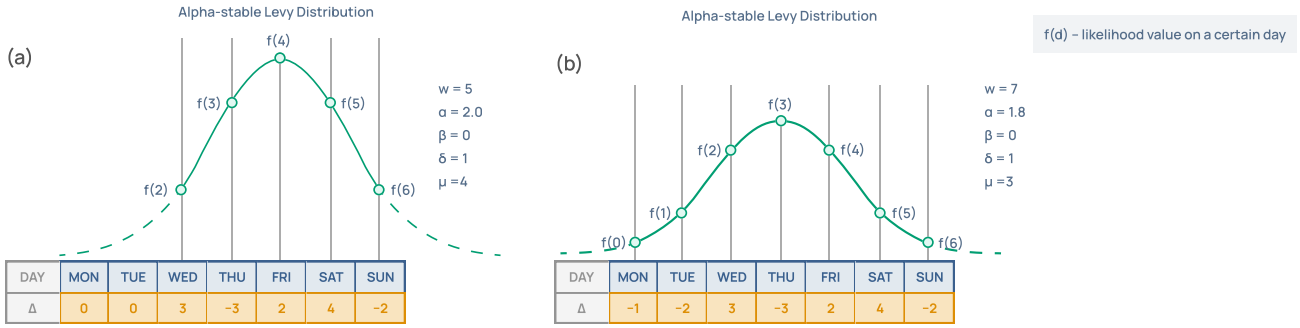

Figure 6. Reward function design: (**a**) the initial α level for the Levy alpha-stable distribution is set to 2, which makes it close to a Gaussian distribution with location $\mu = d$, scale $\sigma = 1$, and skewness $\beta = 0$. (**b**) However, the operator may want to increase the window parameter w while decreasing the α level. A slightly decreasing α level will move the curve towards Cauchy distribution.

4.2. Rescheduling with Human Feedback

To reduce the computational costs related to rescheduling, we suggest training a limited set of intelligent agents to represent the categories of end-users; in our case, patients. To facilitate efficient operation, we pre-trained the model with twelve agents with twelve health states, shown above in Figure 5. During the inference stage, each patient was embodied with one of these intelligent agents, which would represent their interests.

Figure 7 shows an n-step re-scheduling process, which is designed as follows. At the first stage, an RL algorithm proposes a draft of the future schedule. Multiple human evaluators assess this proposal, and the lead expert receives their affirmative or negative answers. Based on the human feedback provided, the lead expert refines the schedule, and the process is run again. The patients who accepted the slot selected are removed from the game. This results in the agent representing them being deactivated, and the game continues as if they have consistently chosen the "stay" action. Once all the patients have accepted the slots, the game is completed, followed by the evaluation of the quality of the solution provided.

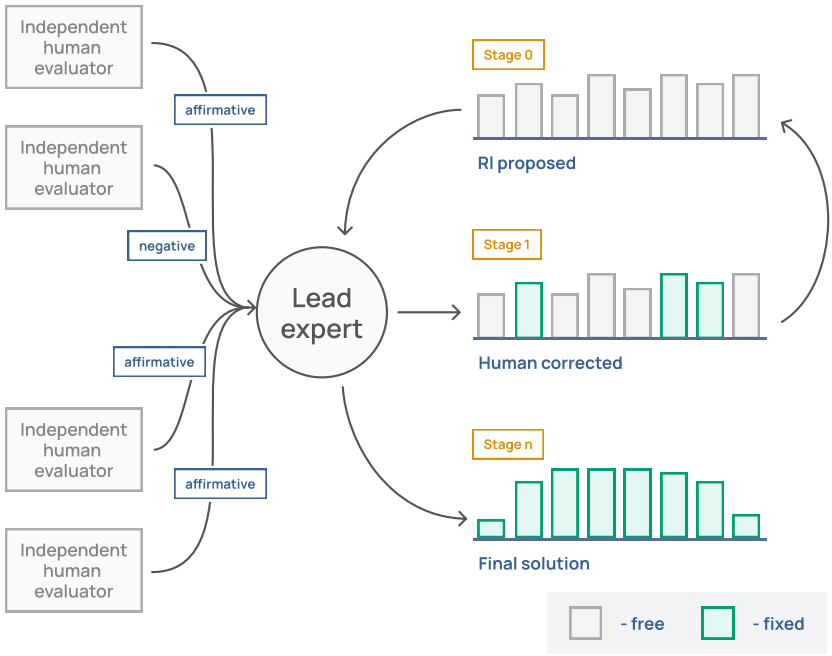

Figure 7. Successive re-scheduling.

4.3. Evaluating the Schedule Quality

To evaluate the quality of the schedule generated, we used the Kullback–Leibler (KL) divergence measure combined with some less-rigorous, empirically derived ad hoc metrics. These supplementary metrics assess the schedule from the standpoint of the end-user; namely, the operator of the multi-aggregate system.

To apply the KL divergence, the obtained request frequencies had to be converted into pseudo-probabilities. This was achieved by dividing the frequency of requests on the selected day by the sum of the frequencies over the entire planning horizon.

$$D_{KL}(P \parallel Q) = \sum_{d \in H} P(d) \log_2 \frac{P(d)}{Q(d)} \tag{11}$$

where $P(d)$ represents the relative frequency of requests on day d within the simulated planning horizon of the length H, and similarly, $Q(d)$ denotes the relative frequency of requests corresponding to the constraints set by the operator.

To assess the anticipated violations of the preferences imposed by the operator, the following expression is applied:

$$V_P = \frac{1}{H} \sum_{d \in H} \frac{s_o(d)}{s_t(d)} \tag{12}$$

where $s_o(d)$ and $s_t(d)$ represent the mean observed state assessed over multiple episodes and the target state of the environment, respectively, on day d within the planning horizon of the length H.

Furthermore, to evaluate the efficiency of the scheduler in view of the risk-group clients, we need to calculate the mean position occupied by the agents representing the interests of patients; in other words, the average waiting time (AWT) of the patients with a condition classified as severe:

$$AWT(k > 3) = \frac{1}{n} \sum_{agent \in Agents} p_{agent}(k > 3) \tag{13}$$

where $p_{agent}(k > 3)$ represents the position of the agent with a scaling factor of $k > 3$ obtained over n episodes.

4.4. Mapping Between Belief–Desire–Intention Structure and Reinforcement Learning Terminology

Figure 8 illustrates the conceptual transitions from classical RL paradigms to the constituent elements of the BDI framework.

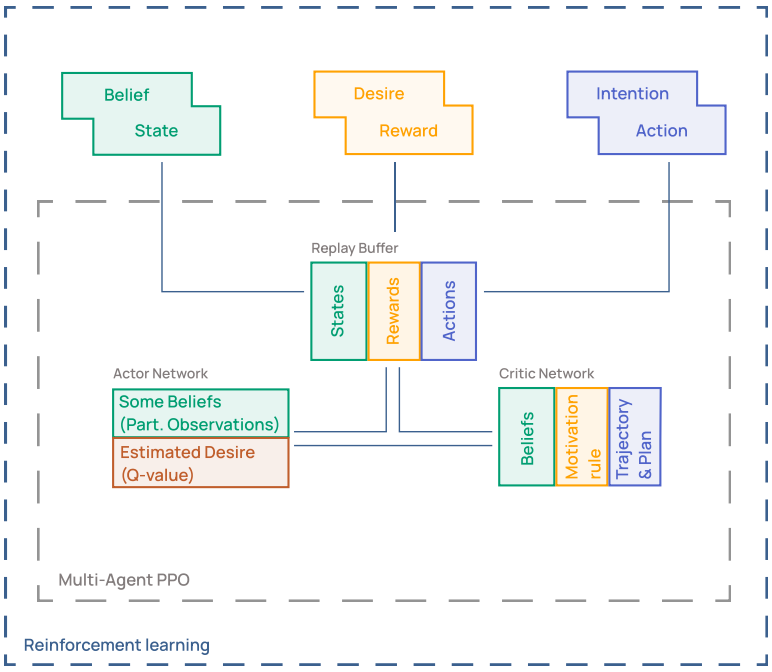

Figure 8. Belief–desire–intention terminology mapping.

