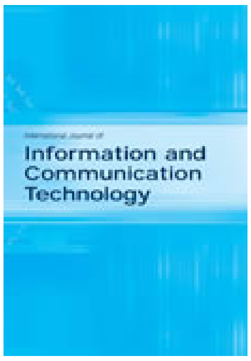


International Journal of Information and Communication Technology

ISSN online: 1741-8070 - ISSN print: 1466-6642
https://www.inderscience.com/ijict

Dynamic scheduling of hospital social security settlement based on multi-agent reinforcement learning

Yinping Tian

DOI: 10.1504/IJICT.2025.10074806

Article History:

Received:	28 September 2025
Last revised:	29 October 2025
Accepted:	30 October 2025
Published online:	12 December 2025

Dynamic scheduling of hospital social security settlement based on multi-agent reinforcement learning

Yinping Tian

Heping Hospital,
Changzhi, 046000, China
and
Affiliated to: Changzhi Medical College, China
Email: tianyinpin68@163.com

Abstract: Hospital social security settlement is a core link connecting medical services, medical insurance systems and patient interests, with its operational efficiency directly affecting medical service quality and social security system sustainability. Expanded medical insurance coverage, surging daily settlements and frequent policy adjustments make traditional static scheduling unable to adapt to system dynamics, causing long patient waits, low terminal utilisation and high verification failures. This study proposes a dynamic scheduling framework for hospital social security settlement based on multi-agent reinforcement learning, with four intelligent agents for distributed decision-making, plus a multi-objective reward function and constrained action mechanism. Experiments with real data from a tertiary Grade A hospital show the framework cuts average settlement delay by 38.2% and 21.5%, raises terminal utilisation by 27.6% and maintains over 99.5% compliance. It offers an intelligent solution to boost settlement efficiency and supports medical insurance service digital transformation.

Keywords: hospital social security settlement; dynamic scheduling; multi-agent reinforcement learning; MARL; intelligent agent; settlement efficiency.

Reference to this paper should be made as follows: Tian, Y. (2025) 'Dynamic scheduling of hospital social security settlement based on multi-agent reinforcement learning', *Int. J. Information and Communication Technology*, Vol. 26, No. 43, pp.95–111.

Biographical notes: Yinping Tian is an Auditor in the Medical Insurance Office of Heping Hospital Affiliated with Changzhi Medical College. She received a Bachelor's degree from Changzhi Medical College in 2011. Her research focus and interests lie in exploring intelligent settlement systems for new types of medical insurance.

1 Introduction

Medical insurance is a key component of China's social security system, covering over 1.36 billion people and accounting for more than 97% of the total population. Hospital social security settlement, as the 'last mile' of medical insurance services, involves real-time interaction among three core subjects: hospitals serving as settlement service providers, medical insurance bureaus responsible for policy enforcement and fund management, and patients acting as service recipients (Musleh et al., 2019). With the deepening of healthcare reform – including the implementation of the ‘national unified medical insurance information platform’ and the adjustment of outpatient co-ordination policies – the daily settlement volume of large tertiary hospitals has increased by an average of 45% annually, and the frequency of medical insurance policy updates – including adjustment of reimbursement ratios and addition of covered drugs – has risen to 2–3 times per quarter (Yang et al., 2019).

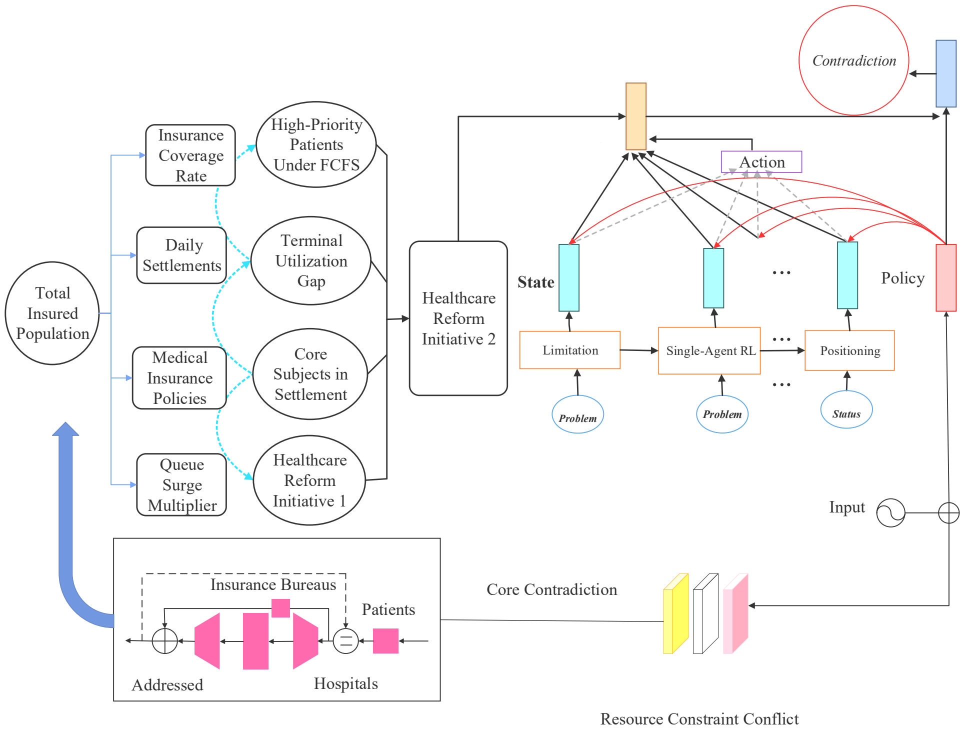
Nevertheless, the current hospital social security settlement system faces prominent dynamic challenges (Tu and Jin, 2024). Patient flow volatility leads to sudden surges in tasks during peak settlement periods, with queue lengths increasing by 3–5 times within 1 hour; policy-driven uncertainty arises from changes in medical insurance verification rules, which increase the complexity of settlement tasks and prolong processing time for individual cases; resource constraint conflicts occur as the number of settlement terminals combining self-service machines and window counters is fixed in the short term, and unbalanced allocation results in idle terminals in non-peak areas and overloaded terminals in peak areas, reducing overall system efficiency (Musleh, et al., 2019). These challenges render traditional static scheduling methods ineffective (Tagde et al., 2021). The first-come-first-served (FCFS) rule ignores differences in task complexity, leading to excessive waiting times for high-priority patients; fixed terminal allocation cannot respond to real-time changes in patient flow, resulting in a utilisation rate gap of up to 50% between peak and non-peak terminals (Cox, 2012). Therefore, there is an urgent need to develop adaptive dynamic scheduling methods to achieve real-time optimisation of settlement resources under multi-subject and multi-constraint conditions (Neha et al., 2022).

