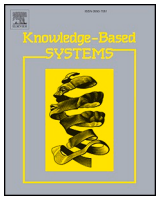
Knowledge-Based Systems 285 (2024) 111381

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys



System-of-systems approach to spatio-temporal crowdsourcing design using improved PPO algorithm based on an invalid action masking

Wei Ding, Zhenjun Ming *, Guoxin Wang, Yan Yan

School of Mechanical Engineering, Beijing Institute of Technology, No. 5 Zhongguancun South Street, Haidian District, Beijing 100081, China

ARTICLE INFO

Keywords:
System-of-systems design
Spatio-temporal crowdsourcing
Improved PPO algorithm
Invalid action masking
Task allocation

ABSTRACT

Spatio-temporal crowdsourcing (STC) is a typical case of complex system-of-systems (SoSs) design, wherein the primary objective is to allocate real-time tasks to suitable groups of workers. Over time, the STC allocation has gradually evolved into a dynamic matching involving three distinct entities: tasks, workers, and workplaces. Aiming at addressing the problems of poor convergence, slow response and sparse actions caused by the spatial complexity and time dynamics of the STC, this paper proposes an improved proximal policy optimization algorithm based on an invalid action masking (IAM-IPPO) for the SoSs design of the STC. Initially, the ternary dynamic matching (TDM) of tasks, workers and workplaces in the STC is described. Furthermore, the STC allocation is formulated as a Markov decision process, with the corresponding definition of state space, action space, and reward mechanism. On this basis, an invalid action masking (IAM) method is mainly introduced to update the policy-based network of proximal policy optimization (PPO), realizing sampling only from valid actions to masking invalid action selection. Subsequently, the algorithmic framework of IAM-IPPO is elaborated upon, and the model is trained to generate an effective allocation scheme. Comparative experiments are conducted on authentic datasets, aiming to assess performance indicators of the presented approach. The findings demonstrate a substantial enhancement in performance for the IAM-IPPO algorithm compared to other baselines, which is helpful in exploring excellent design schemes of the crowdsourcing SoSs, especially in dynamic large-scale cases.

1. Introduction

Crowdsourcing leverages the power of internet technology to allocate an array of tasks among a vast network of users, inviting open collaboration to solicit solutions, innovations, or services from external groups. This approach, in contrast to conventional outsourcing, optimizes the utilization of the public 's free time and harnesses the collective intelligence of non-professionals, achieving high efficiency and low cost.

The widespread adoption of the sharing economy and 5G internet has led to the popularity of spatial crowdsourcing (SC), a decentralized collaboration crowdsourcing model that relies on crowd data. In the SC, task requesters publish tasks *via* specialized platforms, with staff assigned to fulfill these tasks. Its applications span various sectors including traffic optimization [1], urban planning [2], pollution monitoring [3]. However, compared with the SC that focuses on the geographical characteristics of tasks, the spatio-temporal crowdsourcing (STC) emphasizes the necessity for tasks to be executed within precise temporal and spatial constraints, that is, task executors must complete tasks within a specific time window and location area to achieve dynamic matching of timely feedback. Numerous examples of STC can be found in everyday life, such as Uber 's real-time processing of taxi information [4], Gigwalk 's rapid inspection of supermarket products [5], GrubHub 's online matching of ordering service [6], and so on.

In the context of STC, it is indeed essential to consider providing workers with suitable workplaces to perform their tasks. For example, when a task requirement is posted to find a teacher, numerous applications are received. Once the platform completes the matching, there is a need for suitable crowd workplaces where these teachers can conduct their classes. The selection of this workspace thus becomes a critical consideration. To address this need, the traditional bipartite graph matching (BGM) of workers and tasks is extended, which incorporates the concept of a workplace as an additional element in crowdsourcing matching. This extended model gives rise to what is referred to as ternary dynamic matching (TDM) [7], which involves the dynamic matching among tasks, workers, and workplaces.