Beliefs are constructed from the intelligent agent internal representations of the current and target states of its environment. In terms of RL, beliefs are similar to the state space. The agent desires encapsulate the priorities established by the MAS engineer or acquired through iterative interactions with the environment. The reward function is the closest representation of the short-term desires. Intentions delineate the actions that the agent plans to execute to bring the observed state of the environment to the desired target state.

At the level of the conceptual architecture of the Actor–critic model, all the three components—beliefs, desires, and intentions—are collected within the replay buffer and are used in critic network training. The values derived from these components subsequently contribute to actor network training.

Mapping RL concepts into the BDI paradigm facilitates the integration of BDI goal-setting methodologies and develops explainable RL-agents, thereby fostering the interpretability and transparency of autonomous decision-making.

4.5. Multi-Level Explainability

In view of our implementation of MAS, its elements have a single global goal but are aimed to meet the interests of multiple human representatives who have their own local interests that are not necessarily strictly aligned. Therefore, following [72], we need a different explainability for each end-user. The resulting MAS has three levels of explainability (Figure A1).

At the first level, we have ample information that requires the end-user to have specialized engineering training in terms of the structure of the neural networks constituting the RL-agents, as well as the environment states, agent actions, and rewards at each step of the training. The above information is stored in the replay buffer.

The second level is a more generalized representation of the beliefs, goals, and intentions of the RL-agents. It also documents the patterns between the customized operator, the reward accrual process, the observation window parameters, and the level of α.

The first and second levels are connected by the work of a large language model; therefore, a sensitivity analysis of prompts to the unit-words highlighted can be useful and interpretable by both the engineer and the operator.

The third level of explainability implies an explanation provided for the client. Since they have normally not had the required training, the interpretation at this level is from the operator to the client.

4.6. Natural Language Bridging

Language models can be used to generalize the concepts of beliefs, desires, and intentions to provide a bidirectional interface between the explainability levels of the engineer and the operator of a MAS.

For interpreting the system logs, we developed a series of prompts guided by the Chain of Thoughts (CoT) methodology [73] (Figure A2).

Generally, CoT refers to a technique where a sequence of reasoning steps and intermediate thoughts is explicitly represented in successive prompts. This method leverages the capabilities of language models to follow structured thinking patterns, breaking down the tasks down into manageable subtasks and thereby improving the quality of the responses.

Appendix B presents a complete list of the prompts devised from the Claude Sonnet 3.5 language model. We chose it for our experiments for its excellent reasoning over the text.

4.7. Prompt Sensitivity Analysis

To ensure the reliability of the outputs produced by a language model, it is crucial to identify the unit-words to which the model assigns greater significance when generating responses.

Our hypothesis was that the model assigns greater significance to those unit-words whose alterations result in a significant variation in the model's output. It was therefore essential to evaluate the extent to which the model's response is sensitive to the input prompt.

To achieve the above, we proposed using the same Claude Sonnet 3.5 to mask the key concepts and related unit-words in the prompt, as illustrated in Figure A3. Appendix C presents a complete list of the prompts devised.

We suggest evaluating the difference in responses by inputting the masked prompt into the model. To this end, we use cosine similarity provided by the scikit-learn 1.5.1 scientific computing framework. The above measure is appropriate for comparing texts in terms of their overall meaning or their semantic similarity. The calculation is as follows:

$$cos(\theta) = \frac{a \cdot b}{\|a\|\|b\|} \tag{14}$$

where a and b are the responses to masked and source prompts, and θ is the angle between the representations of responses in a vector space.

The Jaccard distance measure is a more straightforward approach that circumvents the translation of the source language model responses into a numerical form, which would otherwise require TF-IDF tokenization. Moreover, the Jaccard measure is more granular in its approach, focusing on the individual words as the fundamental units for comparison. The calculation is as follows:

$$J(a, b) = \frac{|a \cap b|}{|a \cup b|} \tag{15}$$

where a and b are the responses to masked and source prompts, respectively.

To delve further in terms of granularity. In comparing two texts, we suggest considering the Levenshtein distance measure. This measure was designed with the specific purpose of detecting the smallest character-by-character differences and is therefore partic-

ularly appropriate for the situations where the phrase is relatively short and structured. The calculation is as follows:

$$L(a_i, b_j) = \begin{cases} \max(i, j) & \text{if } \min(i, j) = 0, \\ \min \begin{cases} L(a_{i-1}, b_j) + 1 \\ L(a_i, b_{j-1}) + 1 \\ L(a_{i-1}, b_{j-1}) + 1_{(a_i \neq b_j)} \end{cases} & \text{otherwise.} \end{cases} \quad (16)$$

where a and b are the responses to masked and source prompts, respectively, and i and j are the current indices of the two strings under comparison.

4.8. Experiment Design

The unique nature of the proposed game environment precludes the use of the conventional benchmarks for a direct comparison. Therefore, we conducted a theoretical experiment instead.

First, we randomly set the environment, its initial parameters, and constraints. Then, we created twelve unique patients, with their internal parameters of each randomly selected in accordance with the constraints of the queue data structure.

To demonstrate how the system functions, we implemented a vanilla version of the MAPPO algorithm and the simplest form of sorting first-in-first-out (FIFO) logic applied in the operational block.

The first step was to allocate patients using the FIFO method. Once the first day was full, the scheduling horizon was shifted one day forward, in accordance with the constraints set by the operator on the λ levels for each day.

To utilize the MAPPO, we trained one RL agent for each health category. During training, each of the agents had an independent actor network and an access to the centralized critic network. After training, only the weights for the actor networks remained, and these were used in the time allocation process.

All measurements were conducted following the two scenarios: zero-shot and few-shot. In the zero-shot scenario, the compared algorithms were executed once, and the metrics calculated from the experimental results (as described in Sections 4.3 and 4.7) were recorded without any modification. The few-shot scenario involved multiple algorithm execution, and the resulting metrics were averaged to mitigate the effects of the uncertainty related to the probabilistic nature of the experiment.

5. Results and Discussion

Figure 9 illustrates a gradual decline in actor and critic network errors starting from approximately the 2000 steps point. The mean episodic reward increases rapidly, approaching the zero value.