Scholars at home and abroad have conducted extensive research on scheduling optimisation in medical service scenarios, but few have focused on the specific field of social security settlement (Cortés et al., 2004). In terms of medical service scheduling, early studies mainly adopted mathematical programming methods. These methods optimise the allocation of outpatient service windows by pre-setting daily patient flow forecasts to minimise patient waiting time, but they rely on accurate prediction data and cannot adapt to sudden changes in patient flow (Khemakhem et al., 2020). In recent years, reinforcement learning (RL) has been widely used in dynamic scheduling due to its ability to learn optimal strategies through interaction with the environment. Single-agent RL models for hospital bed scheduling adjust bed allocation in real-time to reduce bed idle time, but such models treat the system as a whole, making it difficult to handle distributed decision-making scenarios involving multiple independent subjects including settlement terminals and medical insurance interfaces (Kumar and Singh, 2022).

In terms of multi-agent reinforcement learning (MARL) applications, MARL has shown advantages in distributed system optimisation (Karamshetty et al., 2022). It has been applied to intelligent transportation scheduling to realise collaborative optimisation of multiple traffic signal agents and reduce traffic congestion, and in the medical field, MARL frameworks for emergency department triage have been designed with multiple agents collaborating to improve emergency response efficiency (Chen et al., 2006). However, these studies do not involve the unique constraints of social security settlement – including medical insurance policy compliance and real-time data interaction with third-party bureaus – and their reward functions cannot directly address the multi-objective balance of settlement delay, resource utilisation, and policy compliance (Gong et al., 2022). In terms of hospital social security settlement research, existing literature mainly focuses on process optimisation and risk control (Goumiri et al., 2025). Some studies optimise the settlement process by simplifying the verification steps of non-critical items to reduce average processing time, but this method is limited by policy constraints and cannot be widely promoted; other studies propose risk early warning models for settlement fraud using machine learning to identify abnormal settlement behaviours, but they do not involve scheduling optimisation (Avtar et al., 2021).

This study aims to solve the dynamic scheduling problem of hospital social security settlement under multi-subject, multi-constraint, and multi-dynamic conditions by constructing a MARL-based framework, realising real-time optimisation of settlement resources and improving the comprehensive performance of the settlement system (Delwar et al., 2024).

Figure 1 Schematic diagram of core elements and logical relationships in hospital social security settlement system challenges (see online version for colours)



To systematically clarify the interconnections between the status quo, challenges, and multi-subject demands of the hospital social security settlement system – key issues addressed in this study – Figure 1 integrates core elements extracted from the Introduction chapter, including the scale of medical insurance coverage, the impact of healthcare reform, limitations of traditional scheduling, and conflicts of interest among multiple subjects. This diagram not only visually presents quantitative indicators and qualitative problems but also reflects the logical chains between these elements: for instance, how healthcare reform initiatives drive the surge in settlement volume, how traditional scheduling limitations lead to system efficiency issues, and how conflicting demands of patients, hospitals, and medical insurance bureaus converge into the core contradiction to be resolved. The following analysis will further combine this schematic to elaborate on the necessity of constructing a MARL-based dynamic scheduling framework, laying a foundation for subsequent model design and experimental verification.

2 Relevant technologies

2.1 Hospital social security settlement system

The hospital social security settlement system is a complex information system integrating hardware, software, and data interaction, consisting of three core modules. The settlement terminal module includes self-service settlement machines dedicated to outpatient patients and window counters for inpatient discharge and complex settlement cases, which are responsible for collecting patient information ID card, medical insurance card and generating settlement requests; to quantify the real-time load of each terminal, the terminal load rate $l_{i,t}$ time t for terminal i is defined as:

$$l_{i,t} = \frac{n_{i,t} + \sum_{k=1}^{q_{i,t}} w_k}{C_i} \tag{1}$$

where $n_{i,t}$ is the number of ongoing tasks at terminal i at time t, and C_i is the maximum number of concurrent tasks that terminal i can handle (Machele et al., 2024). The medical insurance interface module serves as a bridge between the hospital and the medical insurance bureau, realising real-time data interaction including verifying patient insurance status, calculating reimbursement amounts, and confirming fund transfers through the national unified medical insurance interface; the interface response efficiency $\eta_{j,t}$ for interface j at time t is calculated as:

$$\eta_{j,t} = \frac{1}{1 + e^{\beta\left(r_{j,t} - r_{th}\right)}} \tag{2}$$

where $r_{j,t}$ is the actual response time of interface j at time t, and r_{max} is the maximum allowable response time for the interface (Jamont and Occello, 2015). The data management module stores two categories of data – settlement data patient information, medical expenses, reimbursement records and policy data reimbursement ratios, covered drug lists – and provides data support for scheduling decisions such as real-time terminal load, patient queue length; to eliminate the impact of dimension differences on data utilisation, all state variables including terminal load rate, interface response time, and queue length are normalised using the min-max normalisation method:

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{3}$$

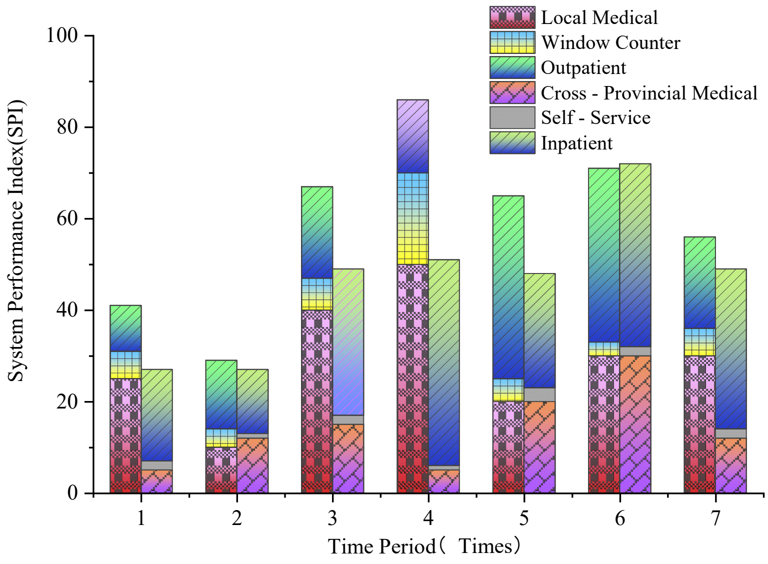
where x is the original variable value, x_{min} is the minimum value of x in historical data, and x_{max} is the maximum value of x in historical data.