Furthermore, when viewed through a system-of-systems (SoSs) lens,

* Corresponding author.
E-mail address: zhenjun.ming@bit.edu.cn (Z. Ming).

https://doi.org/10.1016/j.knosys.2024.111381
Received 31 October 2023; Received in revised form 27 December 2023; Accepted 7 January 2024
Available online 8 January 2024

Nomenclature

TS	task set in the STC
WS	worker set in the STC
PS	workplace set in the STC
n	the total number of crowd tasks
m	the total number of crowd workers
s	the total number of crowd workplaces
tc_i	the category of task t_i
wc_j	the category of worker w_j
pc_k	the category of workplace p_k
tb_i	the budget of task t_i
ws_{ij}	the salary of worker w_j for task t_i
tr_i	the radius of the limited publishing range of task t_i
wd_j	the maximal moving distance of worker w_j
N	the number of workers required to contract each crowd task t_i
x_{ij}	the element of the allocation matrix X
pt_k	the number of tasks that have been assigned to the workplace p_k
tw_i^l	the lower bounds of the time window of the task t_i
tw_i^u	the upper bounds of the time window of the task t_i
d_{ijk}	the total distance d_{ijk} of worker moving to the task and workplace
d_{ij}	the total distance d_{ij} between task and worker
d_{ik}	the total distance d_{ik} between task and workplace
tx_i	the horizontal position coordinates of task t_i
ty_i	the vertical position coordinates of task t_i
wx_j	the horizontal position coordinates of worker w_j
wy_j	the vertical position coordinates of worker w_j
px_k	the horizontal position coordinates of workplace p_k
py_k	the vertical position coordinates of workplace p_k
M	the task-worker-workplace matching set
$(t,\ w,\ p)$	a triplet of task t, worker w and workplace p
$MaxSum\ U(M)$	the matching pair with the maximum total utility
U_1	work utility
U_3	distance utility
A	action set
R	reward
γ	discount factor
θ	policy network parameter
π_θ	the current policy function
ε	truncation coefficient
σ	the gradient clipping threshold
$\omega_1,\ \omega_2,\ \omega_3$	the adjustment factors of the three-part reward
E_t	the mean value of the agent 's reward at the current moment t
$\widehat{A}_t$	the advantage function representing the importance sampling
r_t	the ratio between the current policy and the original policy
t_i	the ith crowd task
w_j	the jth crowd worker
p_k	the kth crowd workplace
tl_i	the geographical location of task t_i
wl_j	the geographical location of worker w_j
pl_k	the geographical location of workplace p_k
td_i	the difficulty of task t_i
we_{ij}	the efficiency of worker w_j for task t_i
pa_k	the capacity of workplace p_k
tw_i	the time window of task t_i
wv_j	the moving velocity of worker w_j
U_2	cost utility
U	total utility
S	state set
P	policy
G	the cumulative return
τ	a trajectory
$\pi_{\theta old}$	the original policy function
lr	learning rate

Abbreviation

STC	Spatio-temporal crowdsourcing
TDM	Ternary dynamic matching
IAM	Invalid action masking
NN	Neural network
LSTM	Long short-term memory
NSGA-II	Nondominated sorting genetic algorithm II
PPO	Proximal policy optimization
STC-SoSs	Spatio-temporal crowdsourcing system-of-systems
IAM-PPO	Proximal policy optimization algorithm based on an invalid action masking
IAM-IPPO	Improved proximal policy optimization algorithm based on an invalid action masking
SoSs	System-of-systems
DRL	Deep reinforcement learning
MDP	Markov decision process
GAE	Generalized advantage estimation
DBGM	Dynamic bipartite graph matching
DQN	Deep Q-network
IPPO	Improved proximal policy optimization

the novel framework encompassing the task publishing system, worker matching system, and workplace management system is conceptualized as a spatio-temporal crowdsourcing system-of-systems (STC-SoSs) [8]. For instance, the commonplace taxi service illustrates this model: it begins with the publishing of taxi tasks, proceeds to driver matching, and culminates in passenger delivery to their destination. Each component of this process collectively constitutes a complex SoSs. How to determine the optimal matching relationship among tasks, workers and workplaces is a technical breakthrough to give full play to the effect of ''1+1>2'' in all aspects of STC, and it is also the key to designing a satisfactory STC-SoSs.