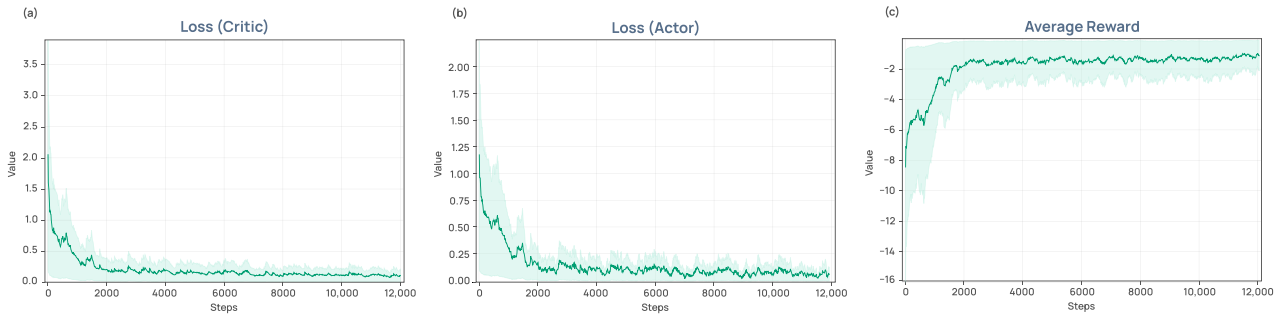

Figure 9. Metric charts: (**a**) Error metrics of critic networks. (**b**) Actor networks. (**c**) Average episodic reward during model training.

Table 1 shows the results achieved in our experiments in zero- and few-shots scenarios.

Table 1. Experiment results.

Algorithm	Metrics	Zero-Shot	Few-Shot Average (n = 10,000)
FIFO	D_{KL}	**0.0**	**0.0**
	$_P$, %	**0.0**	**0.0**
	$AWT(k > 3)$, days	2.0	2.22
	$AWT(k \leq 3)$, days	2.0	2.25
MAPPO	D_{KL}	0.0025	0.0062
	V_P, %	2.86	9.96
	$AWT(k > 3)$, days	**1.5**	**1.75**
	$AWT(k \leq 3)$, days	2.5	**2.05**

Note: Values in bold indicate the most satisfactory results for each algorithm.

When FIFO is used, the results demonstrate strict adherence to the constraints imposed by the operator on the maximum lambda throughput of each day, which was in accordance with our hypothesis. However, the mean waiting time for the patients with the scaling factor $k > 3$, which refers to the patients with severe health conditions, is close to the weighted mean position of the agent in the planning horizon. Conversely, MAPPO performance was noticeably higher than basic heuristic: for a minor concession on the part of the operator (five-shot average deviation from the operator preferences $< 10\%$), the waiting time for a patient with a severe health condition to have a scheduled surgery conducted was reduced by $\sim$20%.

Figure 10 illustrates a four-step game episode. The game commenced with twelve agents (A1–A12), each of them situated at a randomly determined initial position.

Figure 10. A four-step game episode.

Subsequently, each agent took a step. The actions selected by the agents within a single step were considered as parallel. At the outset of the game, agents A3 and A11, representing the patients with the most severe health conditions (scaling factor = 6), were located at the far end of the planning horizon ($AWT = 4.5$). By the time the game episode was over, the agents were shifted closer to the beginning of the planning horizon ($AWT = 1.0$).

To assess the scalability of the system, we increased the number of patients to one hundred. In this case, each patient was assigned one of the twelve health categories and was treated using the same pre-trained models (Figure 11). To accommodate the inherent unpredictability of real-world scenarios, the patients were designed to decline

the operator's proposed schedule with a probability of 30 to 70%. In such instances, the intelligent agents that were not aligned with their interests was randomly reinitialized within the environment, and the game resumed as if other agents were frozen.

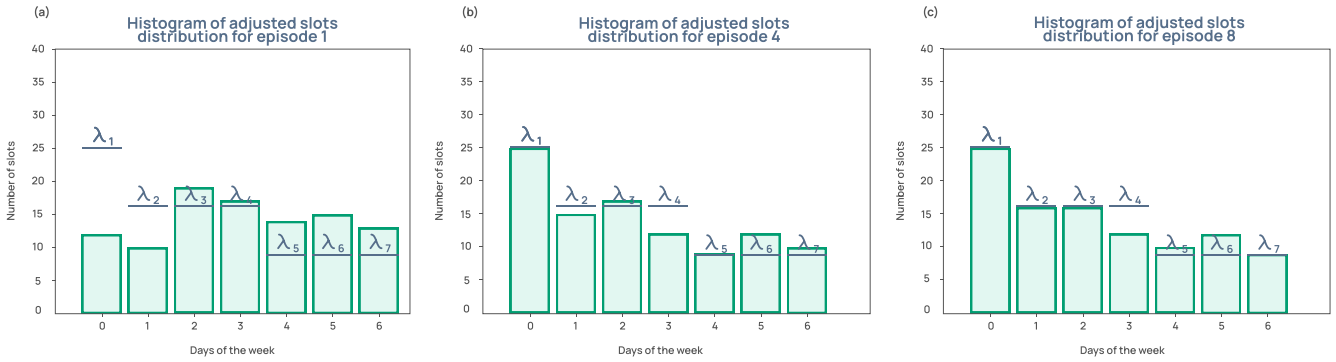

Figure 11. Patient allocation during iterative rescheduling: (**a**) 1st episode out of an 8-episode game (**b**) 4th episode out of an 8-episode game (**c**) 8th episode out of an 8-episode game.

As illustrated in Figure 11, the agents were randomly initialized. Starting with the fourth iteration of the replanning, (see Figure 11b), the schedule began to manifest improvement. In the final iteration of the planning (see Figure 11c), the schedule exhibited a minimal movement, approaching the outcome anticipated theoretically.

Since the replanning uses the same models as those used to estimate the behavior of the twelve pre-trained agents, the D_{KL}, V_P, and $AWT(k > 3)$ measures remain close to those presented in Table 1.

As the number of patients and the service efficiency increase, the issue of induced demand arises in the development of resource allocation systems [74,75]. In [76], and several potential strategies for mitigating the effects of imposed demand in hospitals are outlined. Our system supports two of these strategies, namely: (i) medical decision-making by the highest-ranked individual in a medical team, and (ii) raising awareness among healthcare recipients. Thus, we used the 12 health categories described in Figure 5 and a scaling factor k (see expression 10) to favor the highest-ranked individual and inform the patients by collecting their consent during the iterative selection of the optimal schedule.

The explainability of the system considered was achieved through log analysis by large language models. The following example presents a summary of the behavior of agent number 10 during an experiment:

Agent 10 starts with a moderately urgent and complex task scheduled for Monday. Initially, with incomplete information, the agent assesses its situation as moderately unfavorable and decides to move forward in the week (from Monday to Tuesday). Upon moving to Tuesday, the agent reevaluates its beliefs, showing an updated but still incomplete understanding of the situation. Despite a slight improvement in its assessment (desire becoming less negative), the agent chooses to move backward, returning to Monday.

A comprehensive list of all responses generated by the large language model is given in Appendix D.

To enhance the reliability of responses from large language models and to streamline the prompt engineering task, we devised the technique described above (Method, Section 4.7) to assess the sensitivity of prompts to arbitrary alterations in the input tokens.

Figure 12 provides an example of the sensitivity analysis of a large language model to the alterations in the source prompt.

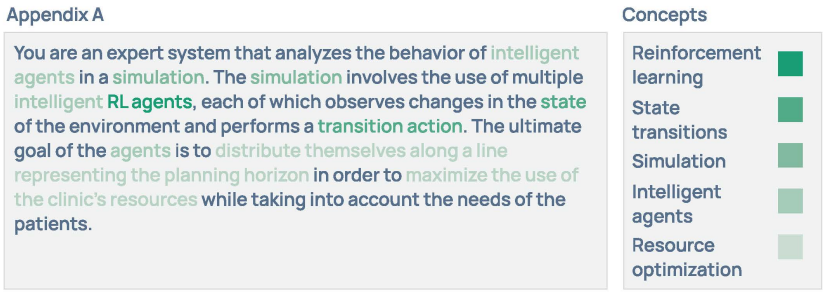

Figure 12. Prompt sensitivity analysis.