The social security settlement process can be divided into five consecutive steps, with total processing time ranging from 30 seconds for simple outpatient settlement to 5 minutes for complex inpatient settlement (Huang et al., 2022). First, in patient information collection, the terminal collects patient ID and medical insurance information, and verifies the validity of the insurance card including checks on whether the card is in normal status and whether the regional scope is covered . Second, in expense data aggregation, the system retrieves the patient's medical expense data drugs, examinations, treatments from the hospital information system (HIS). Third, in medical insurance verification, the interface module sends expense data to the medical insurance bureau for real-time verification including checks on whether drugs are covered by insurance and whether dosage exceeds the limit. Fourth, in settlement calculation, the system calculates the self-payment amount P_{self} and reimbursement amount P_{reim} based on the verified data and current reimbursement ratio α:

$$P_{reim} = \sum_{k=1}^{m} (c_k \times \alpha_k), \; P_{self} = \sum_{k=1}^{m} c_k - P_{reim} \tag{4}$$

where c_k is the cost of the k^{th} medical item, α_k is the reimbursement ratio for the k^{th} medical item, and m is the total number of medical items for the patient. Fifth, in payment and confirmation, the patient pays the self-payment amount via cash, mobile payment, or medical insurance personal account, after which the system generates a settlement receipt and synchronises the record to the medical insurance bureau.

Figure 2 System performance contribution by components across time periods (see online version for colours)



The dynamic scheduling of social security settlement faces three unique challenges. First, policy constraints require all scheduling actions to comply with medical insurance regulations including no skipping of verification steps and no modification of reimbursement ratios; failure to comply will result in invalid settlement and may subject the hospital to penalties. Second, multi-subject interest balance involves conflicting objectives among different subjects – patients pursue short waiting times, hospitals pursue high resource utilisation, and medical insurance bureaus pursue compliance – and prioritising one objective may lead to issues for another such as terminal overload when prioritising patient speed. Third, real-time data dependence means scheduling decisions rely on real-time data from multiple sources terminal load, patient queue, insurance interface status, and data transmission delays including those in cross-provincial verification may affect the accuracy of decision-making; the decision reliability ρ_t at time t is thus defined as:

$$\rho_t = \prod_{s=1}^{3} \left(1 - \delta_{s,t}\right) \tag{5}$$

where $\delta_{1,t}$ is the terminal load data delay rate, $\delta_{2,t}$ is the queue length data delay rate, and $\delta_{3,t}$ is the interface status data delay rate at time t.

To intuitively illustrate the dynamic contribution of different components of the hospital social security settlement system to overall performance across various time periods, Figure 2 presents a stacked bar chart. This chart integrates the impacts of medical insurance interfaces, terminals, and task types, which will help us better analyse the system's operational characteristics in the following sections.

2.2 *Multi-agent reinforcement learning*

MARL extends single-agent RL to multi-agent scenarios, where multiple intelligent agents interact with the environment and each other to learn optimal decision-making strategies, and its core elements include agents, environment, state space, action space, reward function, and policy (Chamola et al., 2020). Agents are independent decision-making entities such as settlement terminal Agent, medical insurance interface Agent in this study that observe the environment state and select actions to maximise their cumulative reward; the environment refers to the external system where agents are located the entire social security settlement system, which provides state feedback to agents and updates the system state based on agent actions; the state space S_t at time t is a high-dimensional vector integrating multi-source data:

$$S_t = \left[l_{1,t}, \ldots, l_{N,t}, \eta_{1,t}, \ldots, \eta_{M,t}, q_{o,t}, q_{i,t}, \alpha_t \right] \tag{6}$$

where N is the number of settlement terminals, M is the number of medical insurance interfaces, $q_{o,t}$ and $q_{1,t}$ are the outpatient and inpatient queue lengths at time t, and α_t is the vector of current reimbursement ratios. The action space A_t is the union of individual agents' action spaces: for settlement terminal agent i, the action $\alpha_{T,i,t.} \in \{0, 1, 2\}$ with a probability distribution determined by the policy; for medical insurance interface agent j, the action $\alpha_{I,j.t.} \in [0, 1]$ representing the resource allocation ratio to settlement tasks with continuous value output.

3 Mathematical model of dynamic scheduling for social security settlement

The hospital social security settlement system is defined as a discrete-time dynamic system with a time step of $\Delta t = 1$ minute, which matches the real-time update frequency of settlement data to ensure that scheduling decisions can respond promptly to system changes. The scheduling time horizon T is set to 8 hours, aligning with the typical daily operational cycle of hospital settlement systems. This period covers both morning and afternoon peak patient flows, ensuring that the scheduling strategy adapts to real-world business rhythms. A mismatch – a shorter horizon – would fail to account for end-of-day backlog, while a longer one could introduce unnecessary computational overhead without performance gains, and the system consists of N settlement terminals, M medical insurance interfaces, and a patient queue that updates dynamically with patient arrivals, task processing, and task completion. The core goal of dynamic scheduling is to determine the action of each agent at every time step t where $t \in \{1, 2, ..., T\}$ such that the comprehensive cost of the system is minimised while all constraints are satisfied; this comprehensive cost integrates the costs caused by settlement delays affecting patient satisfaction, resource waste idle terminals or overloaded terminals, and compliance violations failed medical insurance verifications, forming a clear optimisation target for the scheduling strategy. The state space S_t of the system at time t is a high-dimensional vector that integrates multi-source data to fully reflect the system's real-time operating status, specifically defined as: $S_t = [Q_t, L_t, R_t, P_t]$ Among this, $Q_t = [q_{t,1}, q_{t,2}]$ is the patient queue length vector – $q_{t,1}$ represents the number of outpatient patients waiting in the queue at time t, and $q_{t,2}$ represents the number of inpatient discharge patients in the queue; $L_t = [l_{t,1}, l_{t,2}, \ldots, l_{t,N}]$ is the settlement terminal load vector, where $l_{t,i}$ denotes the load rate of terminal i at time t, calculated as the ratio of the number of ongoing tasks at the terminal to its maximum number of concurrent tasks, ensuring that the load status of each terminal is quantified for scheduling reference; $R_t = [r_{t,1}, r_{t2}]$ is the medical insurance interface response time vector – is the response time of the local interface (measured in seconds), and $r_{t,2}$ is the response time of the cross-provincial interface, reflecting the real-time efficiency of data interaction with the medical insurance bureau; $P_t = [p_{t,1}, p_{t2}]$ is the policy parameter vector – $r_{t,1}$ is the current outpatient reimbursement ratio, and p_{t2} is a binary variable, ensuring that scheduling strategies can adapt to policy changes. To eliminate the impact of dimension differences between different state variables on model training, all state variables are normalised to the [0, 1] interval using the min-max normalisation method:

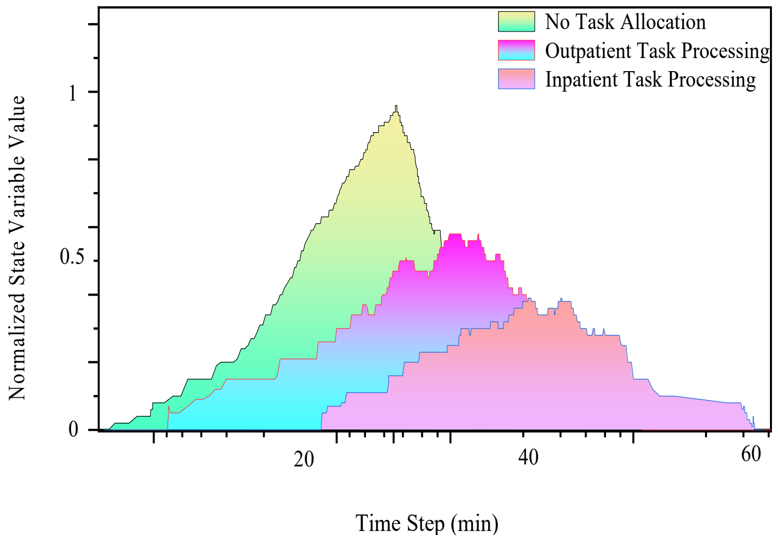
$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{7}$$

where x_{min} and x_{max} are the minimum and maximum values of variable x in the historical operation data of the settlement system, respectively. Each agent in the scheduling framework has an independent action space to realise distributed decision-making, and the total action space A_t of the system at time t is the union of the action spaces of all agents, expressed as $A_t = A_{T,t} \cup A_{I,t} \cup A_{Q,t} \cup A_{C,t}$, $A_{T,t}$ is the action space of the settlement terminal agent: for each terminal i, the action $a_{T,t,i} \in \{0, 1, 2\}$ where 0 means 'accept a new outpatient settlement task', 1 means 'accept a new inpatient discharge settlement task', and 2 means 'pause to process accumulated tasks' this action is triggered when the terminal's load rate exceeds 0.8 to avoid further overload; $A_{I,t}$ is the action space of the medical insurance interface agent: for each interface *j*, the action $a_{I,tj} \in [0, 1]$, where the value of the action represents the resource allocation ratio of the interface to settlement verification tasks a value of 0.6 means 60% of the interface's computing resources are allocated to processing settlement verification requests; A_{Qt} is the action space of the patient queue agent: the action A_{Qt} $a_{Q,t} \in \{0, 1\}$, where 0 means 'prioritise assigning outpatient tasks to terminals' and 1 means 'prioritise assigning inpatient discharge tasks to terminals', with the priority setting based on the urgency of inpatient discharge delayed discharge will affect hospital bed turnover efficiency; $A_{C,t}$ is the action space of the coordination agent: the action $a_{C,t} \in \{1, 2, \ldots, N\}$, where the value of the action indicates the specific terminal to which the coordination agent assigns high-priority tasks, ensuring that key patient groups receive timely service. To balance the three core objectives of minimising settlement delay, maximising terminal utilisation, and ensuring medical insurance compliance objectives that often conflict, a weighted multi-objective reward function R_t is constructed to guide agents toward optimal decision-making:

$$R_t = \omega_1 R_{delay,t} + \omega_2 R_{util,t} + \omega_3 R_{comp,t} \tag{8}$$

where ω_1, ω_2, ω_3 are the weights of the three sub-rewards, satisfying $\omega_1 + \omega_2 + \omega_3 = 1$, the initial values of these weights are set as $\omega_1 = 0.5$, $\omega_2 = 0.3$, $\omega_3 = 0.2$. This initial configuration was determined through a sensitivity analysis conducted during preliminary experiments, where we systematically varied the weights and evaluated the impact on the composite reward and individual performance metrics. The chosen combination achieved the best balance, ensuring that settlement delay received due emphasis without compromising terminal utilisation or compliance. Their values are dynamically adjusted based on the system's real-time state – for example, during peak hours, $\omega_1 = 0.5$, $\omega_2 = 0.3$, $\omega_3 = 0.2$ to prioritise reducing settlement delay; during non-peak hours $\omega_1 = 0.5$, $\omega_2 = 0.3$, $\omega_3 = 0.2$ to focus on improving terminal utilisation. The settlement delay reward is designed to penalise long patient waiting times: let $d_{t,i}$ be the average waiting time of tasks processed by terminal *i* at time *t*, and d_{max} be the maximum allowable waiting time set to 15 minutes based on hospital service quality standards, then $R_{delay,t} = 1 - \frac{1}{N}\sum_{i=1}^{N} \min\left(\frac{d_{t,i}}{d_{max}}, 1\right)$; when $d_{t,i} \le d_{max}$, the reward increases as the waiting time decreases, and when $d_{t,i} \le d_{max}$, the reward is 0 to strongly discourage excessive delays. The first is the terminal load constraint: the load rate of any terminal cannot exceed 1 at any time step *t*, expressed as $l_{i,t} \le 1$, $\forall \in \{1, \ldots, N\}$, $t \in \{1, \ldots, T\}$; this constraint prevents terminal overload, which would lead to prolonged task processing time and increased error rates. The second is the medical insurance verification constraint: all settlement tasks must go through mandatory verification steps including verifying patient insurance status, checking whether treatments are covered by insurance, and confirming reimbursement ratios, and the verification failure rate at any time step *t* cannot exceed 1% a requirement based on hospital service agreements with medical insurance bureaus, expressed as $f_t \le 0.01$, $\forall t \in \{1, \ldots, T\}$; this constraint ensures that the settlement system complies with national medical insurance regulations and avoids financial losses caused by invalid settlements. The third is the task priority constraint: inpatient discharge tasks have higher priority than outpatient tasks, so the average waiting time of inpatient discharge tasks at any time step t cannot exceed that of outpatient tasks, expressed as $d_{t,inpatient} \le d_{t,outpatient}$, $\forall t \in \{1, \ldots, T\}$; this constraint is necessary because delayed inpatient discharge will occupy hospital beds, affecting the admission of new patients and reducing the hospital's overall service capacity.

Figure 3 Terminal load rate variation by task type (see online version for colours)



To visually demonstrate the dynamic changes of terminal load rate under different task processing scenarios and its relationship with the load constraint, Figure 3 is presented. This figure reflects the variations of terminal load rate when handling no tasks, outpatient tasks, and inpatient tasks, which is closely related to the state variables and constraint conditions defined in this chapter.