Fig. 1 displays the STC-SoSs based on the TDM. The STC platform 's scheduling center will match the standby workers to complete the job once the demander publishes it over the network transmission layer. Then the worker will arrive at the workplace together with the task publisher, and feedback the information to the STC platform [9]. Simply, the design of the STC-SoSs based on TDM can be understood as: while satisfying spatial and temporal constraints, tasks with spatial-temporal attributes are allocated for specific workers and corresponding workplaces are selected. For such design problems, on the one hand, it is necessary to consider matching three distinct entities of tasks, workers and workplaces. On the other hand, the design process of the STC-SoSs is dynamically evolving, that is, tasks are usually published in real time by the demanders, workers' location, status and other information are updated over time, and the availability of workplaces is also constantly changing. Given that tasks, workers, and workplaces might arrive or disappear at any time and from anyplace, so it is indispensable for the designed STC-SoSs to respond immediately and generate matching schemes. Therefore, the primary challenge in designing STC-SoSs lies in formulating a mathematical model that accurately represents the constantly changing nature of the three object types, and devising a reliable algorithm to facilitate the dynamic

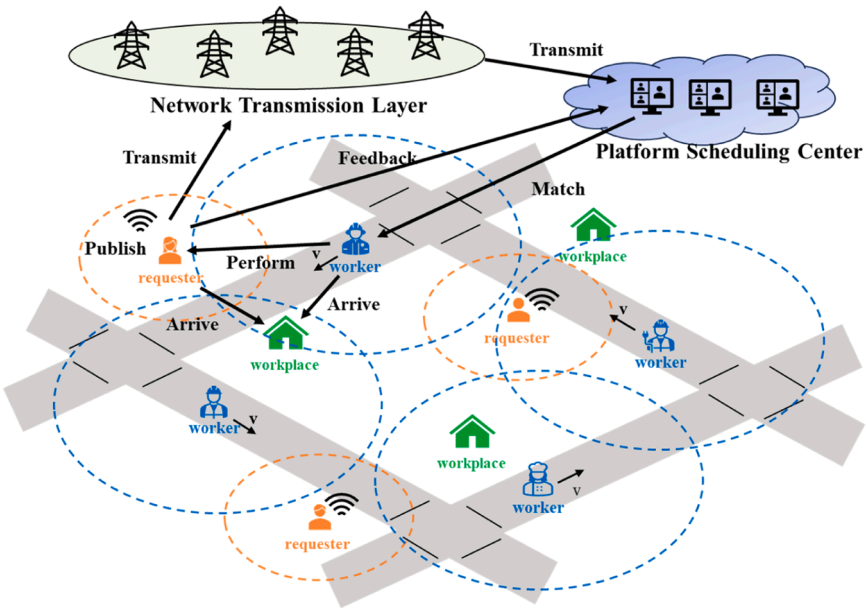


Fig. 1. The system-of-systems of spatio-temporal crowdsourcing based on TDM.

intelligent matching of tasks, workers, and workplaces.

The dynamic matching of tasks, workers and workplaces in the design of STC-SoSs can be simplified as a NP-hard combinatorial optimization problem [10]. Its essence is to select a preference solution set from the candidate solutions to meet the design requirements of maximum total utility and minimum cost in the matching process. Relevant scholars [11–13] have proposed a series of heuristic algorithms to find the optimal solution on several crucial design indicators. However, there are three significant limitations in such methods. First, it is essential to ensure the accuracy and fidelity of the mathematical model used to characterize STC, which is almost impossible in practical applications. Second, heuristic algorithms require multi-iteration obtaining sufficient alternatives to identify an optimum option, which might lead to a decrease in exploration efficiency, and more seriously, the optimization process is difficult to converge. Third, heuristic algorithms lack flexible scalability and do not possess the capability to acquire knowledge from problem-solving or historical data. The optimum solution must be found anew by starting over under the new SoSs if the input changes, such as the quantity of tasks, workers or workplaces.