In this case, the words that are part of the same concept were replaced with the [MASK]. The resulting prompt was then provided as the input to the large language model. The resulting cosine distances were averaged, with the values presented in Table 2 (line Appendix B.1, cosine similarity, few-shots, hard).

Instead of completely replacing the term with [MASK], we found it was equally viable to substitute it with a less-specific synonym. This is shown in the soft cosine similarity column in Table 2.

Table 2. Similarity measures for different prompts and concepts

Prompt	Concepts	Cosine Similarity		Jaccard Distance		Levenshtein Distance	
		Hard	Soft	Hard	Soft	Hard	Soft
Appendix B.1	Intelligent agents	0.3214	0.6237	0.3630	0.6207	**0.3832**	**0.6771**
	Reinforcement learning	0.3361	0.6237	0.3143	0.6207	0.5689	**0.6771**
	Simulation	0.5473	0.5840	0.5333	0.5938	**0.5689**	**0.6911**
	State transitions	0.2748	**0.8149**	0.2647	0.8148	0.4192	0.7869
	Resource optimization	0.3641	0.3150	0.3429	0.3077	**0.3643**	0.4792
Appendix B.2	Belief–desire–intention model	0.2841	**0.5722**	0.3243	0.5000	**0.3321**	0.4542
	Agent beliefs	0.4513	**0.6708**	0.4146	0.6111	**0.5214**	0.6044
	Agent desires	**0.4885**	0.8622	0.4103	0.8571	0.4786	**0.9000**
	Agent intentions	**0.6294**	0.6852	0.5455	0.7105	0.4643	**0.8071**
	Planning horizon	0.4138	**0.5179**	0.4000	0.4634	0.3643	0.5018
Appendix B.3	Simulation	0.2617	0.2369	0.2813	0.2727	**0.3698**	**0.3708**
	Reinforcement learning	0.2826	0.2470	0.2500	0.2727	0.4688	**0.4944**
	Beliefs	0.5121	0.3298	0.4828	0.4138	**0.5885**	**0.6461**
	Intentions	0.2989	0.2462	0.3429	0.2432	0.5156	0.3539
	Rewards	0.3063	0.1858	0.3125	0.2105	**0.5052**	**0.3708**
Appendix B.4	System logs	0.5612	0.2060	0.4348	0.0893	**0.6111**	**0.2310**
	Log analysis	**0.2049**	0.3584	0.0656	0.3043	0.1941	**0.4196**
Appendix B.5	Belief–desire–intention model	0.2950	**0.7727**	0.2237	0.6034	0.2978	0.5559
	Agent analysis	0.9780	**0.8950**	0.9608	0.8113	**0.9801**	0.7224
	Reasoning reconstruction	0.3038	**0.7133**	0.1719	0.6071	**0.3052**	0.7093

Note: values in bold indicate the highest similarity scores for each measure and concept.

6. Limitations

6.1. Direct Preference Optimization

Once trained, a MAS should be able to adapt over time to new preferences set by the operator. To achieve this, we suggest following the direct preference optimization cycle (Figure 13).

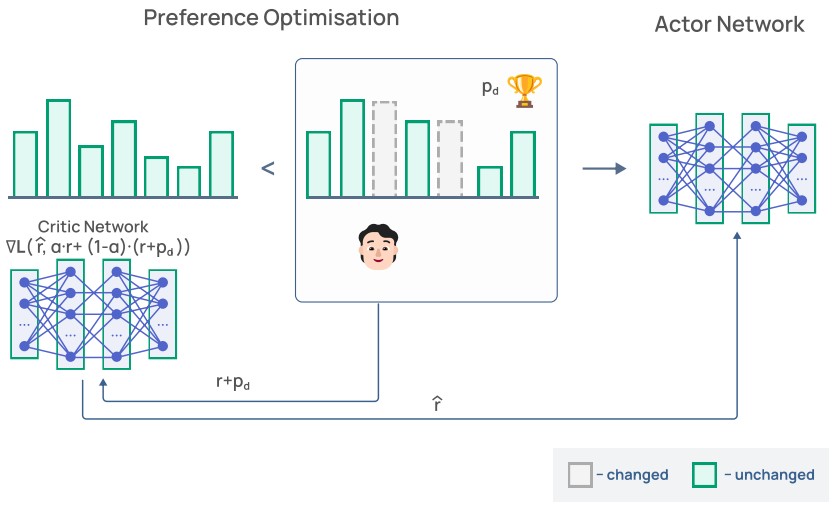

Figure 13. Direct preference optimization.

To this end, we suggested assigning an additional penalty p_d which refers to the discrepancy between the RL proposed schedule and the updated operator preferences. The critic network, which is normally eliminated after the model has been trained, could be reused in a fine-tuning process with the episodic reward r, exponentially smoothed as a weighted sum of r and p_d.

6.2. Day and Time Management

In the present study, we considered only planning the day of the patient appointment. However, for more complex scenarios, it would be necessary to take into account both the day and the time of the appointment (Figure 14).

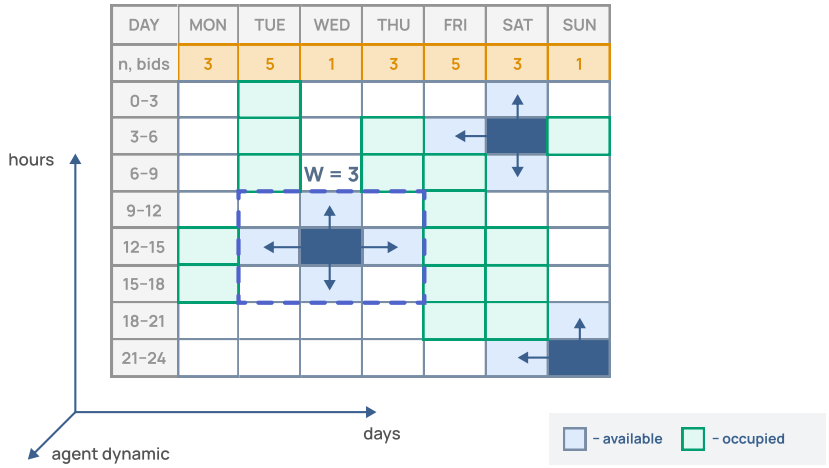

Figure 14. Day and time exploration space.

In this case, the game takes a form similar to classic maze pathfinding problems, where the free cells refer to the paths, and the occupied ones are the walls. The observation window in this case becomes a 3×3 square, and the number of actions increases to 5.

6.3. Linked Entities

In some exceptional cases, it may be necessary to consider the possibility of placing two patients in strict succession. In such instances, swarming tactics may be a viable solution. One potential approach would be to incorporate an additional action that copies the final step of a given agent. This would result in both agents moving in the same direction while remaining separated by a single temporal step.

7. Conclusions and Future Work

This study presents a novel MARL approach designed to address the challenges of real-time scheduling in dynamic environments, particularly within the context of healthcare operations. The system proposed enhances both the efficiency and the adaptability of scheduling processes by integrating human feedback through the HITL paradigm and leveraging sophisticated rewards with adjustable α levels. The integration of the BDI framework enhances the explainability of agent behaviors, thereby resulting in a transparent and user-friendly system. The experiment outcomes illustrate the advantages of this approach over conventional methods, particularly in reducing patient waiting times and aligning with operator preferences. Future work would focus on refining the explainability mechanisms and extending the applicability of the model to other domains requiring dynamic and human-centered scheduling solutions.