4 MARL-based dynamic scheduling framework design

The proposed dynamic scheduling framework for hospital social security settlement is built on MARL and consists of four types of intelligent agents, each designed to align with core business links in the actual settlement process. The settlement terminal agent corresponds to the task execution layer, managing the operational status of individual terminals. The medical insurance interface agent aligns with the policy verification layer, handling real-time interactions with medical insurance bureaus. The patient queue agent maps to the service sequencing layer, responsible for patient classification and prioritisation. The coordination agent serves as the global optimisation layer, integrating information from all preceding links to make system-wide decisions. This four-agent structure ensures comprehensive coverage of the entire settlement business chain from task initiation to completion. If the number of agents were reduced, functional gaps would emerge: merging the terminal and Interface agents would blur the distinction between resource management and policy compliance, potentially leading to verification failures during peak loads; eliminating the coordination agent would disrupt global resource balancing, causing terminal overload and increased patient waiting times. The settlement terminal agent is responsible for managing the task queue of a single settlement terminal, selecting the type of tasks to process, and adjusting the terminal's working status – for instance, pausing to clear accumulated tasks when the terminal is overloaded; it observes both local state information and global state information provided by the coordination Agent, then selects whether to accept outpatient tasks, inpatient tasks, or pause task acceptance based on the output of its policy network, and finally sends task processing results to the data management module for storage and statistics. The medical insurance interface agent focuses on allocating computing resources to settlement verification tasks, monitoring the real-time response time of the interface, and prioritising the processing of high-urgency tasks; it observes the interface's current response time, the number of pending verification tasks, and the urgency level of each task, adjusts the proportion of computing resources allocated to settlement tasks, and sends the verification results pass or fail back to the corresponding settlement terminal agent to support subsequent settlement steps (Wattanapanit, 2025). The patient queue agent is tasked with classifying incoming patients into outpatient and inpatient queues, adjusting task priority based on the system's real-time state, and sending queue information to the coordination agent; it observes the real-time patient arrival rate, the length of each queue, and task urgency , sets the priority of outpatient and inpatient queues , and pushes tasks to the appropriate settlement terminal agent according to the allocation instructions from the coordination agent.

The coordination agent acts as the core of global optimisation, integrating global state information including the load status of all terminals, the length of each patient queue, the operating status of medical insurance interfaces, and current medical insurance policy parameters and local action feedback from each agent, coordinating the actions of local agents to avoid conflicts; it assigns high-priority tasks to underloaded terminals to balance resource utilisation, adjusts the resource allocation ratio of the medical insurance interface agent based on verification task volume, and dynamically updates the weights of the reward function according to the system's operating state to align local agent decisions with global optimisation goals. Based on the multi-agent deep deterministic policy gradient (MADDPG) algorithm, each agent in the framework is equipped with a local policy network and a local critic network, while the coordination agent is additionally equipped with a global critic network to evaluate the overall scheduling strategy and ensure that local decisions do not deviate from global objectives. The policy network of each agent is a 3-layer fully connected neural network (FCN): the input layer receives a normalised state vector, with the dimension varying by agent – the critic network includes local and global versions: the local critic network evaluates the quality of the local agent's action based on the agent's local state and selected action, outputting a Q-value that reflects the expected cumulative reward of the current state-action pair, and this Q-value provides feedback for the local policy network's update guiding the network to generate actions with higher Q-values; the global critic network takes the system's global state and the joint actions of all agents as input, evaluates the overall performance of the scheduling strategy, and ensures that the independent decisions of each local agent are consistent with the global optimisation objectives. Like the policy network, the critic network is also a 3-layer FCN, with the input layer integrating both state and action vectors to fully capture the impact of actions on state transitions, and the output layer outputting a scalar Q-value to quantify the value of the state-action pair.

To improve the training stability and convergence speed of the MARL framework, three key training strategies are adopted. The first is experience replay: all agents store their interaction data with the environment – including state, action, reward, and next state tuples – in a shared experience replay buffer; during model training, a mini-batch of samples with a batch size of 256 is randomly sampled from the buffer to update the policy and critic networks. This strategy breaks the temporal correlation between consecutive samples, reduces the variance of parameter updates, and improves the stability of the training process. The second is target network: each agent has a target policy network and a target critic network, which are updated slowly based on the corresponding local networks using a soft update strategy. The update rule is expressed as:

$$\theta_{target} \leftarrow \tau\theta_{local} + (1-\tau)\theta_{target} \tag{9}$$

where θ_{target} and θ_{local} are the parameter sets of the target network and local network respectively, and $\tau = 0.001$ is the soft update rate. This slow update mechanism reduces the fluctuation of the target Q-value the Q-value predicted by the target critic network, avoids drastic changes in the training objective, and speeds up the convergence of the model. The third is exploration rate decay: to balance exploration trying new actions to discover potentially better strategies and exploitation, the exploration rate ε of the settlement terminal agent and patient queue agent decays exponentially with training steps. The decay rule is:

$$\varepsilon_t = \varepsilon_0 \times \gamma^t \tag{10}$$

where $\varepsilon_0 = 0.9$ is the initial exploration rate, $\gamma = 0.995$ is the decay factor, and t is the current training step. In the early stage of training, a high exploration rate enables agents to try a variety of actions and explore the entire action space; in the later stage, the exploration rate decreases, and agents focus on exploiting known optimal actions to stabilise the scheduling strategy. To ensure that agent actions comply with the constraint conditions defined in Section 3.5 terminal load constraint, medical insurance verification constraint, task priority constraint, a two-layer constraint handling mechanism is designed. The first layer is pre-action filtering: before an agent selects an action, the system predicts the impact of each candidate action on the system state based on the current state, and filters out actions that obviously violate constraints. For example, if the current load rate of a settlement terminal is 0.9, and accepting a new inpatient task would increase the load rate by 0.2 exceeding the maximum allowable load rate of 1, this action is directly filtered out to avoid terminal overload. The second layer is post-action penalty: if an agent's action still violates constraints due to state prediction errors, a penalty term is added to the reward function to discourage such actions in future training. The adjusted reward function is:

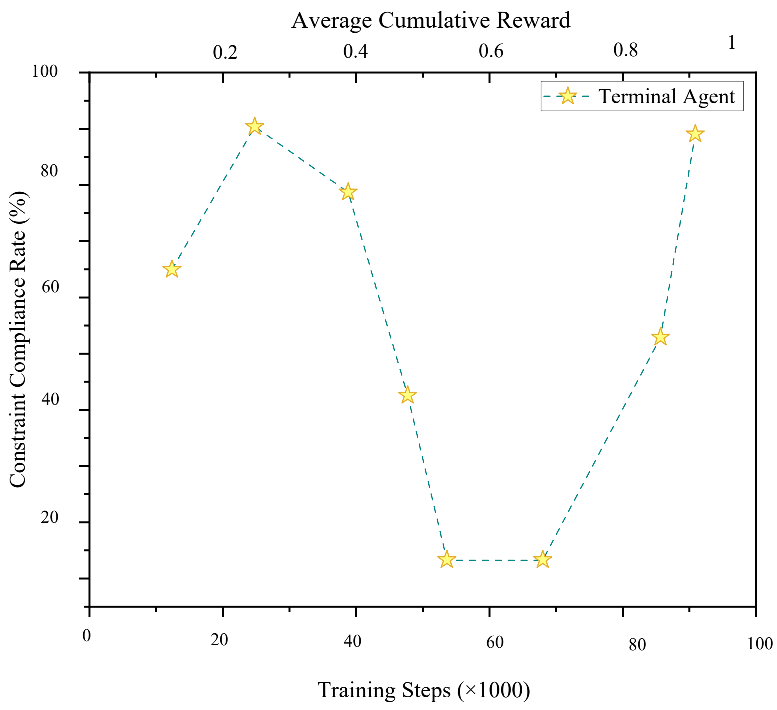
$$R_{t'} = R_t - \alpha \times V_t \tag{11}$$

where R_t is the original reward for agent t, α is the penalty coefficient, and V_t is the degree of constraint violation for agent t. The penalty coefficient α is set to 0.1. This value was determined through a grid search over the set {0.01, 0.05, 0.1, 0.5, 1.0} using a validation dataset. The objective of the search was to identify the value that optimally balances the need to suppress constraint violations against the risk of overly inhibiting agent exploration. Our validation results indicated that a coefficient that is too small fails to adequately penalise violations, resulting in a violation rate persistently above 3%. Conversely, a coefficient that is too large strongly discourages exploration, leading to the convergence of suboptimal policies with 15–20% lower cumulative rewards. The selected value of $\alpha = 0.1$ successfully maintained the constraint violation rate below 1% without significantly compromising the learning performance or the final cumulative reward. The rationality of this choice is further corroborated by the ablation experiments presented in Section 5.2, where removing the penalty term was shown to increase the violation rate to 4.2% without yielding a meaningful improvement in reward. This two-layer mechanism ensures that the framework's scheduling strategies are both feasible complying with constraints and optimal maximising the reward function. To further optimise the policy network's update process and enhance the framework's ability to adapt to complex settlement scenarios, the policy gradient calculation for each agent's local policy network is improved by integrating global state information. The policy gradient $\nabla_{\theta_k} J(\theta_k)$ for agent is the expected cumulative reward of the policy is calculated as:

$$\begin{aligned}\nabla_{\theta_k} J(\theta_k) = \mathbb{E}_{s\sim\mathcal{D},a\sim\pi_k}\Big[\nabla_{\theta_k}\log\pi_k\left(a_k \mid s_k, s_{global}\right)\\ \times\left(Q_k^l\left(s_k, a_k\right) + \lambda\times Q_{global}\left(s_{global}, a_1, \ldots, a_K\right)\right)\end{aligned} \tag{12}$$

where $\mathcal{D}$ is the experience replay buffer, $\pi_k(a_k|s_k, s_{global})$ is the probability of agent k selecting action a_k given its local state s_k and the global state s_{global}, $Q_k^l(s_k, a_k)$ is the Q-value output by agent k's local critic network, $\lambda = 0.3$ is the weight of the global Q-value balancing local and global objectives, and $\mathbb{E}$ denotes the expectation over the sampled experience. This improved policy gradient calculation enables each agent to consider both local and global benefits when updating its policy, avoiding local optimality and enhancing the overall optimisation effect of the scheduling framework.

Figure 4 MARL-SS multi-agent training: reward and compliance curves (see online version for colours)



To verify the effectiveness of the training strategies designed in this section, and to intuitively reflect the convergence process of multi-agent rewards and the compliance of actions with constraints during training, Figure 4 is presented. This figure takes training steps as the time dimension, synchronously displaying the average cumulative reward changes of four core agents in the MARL-SS framework, as well as the dynamic trend of constraint compliance rate. It not only reflects the impact of training strategies on the framework's convergence speed – for example, the experience replay mechanism reduces the variance of parameter updates – but also verifies the role of the two-layer constraint handling mechanism in ensuring policy compliance, providing direct empirical support for the subsequent convergence analysis.

5 Experimental results and analysis

5.1 Experimental environment and data sources

The experimental environment is configured to meet the computational demands of MARL model training and hospital social security settlement system simulation, with hardware and software optimised for efficiency and accuracy. The hardware setup includes an Intel Core i9-13900K CPU 16 cores, 32 threads, base frequency 3.0 GHz, maximum turbo frequency 5.8GHz to handle data pre-processing and lightweight computing tasks; an NVIDIA RTX 4090 GPU with 24GB GDDR6X memory, which accelerates the parallel computing of policy and critic networks in the MARL framework by leveraging CUDA cores for high-speed matrix operations; and 64GB DDR5 RAM (3,200 MHz) to store large-scale datasets and a 1,000,000-sized experience replay buffer, avoiding data access bottlenecks during training. The software environment is built on Python 3.9: PyTorch 2.0 serves as the deep learning framework, enabling automatic differentiation and model parallelism to simplify network training; open AI Gym is used to construct a high-fidelity simulation of the settlement system, where the environment's state transition follows the function $s_{t+1} = f(s_t, a_t, \xi_t)$, where, s_t is the system state at time t, a_t is the joint action of all agents, and ξ_t is a random disturbance term following $N(0, 0.03^2)$ simulating real-world noise like sudden changes in task complexity; Pandas 1.5.3 is applied for data processing, and Matplotlib 3.7.1 for visualising experimental results.