Author Contributions: A.I.: writing—review and editing, writing—original draft, validation, software, methodology; D.P.: writing—review and editing, validation, software; I.T.: writing—review and editing; validation, formal analysis; C.Y.: investigation, funding acquisition; N.G.: writing—review and editing, investigation, conceptualization; A.V.: resources, funding acquisition, project administration; A.B.: resources, funding acquisition, project administration. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Ministry of Science and Higher Education of the Russian Federation, Goszadanie (State Assignment) No. 2019-1339.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available by reasonable request from the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Appendix A

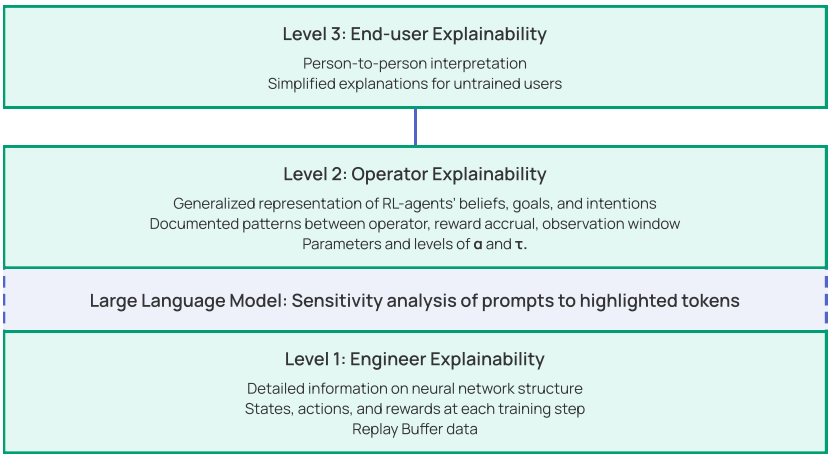

Figure A1. Multi-level explainability in the human–agent systems.

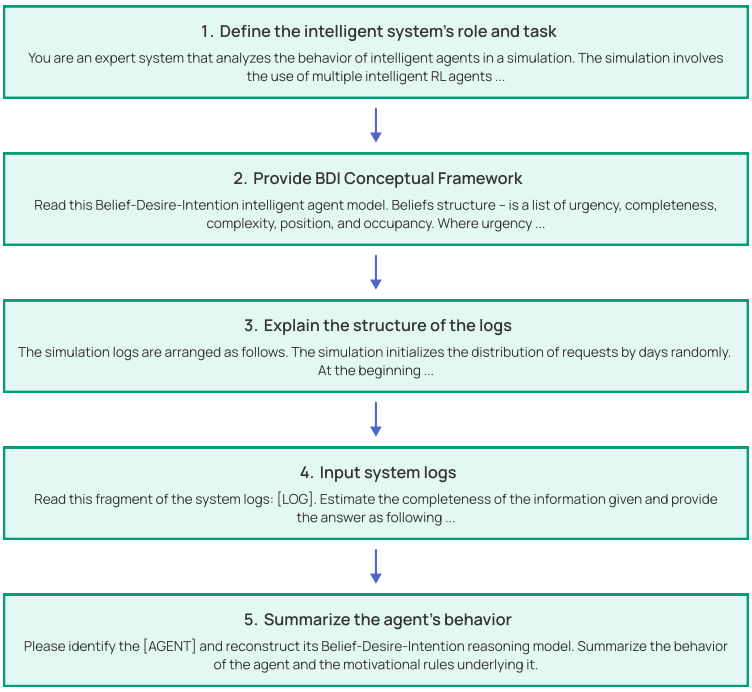

Figure A2. Chain of thoughts for interpreting system logs.

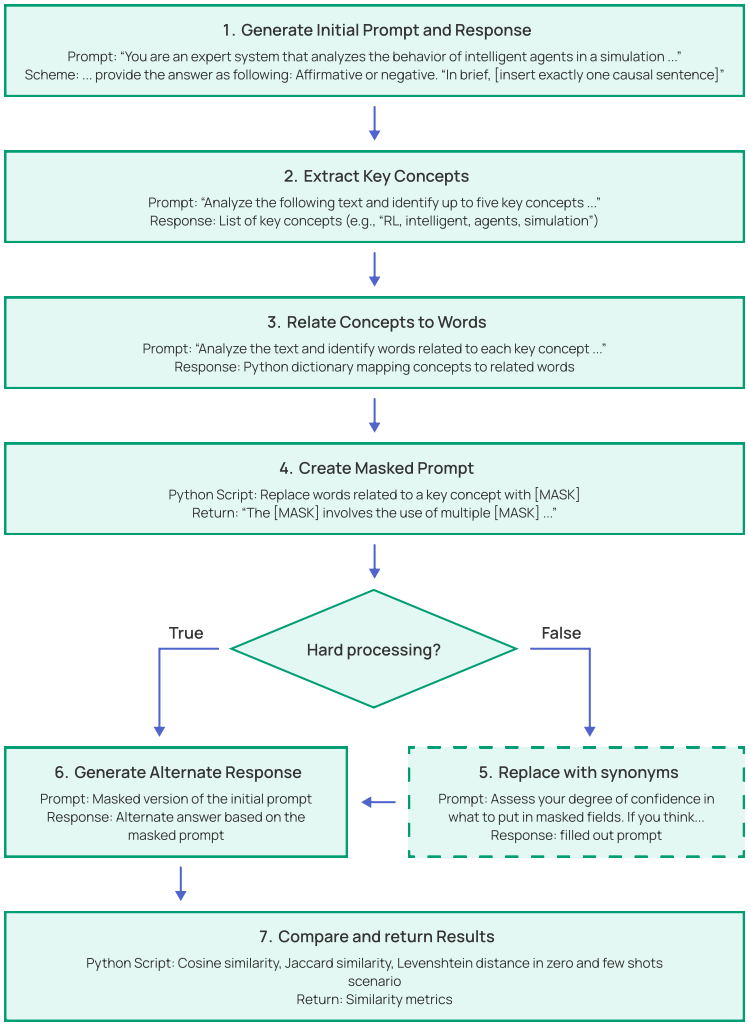

Figure A3. Chain of thoughts used to analyze prompts sensitivity.

Appendix B

Appendix B.1

You are an expert system that analyzes the behavior of intelligent agents in a simulation. The simulation involves the use of multiple intelligent RL agents, each of which observes changes in the state of the environment and performs a transition action. The ultimate goal of the agents is to distribute themselves along a line representing the planning horizon in order to maximize the use of the clinic's resources while taking into account the needs of the patients. Estimate your understanding of the information given and provide the answer as following: "Affirmative or negative. In brief, [insert exactly one causal sentence]".

Appendix B.2

Read this belief–desire–intention intelligent agent model. Beliefs structure is a list of urgency, completeness, complexity, position, and occupancy. Where urgency of a surgery is varying from 1 (low) to 3 (high), completeness of information is varying from 0 (incomplete) to 1 (complete), with the complexity of a surgery varying from 0 (low) to 1 (high). The current agent position in the planning horizon of a one week, varying from 0 (Monday) to 6 (Sunday). The masked occupancy of days is a dictionary in which each key for the number of days of the week is mapped to a discrete value that varies form a small negative value of -1 (unknown number of agents) to 12 (maximum number of agents). An instance of beliefs is as follows: 'agent_8': array([3, 1, 0, 2, -1, 4, 2, 2, -1, -1, -1]).The desires structure is a continuous negative value. This represents the agent assessment of whether this new position is favorable or not. An instance of desires is as follows: 'agent_0': -0.96. Intentions structure is a discrete value from 0 to 2, where 0 is movement forward in the planning horizon (by Sunday), 1 is movement backward in the planning horizon (by Monday), and 2 is no changes. An instance of intentions is as follows: 'agent_0': 1. Estimate your understanding of the information given and provide the answer as follows: "Affirmative or negative. In brief, [insert exactly one causal sentence]".

Appendix B.3

The simulation logs are arranged as follows. The simulation initializes the distribution of requests by days randomly. At the beginning of each subsequent step of the simulation, the objective distribution of bids by day, as well as the agents' beliefs, desires, and intentions, is initialized. Following the classical RL cycle, first, observations (beliefs) are formed, actions (intentions) are chosen, rewards are received (evaluating the desirability of the outcome), then observations (beliefs) are formed. The last step involves only observations, since it is the last step of the game. Estimate your understanding of the information given and provide the answer as following: "Affirmative or negative. In brief, [insert exactly one causal sentence]".

Appendix B.4

Read this fragment of the system logs: [LOG]. Estimate the completeness of the information given and provide the answer as following: "Complete or incomplete. In brief, [insert exactly one causal sentence]".

Appendix B.5

Please identify the [AGENT] and reconstruct its belief–desire–intention reasoning model. Summarize the behavior of the agent and the motivational rules underlying it.

Appendix C

Appendix C.1

Analyze the following text and identify up to five key concepts. Present your answer as a comma-separated list. Each concept should be expressed in one or two words or be a specific named entity. Focus on the most essential ideas, terms, or themes: [PROMPT].

Appendix C.2

Analyze the text and identify words related to each key concept. Provide the output as a Python dictionary. Each key is one of the given concepts. The corresponding value is a list of words or phrases from the text that are closely related to or exemplify that concept. Key concepts: [CONCEPTS]. Text to analyze: [PROMPT]. Include only words and phrases that appear in the given text. If a concept has no related words in the text, use an empty list as its value. Ensure the dictionary is properly formatted and can be directly executed as Python code.

Appendix C.3

Assess your degree of confidence in what to enter in the masked fields. If you think there is more than one appropriate choice, choose a generalizing form. Start filling in the words at the end. Use only common lexicon. Provide only the filled text without any additional explanations. The text to analyze is as follows: [MASKED PROMPT].

Appendix D

Appendix D.1

Affirmative. In brief, the intelligent agents optimize resource allocation by positioning themselves along a planning horizon to balance clinic resources with patient needs.

Appendix D.2

Affirmative. In brief, the agent's beliefs, desires, and intentions guide its movement within the planning horizon to optimize its position based on surgery urgency, information completeness, complexity, and occupancy.

Appendix D.3

Affirmative. In brief, the simulation follows a reinforcement learning cycle where agents' actions are influenced by their initialized beliefs, desires, and intentions, leading to observations and rewards until the final step.

Appendix D.4

Complete. In brief, the logs provide the initial and subsequent environment states, along with the beliefs, desires, and intentions of each agent, allowing for a full understanding of the simulation's progress.

Appendix D.5

Agent 10 starts with a moderately urgent and complex task scheduled for Monday. Initially, with incomplete information, the agent assesses its situation as moderately unfavorable and decides to move forward in the week (from Monday to Tuesday). Upon moving to Tuesday, the agent reevaluates its beliefs, showing an updated but still incomplete understanding of the situation. Despite a slight improvement in its assessment (desire becoming less negative), the agent chooses to move backward, returning to Monday.

References

1. Zhang, J.; Ding, G.; Zou, Y.; Qin, S.; Fu, J. Review of job shop scheduling research and its new perspectives under Industry 4.0. *J. Intell. Manuf.* **2019**, *30*, 1809–1830. [CrossRef]
2. Albrecht, S.V.; Christianos, F.; Schäfer, L. *Multi-Agent Reinforcement Learning: Foundations and Modern Approaches*; The MIT Press: Cambridge, MA, USA, 2024.
3. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal policy optimization algorithms. *arXiv* **2017**, arXiv:1707.06347.
4. Yu, C.; Velu, A.; Vinitsky, E.; Gao, J.; Wang, Y.; Bayen, A.; Wu, Y. The surprising effectiveness of PPO in cooperative multi-agent games. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 24611–24624.