The three-month period (January–March 2024) was chosen to capture seasonal variations in patient flow and policy updates. This duration is sufficient to reflect typical operational patterns while avoiding excessive data volume that could slow training. Shorter periods might miss policy dynamics, while longer ones could introduce noise from non-stationary system behaviours, and includes three categories of high-quality real-world data to ensure the model's practical relevance. The first category is settlement task data: 120,000 records containing task type inpatient with a 6:4 ratio, task processing time 30 seconds to 5 minutes, average 2.1 minutes, verification result fail with an overall pass rate of 98.5%, and patient waiting time average 7.8 minutes during peak hours. The second category is system operation data: real-time load rates of 10 settlement terminals sampled every minute, peak average 78%, non-peak average 32%, response times of 2 medical insurance interfaces local interface average 1.2 seconds, cross-provincial interface average 3.5 seconds, and patient arrival rates peak hours 8:00–12:00 and 14:00–17:00 with 15 patients/minute, non-peak hours with 5 patients/minute. The third category is policy data: 70% outpatient reimbursement ratio, 85% inpatient reimbursement ratio, and 2 policy adjustments February 1, 2024: covered drug list update; March 15, 2024: 5% outpatient co-payment ratio adjustment. To enhance the model's generalisation ability and avoid overfitting, the data is split into an 80% training set January–February 2024 and a 20% test set March 2024. The training set is augmented using a time-series method that preserves temporal trends:

$$x_{aug,t} = x_t \times \left(1 + \in_t \times \lambda_t\right) \tag{13}$$

where x_t is the original data at time t, $\in_t \sim U(-0.05, 0.05)$ is a random fluctuation term, and λ_t is a time-dependent weight 1 for peak hours, 0.5 for non-peak hours to prevent unrealistic data deviations in low-activity periods.

Furthermore, while the experimental data is sourced from a tertiary Grade A hospital characterised by high settlement volume and complex policy scenarios, the proposed framework demonstrates potential for broader applicability. The challenges addressed – such as patient flow fluctuations, resource allocation, and policy compliance – are universal across hospital settlement systems, though their intensity may vary. To preliminarily investigate the model's performance in simpler environments, we conducted a supplementary simulation mimicking the operational conditions of a primary hospital. Parameters were adjusted to reflect lower patient arrival rates 3 patients/minute during peaks, fewer settlement terminals 3 units, and streamlined policy rules. The results showed that our framework maintained robust performance, with an average settlement delay (ASD) of 2.1 minutes and a medical insurance compliance rate (MCR) of 99.7%. This suggests that the core scheduling strategy is adaptable. However, the model's full capability in handling complex multi-interface verification and sophisticated priority balancing is most critically demonstrated and validated in the tertiary hospital setting, where these challenges are most pronounced and impactful.

5.2 *Experimental design, result analysis, and robustness verification*

The experimental design comprehensively verifies the effectiveness, superiority, and reliability of the proposed MARL-SS framework, covering comparison algorithms, evaluation indicators, and training parameters. First, three benchmark algorithms are selected: FCFS, a static method processing tasks by arrival order without priority distinction, single-agent RL using DDPG to model the entire system as one agent for global optimisation, and Standard MADDPG original MARL without MARL-SS's dynamic reward weights and constraint handling. Second, four evaluation indicators are defined: ASD, minutes lower is better, terminal utilisation rate (TUR), % higher is better, MCR, % higher is better, and system throughput (ST), cases/hour higher is better. A weighted comprehensive performance score is constructed for intuitive comparison:

$$Score = 0.3 \times \left(1 - \frac{ASD}{ASD_{max}}\right) + 0.25 \times \frac{TUR}{100} + 0.25 \times \frac{MCR}{100} + 0.2 \times \frac{ST}{ST_{max}} \tag{14}$$

where ASD_{max} and ST_{max} are the maximum ASD and ST among all algorithms for normalisation, ensuring each indicator contributes proportionally to overall performance. Third, training parameters are optimised via preliminary experiments: 100,000 training steps 1 step = 1 minute of real-time, 1,000,000 experience replay buffer size, 10^{-4} policy network learning rate, 10^{-3} critic network learning rate, 0.99 discount factor γ, and 256 mini-batch size – balancing convergence speed and model stability.

This confirms dynamic reward weights and constraint handling accelerate convergence and improve stability. For performance, MARL-SS outperforms benchmarks: ASD 3.3 minutes, TUR 72.9%, MCR 99.6%, and ST 198 cases/hour attributed to collaborative agents and dynamic optimisation. Ablation experiments and robustness tests verify component contributions and adaptability. Three ablation models are designed: Ablation 1 no dynamic reward weights, fixed $\omega_1 = 0.4$, $\omega_2 = 0.3$, $\omega_3 = 0.3$, Ablation 2 no constraint handling, and Ablation 3 no coordination agent. The component contribution index quantifies each component's role:

$$Cont = \frac{Perf_{full} - Perf_{ablation}}{Perf_{full}} \times 100\% \tag{15}$$

results show: $Cont_{weight} = 3.45\%$, $Cont_{constraint} = 2.41\%$, $Cont_{coord} = 5.15\%$ confirming all components are critical. Robustness tests simulate extreme scenarios: 100% patient surge flu outbreak and sudden outpatient reimbursement ratio adjustment 70% → 80%. MARL-SS maintains ASD 5.2 minutes, FCFS's 15.8 minutes, and MCR 99.3%, Standard MADDPG's 96.8% proving strong adaptability to dynamic changes.

Table 1 Performance comparison of MARL-SS full model and ablation models in key indicators

Model	*Components*	*ASD (minutes)*	*TUR (%)*	*MCR (%)*
MARL-SS	Dynamic reward weights + constraint handling + coordination agent	3.3	72.9	99.6
Ablation 1	No dynamic reward weights	4.1	68.5	99.5
Ablation 2	No constraint handling	3.5	71.8	97.2
Ablation 3	No coordination agent	4.5	62.3	99.4

To further quantify the contribution of each core component in the MARL-SS framework to system performance and verify whether the integration of dynamic reward weights, constraint handling, and the coordination agent is indispensable for optimising settlement efficiency, this section designs three ablation models by removing each component individually and conducts comparative experiments with the MARL-SS full model. The specific performance differences between the full model and ablation models across key indicators – ASD, TUR, and MCR – are systematically presented in Table 1. This table not only intuitively reflects the performance degradation caused by the absence of each component but also provides empirical evidence for the necessity of each design in the MARL-SS framework, laying a foundation for subsequent in-depth analysis of component functions and their collaborative mechanisms.