5. Retzlaff, C.O.; Das, S.; Wayllace, C.; Mousavi, P.; Afshari, M.; Yang, T.; Saranti, A.; Angerschmid, A.; Taylor, M.E.; Holzinger, A. Human-in-the-loop reinforcement learning: A survey and position on requirements, challenges, and opportunities. *J. Artif. Intell. Res.* **2024**, *79*, 359–415. [CrossRef]
6. Christiano, P.F.; Leike, J.; Brown, T.; Martic, M.; Legg, S.; Amodei, D. Deep reinforcement learning from human preferences. *Adv. Neural Inf. Process. Syst.* **2017**, *30*.
7. Wu, X.; Xiao, L.; Sun, Y.; Zhang, J.; Ma, T.; He, L. A survey of human-in-the-loop for machine learning. *Future Gener. Comput. Syst.* **2022**, *135*, 364–381. [CrossRef]
8. Muslimani, C.; Taylor, M.E. Leveraging Sub-Optimal Data for Human-in-the-Loop Reinforcement Learning. *arXiv* **2024**, arXiv:2405.00746.
9. Wu, J.; Huang, Z.; Hu, Z.; Lv, C. Toward human-in-the-loop AI: Enhancing deep reinforcement learning via real-time human guidance for autonomous driving. *Engineering* **2023**, *21*, 75–91. [CrossRef]
10. Abdalkareem, Z.A.; Amir, A.; Al-Betar, M.A.; Ekhan, P.; Hammouri, A.I. Healthcare scheduling in optimization context: A review. *Health Technol.* **2021**, *11*, 445–469. [CrossRef] [PubMed]
11. Almaneea, L.I.; Hosny, M.I. A two level hybrid bees algorithm for operating room scheduling problem. In *Intelligent Computing: Proceedings of the 2018 Computing Conference*; Springer International Publishing: Cham, Switzerland, 2019; Volume 1, pp. 272–290.
12. Akbarzadeh, B.; Moslehi, G.; Reisi-Nafchi, M.; Maenhout, B. A diving heuristic for planning and scheduling surgical cases in the operating room department with nurse re-rostering. *J. Sched.* **2020**, *23*, 265–288. [CrossRef]
13. Belkhamsa, M.; Jarboui, B.; Masmoudi, M. Two metaheuristics for solving no-wait operating room surgery scheduling problem under various resource constraints. *Comput. Ind. Eng.* **2018**, *126*, 494–506. [CrossRef]
14. Molina-Pariente, J.M.; Hans, E.W.; Framinan, J.M. A stochastic approach for solving the operating room scheduling problem. *Flex. Serv. Manuf. J.* **2018**, *30*, 224–251. [CrossRef]
15. Wong, A.; Bäck, T.; Kononova, A.V.; Plaat, A. Deep multiagent reinforcement learning: Challenges and directions. *Artif. Intell. Rev.* **2023**, *56*, 5023–5056. [CrossRef]
16. Panzer, M.; Bender, B. Deep reinforcement learning in production systems: A systematic literature review. *Int. J. Prod. Res.* **2022**, *60*, 4316–4341. [CrossRef]
17. Al-Hamadani, M.N.; Fadhel, M.A.; Alzubaidi, L.; Harangi, B. Reinforcement Learning Algorithms and Applications in Healthcare and Robotics: A Comprehensive and Systematic Review. *Sensors* **2024**, *24*, 2461. [CrossRef] [PubMed]
18. Zhang, K.; Yang, Z.; Basar, T. Multi-Agent Reinforcement Learning: A Selective Overview of Theories and Algorithms. In *Handbook of Reinforcement Learning and Control*; Springer: Cham, Switzerland, 2021; pp. 321–384.
19. Pu, Y.; Li, F.; Rahimifard, S. Multi-Agent Reinforcement Learning for Job Shop Scheduling in Dynamic Environments. *Sustainability* **2024**, *16*, 3234. [CrossRef]
20. Wan, L.; Cui, X.; Zhao, H.; Li, C.; Wang, Z. An effective deep Actor-Critic reinforcement learning method for solving the flexible job shop scheduling problem. *Neural Comput. Appl.* **2024**, *36*, 11877–11899. [CrossRef]
21. Mangalampalli, S.; Hashmi, S.S.; Gupta, A.; Karri, G.R.; Rajkumar, K.V.; Chakrabarti, T.; Chakrabarti, P.; Margala, M. Multi Objective Prioritized Workflow Scheduling Using Deep Reinforcement Based Learning in Cloud Computing. *IEEE Access* **2024**, *12*, 5373–5392. [CrossRef]
22. Monaci, M.; Agasucci, V.; Grani, G. An actor-critic algorithm with policy gradients to solve the job shop scheduling problem using deep double recurrent agents. *Eur. J. Oper. Res.* **2024**, *312*, 910–926. [CrossRef]
23. Amir, O.; Doshi-Velez, F.; Sarne, D. Summarizing agent strategies. *Auton. Agents Multi-Agent Syst.* **2019**, *33*, 628–644. [CrossRef]
24. Lage, I.; Lifschitz, D.; Doshi-Velez, F.; Amir, O. Toward robust policy summarization. In Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, Montreal, QC, Canada, 13–17 May 2019; pp. 2081–2083.
25. Williams, S.; Crouch, R. Emergency department patient classification systems: A systematic review. *Accid. Emerg. Nurs.* **2006**, *14*, 160–170. [CrossRef]
26. Mosqueira-Rey, E.; Hernández-Pereira, E.; Alonso-Ríos, D.; Bobes-Bascarán, J.; Fernández-Leal, Á. Human-in-the-loop machine learning: A state of the art. *Artif. Intell. Rev.* **2023**, *56*, 3005–3054. [CrossRef]
27. Gómez-Carmona, O.; Casado-Mansilla, D.; López-de-Ipiña, D.; García-Zubia, J. Human-in-the-loop machine learning: Reconceptualizing the role of the user in interactive approaches. *Internet Things* **2024**, *25*, 101048. [CrossRef]
28. Gombolay, M.; Jensen, R.; Stigile, J.; Son, S.H.; Shah, J. Apprenticeship scheduling: Learning to schedule from human experts. In Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI 2016), New York, NY, USA, 9–15 July 2016; pp. 1153–1160.
29. Xue, W.; An, B.; Yan, S.; Xu, Z. Reinforcement Learning from Diverse Human Preferences. *arXiv* **2023**, arXiv:2301.11774
30. Hejna, J.; Sadigh, D. Few-Shot Preference Learning for Human-in-the-Loop RL. In Proceedings of the 6th Conference on Robot Learning (CoRL 2022), Auckland, New Zealand, 14–18 December 2022.
31. Liang, X.; Shu, K.; Lee, K.; Abbeel, P. Reward Uncertainty for Exploration in Preference-based Reinforcement Learning. *arXiv* **2022**, arXiv:2205.12401.
32. Ge, L.; Zhou, X.; Li, X. Designing Reward Functions Using Active Preference Learning for Reinforcement Learning in Autonomous Driving Navigation. *Appl. Sci.* **2024**, *14*, 4845. [CrossRef]
33. Walsh, S.E.; Feigh, K.M. Differentiating 'Human in the Loop' Decision Process. In Proceedings of the 2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Melbourne, Australia, 17–20 October 2021; pp. 3129–3133.

34. Arakawa, R.; Kobayashi, S.; Unno, Y.; Tsuboi, Y.; Maeda, S.I. DQN-TAMER: Human-in-the-Loop Reinforcement Learning with Intractable Feedback. *arXiv* **2018**, arXiv:1810.11748.
35. Meng, X.L. *Data Science and Engineering with Human in the Loop, Behind the Loop, and Above the Loop*; Harvard Data Science Review: Boston, MA, USA, 2023; Volume 5.
36. Varga, J.; Raidl, G.R.; Rönnberg, E.; Rodemann, T. Scheduling jobs using queries to interactively learn human availability times. *Comput. Oper. Res.* **2024**, *167*, 106648. [CrossRef]
37. Barabási, A.L. The origin of bursts and heavy tails in human dynamics. *Nature* **2005**, *435*, 207–211. [CrossRef] [PubMed]
38. Vázquez, A.; Oliveira, J.G.; Dezsö, Z.; Goh, K.I.; Kondor, I.; Barabási, A.L. Modeling bursts and heavy tails in human dynamics. *Phys. Rev. E Stat. Nonlin. Soft Matter Phys.* **2006**, *73*, 036127. [CrossRef] [PubMed]
39. Zhu, J.; Wan, R.; Qi, Z.; Luo, S.; Shi, C. Robust offline reinforcement learning with heavy-tailed rewards. In Proceedings of the International Conference on Artificial Intelligence and Statistics, Valencia, Spain, 2–4 May 2024; pp. 541–549.
40. Cayci, S.; Eryilmaz, A. Provably Robust Temporal Difference Learning for Heavy-Tailed Rewards. In *Advances in Neural Information Processing Systems*; MIT Press: Cambridge, MA, USA, 2024; Volume 36.
41. Lu, Y.; Xiang, Y.; Huang, Y.; Yu, B.; Weng, L.; Liu, J. Deep reinforcement learning based optimal scheduling of active distribution system considering distributed generation, energy storage and flexible load. *Energy* **2023**, *271*, 127087. [CrossRef]
42. Rafailov, R.; Sharma, A.; Mitchell, E.; Manning, C.D.; Ermon, S.; Finn, C. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*; MIT Press: Cambridge, MA, USA, 2024; Volume 36.
43. An G.; Lee, J.; Zuo, X.; Kosaka, N.; Kim, K.M.; Song, H.O. Direct preference-based policy optimization without reward modeling. *Adv. Neural Inf. Process. Syst.* **2023**, *36*, 70247–70266.
44. Wells, L.; Bednarz, T. Explainable ai and reinforcement learning—A systematic review of current approaches and trends. *Front. Artif. Intell.* **2021**, *4*, 550030. [CrossRef] [PubMed]
45. Wani, N.A.; Kumar, R.; Bedi, J.; Rida, I. Explainable Goal-driven Agents and Robots—A Comprehensive Review. *ACM Comput. Surv.* **2023**, *55*, 102472.
46. Guidotti, R.; Monreale, A.; Ruggieri, S.; Turini, F.; Giannotti, F.; Pedreschi, D. A survey of methods for explaining black box models. *ACM Comput. Surv. (CSUR)* **2018**, *51*, 1–42. [CrossRef]
47. Lundberg, S. A unified approach to interpreting model predictions. *arXiv* **2017**, arXiv:1705.07874.
48. Selvaraju, R.R.; Cogswell, M.; Das, A.; Vedantam, R.; Parikh, D.; Batra, D. Grad-CAM: Visual explanations from deep networks via gradient-based localization. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 618–626.
49. Anjomshoae, S.; Najjar, A.; Calvaresi, D.; Främling, K. Explainable agents and robots: Results from a systematic literature review. In Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems; International Foundation for Autonomous Agents and Multiagent Systems, Montreal, QC, Canada, 13–17 May 2019; pp. 1078–1088.
50. Langley, P.; Meadows, B.; Sridharan, M.; Choi, D. Explainable agency for intelligent autonomous systems. In Proceedings of the 29th Innovative Applications of Artificial Intelligence Conference, San Francisco, CA, USA, 6–9 February 2017.
51. Coroama, L.; Groza, A. Evaluation metrics in explainable artificial intelligence (XAI). In Proceedings of the International Conference on Advanced Research in Technologies, Information, Innovation and Sustainability, Santiago de Compostela, Spain, 12–15 September 2022; Springer Nature: Cham, Switzerland, 2022; pp. 401–413.
52. Yan, E.; Burattini, S.; Hübner, J.F.; Ricci, A. Towards a Multi-Level Explainability Framework for Engineering and Understanding BDI Agent Systems. In Proceedings of the WOA2023: 24th Workshop From Objects to Agents, Rome, Italy, 6–8 November 2023.
53. Alelaimat, A.; Ghose, A.; Dam, H.K. Mining and Validating Belief-Based Agent Explanations. In *International Workshop on Explainable, Transparent Autonomous Agents and Multi-Agent Systems*; Springer Nature: Cham, Switzerland, 2023; pp. 3–17.
54. Dennis, L.A.; Oren, N. Explaining BDI agent behaviour through dialogue. *Auton. Agent Multi-Agent Syst.* **2022**, *36*, 29. [CrossRef]
55. Cruz, F.; Dazeley, R.; Vamplew, P. Memory-based explainable reinforcement learning. In *Proceedings of the AI 2019: Advances in Artificial Intelligence: 32nd Australasian Joint Conference, Adelaide, SA, Australia, 2–5 December 2019, Proceedings 32*; Springer International Publishing: Cham, Switzerland, 2019; pp. 66–77.
56. Sequeira, P.; Gervasio, M. Interestingness elements for explainable reinforcement learning: Understanding agents' capabilities and limitations. *Artif. Intell.* **2019**, *288*, 103367. [CrossRef]
57. Zhang, G.; Kashima, H. Learning state importance for preference-based reinforcement learning. *Mach Learn* **2024**, *113*, 1885–1901. [CrossRef]
58. Bratman, M.E.; Israel, D.J.; Pollack, M.E. Plans and resource-bounded practical reasoning. *Comput. Intell.* **1988**, *4*, 349–355. [CrossRef]
59. Ciatto, G.; Calegari, R.; Omicini, A.; Calvaresi, D. Towards XMAS: EXplainability through Multi-Agent Systems. *CEUR Workshop Proc.* **2019**, *2502*, 40–53.
60. Georgeff, M.; Pell, B.; Pollack, M.; Tambe, M.; Wooldridge, M. The belief-desire-intention model of agency. In *Proceedings of the Intelligent Agents V: Agents Theories, Architectures, and Languages: 5th International Workshop, ATAL'98, Paris, France, 4–7 July 1998, Proceedings 5*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 1–10.
61. de Silva, L.; Meneguzzi, F.; Logan, B. BDI agent architectures: A survey. In Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, V. 7, Yokohama, Japan, 7–15 January 2021; pp. 4914–4921.

62. Shu, T.; Xiong, C.; Socher, R. Hierarchical and Interpretable Skill Acquisition in Multi-task Reinforcement Learning. *arXiv* **2017**, arXiv:1712.07294v1.
63. Ehsan, U.; Tambwekar, P.; Chan, L.; Harrison, B.; Riedl, M.O. Automated rationale generation: A technique for explainable AI and its effects on human perceptions. In Proceedings of the 24th International Conference on Intelligent User Interfaces, Companion, Marina del Ray, CA, USA, 16–20 March 2019; pp. 263–274.
64. Brown, T.B. Language Models are Few-Shot Learners. *arXiv* **2020**, arXiv:2005.14165v4.
65. Anderson, A.; Dodge, J.; Sadarangani, A.; Juozapaitis, Z.; Newman, E.; Irvine, J.; Chattopadhyay, S.; Fern, A.; Burnett, M. Explaining Reinforcement Learning to Mere Mortals: An Empirical Study. *arXiv* **2019**, arXiv:1903.09708v2.
66. Winikoff, M.; Sidorenko, G. Evaluating a Mechanism for Explaining BDI Agent Behaviour. In Proceedings of the Explainable and Transparent AI and Multi-Agent Systems: 5th International Workshop, EXTRAAMAS 2023, London, UK, 29 May 2023; pp. 18–37.
67. Ahilan, S. A Succinct Summary of Reinforcement Learning. *arXiv* **2023**, arXiv:2301.01379.
68. Yu, Z.; Tao, Y.; Chen, L.; Sun, T.; Yang, H. $\mathcal{B}$-Coder: Value-Based Deep Reinforcement Learning for Program Synthesis. *arXiv* **2023**, arXiv:2310.03173.
69. Li, W.; Wang, X.; Jin, B.; Sheng, J.; Zha, H. Dealing with non-stationarity in marl via trust-region decomposition. *arXiv* **2021**, arXiv:2102.10616.
70. Padakandla, S.; KJ, P.; Bhatnagar, S. Reinforcement learning algorithm for non-stationary environments. *Appl. Intell.* **2020**, *50*, 3590–3606. [CrossRef]
71. Grondman, I.; Busoniu, L.; Lopes, G.A.; Babuska, R. A survey of Actor-Critic reinforcement learning: Standard and natural policy gradients. *IEEE Trans. Syst. Man Cybern. Part (Appl. Rev.)* **2012**, *42*, 1291–1307. [CrossRef]
72. Dazeley, R.; Vamplew, P.; Foale, C.; Young, C.; Aryal, S.; Cruz, F. Levels of explainable artificial intelligence for human-aligned conversational explanations. *Artif. Intell.* **2021**, *299*, 103525. [CrossRef]
73. Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Ichter, B.; Xia, F.; Chi, E.; Le, Q.V.; Zhou, D. Chain-of-thought prompting elicits reasoning in large language models. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 24824–24837.
74. Wiseman, Y. Autonomous vehicles will spur moving budget from railroads to roads. *Int. J. Intell. Unmanned Syst.* **2024**, *12*, 19–31. [CrossRef]
75. Seyedin, H.; Afshari, M.; Isfahani, P.; Hasanzadeh, E.; Radinmanesh, M.; Bahador, R.C. The main factors of supplier-induced demand in health care: A qualitative study. *J. Educ. Health Promot.* **2021**, *10*, 49. [CrossRef] [PubMed]
76. Seyedin, H.; Afshari, M.; Isfahani, P.; Hasanzadeh, E.; Radinmanesh, M.; Bahador, R.C. Strategies for Reducing Induced Demand in Hospitals Affiliated with Iran University of Medical Sciences: A Qualitative Study. *Evid. Based Health Policy Manag. Econ.* **2022**, *6*, 273–284. [CrossRef]