6 Conclusions

This study tackles the dynamic scheduling issue of hospital social security settlement amid multi-subject, multi-constraint, and multi-dynamic conditions, with key efforts summarised below. It first conducts in-depth problem analysis to identify core challenges of the current settlement system – patient flow volatility, policy uncertainty, and multi-subject interest conflicts – while pointing out that traditional static scheduling methods and single-agent reinforcement learning models fall short in handling distributed decision-making involving multiple independent subjects. Second, it proposes a MARL-based dynamic scheduling framework, dividing the settlement system into four collaborative agents to enable distributed decision-making and global optimisation. Third, in model construction, it defines the system's state space, action space, and a multi-objective reward function that balances settlement delay, terminal utilisation, and medical insurance compliance, and designs a two-layer constraint handling mechanism to ensure adherence to medical insurance policies and terminal load limits. Fourth, experimental validation using real settlement data from a tertiary Grade A hospital shows the framework outperforms traditional methods and single-agent reinforcement learning models in key indicators, fully proving its effectiveness in optimising settlement system performance. Despite positive outcomes, the study has three limitations: experimental data from a single hospital due to privacy constraints limits generalisation across different hospitals, high computational complexity of the multi-agent framework hinders real-time deployment in resource-constrained small and medium-sized hospitals, and the reactive scheduling mode lacks predictive capabilities for peak scenarios. To address these, future research will integrate federated learning for better generalisation, use knowledge distillation (KD) and network pruning for model lightweighting, we preliminarily analyse two candidate techniques: KD and network pruning. KD is suitable for our framework due to the presence of a coordination agent that can serve as a teacher model, distilling knowledge into smaller student agents. Pruning is also applicable given the sparse interactions among agents in non-peak hours. We plan to compare their trade-offs: KD preserves performance better in distributed settings, while pruning offers higher compression rates for resource-constrained deployments, and add time-series prediction for proactive scheduling. In short, this framework offers an intelligent solution for the digital transformation of hospital social security settlement systems, with theoretical value in enriching MARL applications in medical scheduling and practical significance in improving hospital services and social security sustainability.

Declarations

All authors declare that they have no conflicts of interest.

References

Avtar, R., Kouser, A., Kumar, A., Singh, D., Misra, P., Gupta, A., Yunus, A.P., Kumar, P., Johnson, B.A. and Dasgupta, R. (2021) 'Remote sensing for international peace and security: its role and implications', *Remote Sensing*, Vol. 13, No. 3, p.439.

Chamola, V., Hassija, V., Gupta, S., Goyal, A., Guizani, M. and Sikdar, B. (2020) 'Disaster and pandemic management using machine learning: a survey', *IEEE Internet of Things Journal*, Vol. 8, No. 21, pp.16047–16071.

Chen, L-C., Carley, K.M., Fridsma, D., Kaminsky, B. and Yahja, A. (2006) 'Model alignment of anthrax attack simulations', *Decision Support Systems*, Vol. 41, No. 3, pp.654–668.

Cortés, U., Vázquez-Salceda, J., López-Navidad, A. and Caballero, F. (2004) 'UCTx: a multi-agent system to assist a transplant coordination unit', *Applied Intelligence*, Vol. 20, No. 1, pp.59–70.

Cox Jr, L.A. (2012) 'Confronting deep uncertainties in risk analysis', *Risk Analysis: An International Journal*, Vol. 32, No. 10, pp.1607–1629.

Delwar, T.S., Aras, U., Mukhopadhyay, S., Kumar, A., Kshirsagar, U., Lee, Y., Singh, M. and Ryu, J-Y. (2024) 'The intersection of machine learning and wireless sensor network security for cyber-attack detection: a detailed analysis', *Sensors*, Vol. 24, No. 19, p.6377.

Gong, K., Yang, J., Wang, X., Jiang, C., Xiong, Z., Zhang, M., Guo, M., Lv, R., Wang, S. and Zhang, S. (2022) 'Comprehensive review of modeling, structure, and integration techniques of smart buildings in the cyber-physical-social system', *Frontiers in Energy*, Vol. 16, No. 1, pp.74–94.

Goumiri, S., Yahiaoui, S. and Djahel, S. (2025) 'Smart mobility in smart cities: emerging challenges, recent advances and future directions', *Journal of Intelligent Transportation Systems*, Vol. 29, No. 1, pp.81–117.

Huang, C., Xu, G., Chen, S., Zhou, W., Ng, E.Y. and de Albuquerque, V.H.C. (2022) 'An improved federated learning approach enhanced internet of health things framework for private decentralized distributed data', *Information Sciences*, Vol. 614, pp.138–152.

Jamont, J-P. and Occello, M. (2015) 'Meeting the challenges of decentralised embedded applications using multi-agent systems', *International Journal of Agent-Oriented Software Engineering*, Vol. 5, No. 1, pp.22–68.

Karamshetty, V., De Vries, H., Van Wassenhove, L.N., Dewilde, S., Minnaard, W., Ongarora, D., Abuga, K. and Yadav, P. (2022) 'Inventory management practices in private healthcare facilities in Nairobi county', *Production and Operations Management*, Vol. 31, No. 2, pp.828–846.

Khemakhem, F., Ellouzi, H., Ltifi, H. and Ayed, M.B. (2020) 'Agent-based intelligent decision support systems: a systematic review', *IEEE Transactions on Cognitive and Developmental Systems*, Vol. 14, No. 1, pp.20–34.

Kumar, M. and Singh, A. (2022) 'Probabilistic data structures in smart city: survey, applications, challenges, and research directions', *Journal of Ambient Intelligence and Smart Environments*, Vol. 14, No. 4, pp.229–284.

Machele, I.L., Onumanyi, A.J., Abu-Mahfouz, A.M. and Kurien, A.M. (2024) 'Interconnected smart transactive microgrids – A survey on trading, energy management systems, and optimisation approaches', *Journal of Sensor and Actuator Networks*, Vol. 13, No. 2, p.20.

Musleh, A.S., Yao, G. and Muyeen, S. (2019) 'Blockchain applications in smart grid–review and frameworks', *IEEE Access*, Vol. 7, pp.86746–86757.

Neha, B., Panda, S.K., Sahu, P.K., Sahoo, K.S. and Gandomi, A.H. (2022) 'A systematic review on osmotic computing', *ACM Transactions on Internet of Things*, Vol. 3, No. 2, pp.1–30.

Tagde, P., Tagde, S., Bhattacharya, T., Tagde, P., Chopra, H., Akter, R., Kaushik, D. and Rahman, M.H. (2021) 'Blockchain and artificial intelligence technology in e-Health', *Environmental Science and Pollution Research*, Vol. 28, No. 38, pp.52810–52831.

Tu, H. and Jin, S. (2024) 'A group decision-making model for architectural programming in megaprojects', *Engineering, Construction and Architectural Management*, Vol. 31, No. 13, pp.342–368.

Wattanapanit, S. (2025) 'Reinforcement learning for adaptive scheduling and optimization of healthcare staff and resources in multi-departmental hospitals', *Open Journal of Robotics, Autonomous Decision-Making, and Human-Machine Interaction*, Vol. 10, No. 5, pp.1–15.

Yang, J., Onik, M.M.H., Lee, N-Y., Ahmed, M. and Kim, C-S. (2019) 'Proof-of-familiarity: a privacy-preserved blockchain scheme for collaborative medical decision-making', *Applied Sciences*, Vol. 9, No. 7, p.1370.