To overcome the limitations of heuristic algorithms in the design of STC-SoSs, researchers are further exploring new methodologies that are suitable for high-dynamic large-scale scenarios and can generally handle problems with the same structure. Deep reinforcement learning (DRL) [14,15] has been identified as a promising approach to meet the aforementioned requirement. This approach leverages deep neural networks to represent problem models and employs reinforcement learning to iteratively optimize network parameters without the need for explicit training data. Among the various policy gradient algorithms in the DRL domain, proximal policy optimization (PPO) [16] stands out as one of the most extensively utilized. This algorithm is simple to understand, stable in performance, conducive to large-scale training, and can simultaneously handle discrete or continuous action space problems. Compared with traditional linear programming, genetic algorithm and other heuristic approaches, there is evidence that the PPO-based design strategies not only exhibit superior or similar performance, but also can be demonstrated potential for scalability to address intricate large-scale STC problems. What 's more, if the model 's training time is ignored, the PPO algorithm can instantly generate a near-optimal design scheme and realize application generalization by interacting with the environment.

However, in the process of using PPO algorithm to solve the design of STC-SoSs, considering the introduction of workplace elements, the difficulty of setting state space, action space and reward function is further increased. If these settings are unreasonable, it could result in the generation of design schemes that fail to meet the requirements. This challenge is particularly pronounced in large-scale STC scenarios with numerous crowdsourcing elements, where model training time can become excessively prolonged, and model convergence may be adversely impacted. In addition, the conventional PPO policy network is typically designed to output a single action, whereas in STC matching, it is necessary to output multiple discrete sequences that correspond to tasks, workers, and workplaces. As a result, modifications are needed for the policy network to accommodate this requirement.

Considering the real STC application scenario, this paper proposes a novel approach named improved proximal policy optimization based on invalid action masking (IAM-IPPO) for the intelligent design of the STC-SoSs. Inspired by the limitations of the standard PPO algorithm, the IAM-IPPO algorithm adds two core optimization modules. In one module, as the complexity of the STC-SoSs continues to increase, the action matrix gradually becomes sparse during the design process, resulting in a high probability of selecting an unreasonable matching scheme. To counter this, a hard constraint strategy is used to violently mask invalid actions to ensure the convergence of the proposed model and the feasibility of the output scheme. In another module, the Actor-Critic network framework is modified using various tricks, including state normalization, reward scaling, orthogonal initialization, and gradient clipping, to increase model accuracy and reduce training time. This ensures that the policy network 's output satisfies the design requirements of STC-SoSs. The evaluation shows that IAM-IPPO is superior to the standard PPO and heuristic algorithm, especially in the design of larger STC-SoSs. The proposed IAM-IPPO method, which belongs to the category of DRL, can not only obtain matching schemes immediately, but also has the generalization ability to train in small-scale settings and generate in large-scale instances.

The key contributions of this paper can be summarized as follows:

(1) A novel spatio-temporal crowdsourcing (STC) model considering the ternary dynamic matching (TDM) of tasks, workers and workplaces is adopted, and the corresponding mathematical definition is given.

(2) To ensure the convergence of model training, an invalid action masking (IAM) technique is introduced to update the PPO policy network, so that only valid actions are sampled.
(3) By adding two core modules to optimize the standard PPO algorithm, this paper proposes an improved proximal policy optimization based on invalid action masking (IAM-IPPO) algorithm for the design of STC-SoSs, realizing end-to-end model training and dynamic scheme output.
(4) Comparative experiments demonstrate the efficacy of the presented approach on actual datasets. The findings show that the performance of IAM-IPPO is noticeably improved compared with several closely related algorithms.

This is how the remainder of the paper is structured. Related background on SoSs design method, spatio-temporal crowdsourcing, deep reinforcement learning is given in Section 2. Section 3.1 defines the related concepts and mathematical models of TDM in the design of STC-SoSs. Section 3.2 describes the Markov process as it pertains to STC, offering insights into the probabilistic transitions and decision-making processes within the STC environment. In this comprehensive Section 3.3, it first introduces the invalid action masking strategy, then proceeds to construct a policy network model tailored to crowdsourcing challenges, and, finally, culminating in a systematic presentation of the novel IAM-IPPO algorithm. Section 4 is dedicated to empirical validation through contrast experiments and ablation experiments, and discusses the matching results of STC-SoSs design under different conditions.

2. Background and related work

Recognizing the high applicability and relevance of complex SoSs design in real-world scenarios, the crowdsourcing model has garnered significant interest, particularly as a quintessential example in this field. The emergence and growth of novel STC models, fueled by the sharing economy and advancements in Internet technology, further highlight the evolving landscape of this domain. The progression of DRL technology provides a powerful tool for complex SoSs design, which makes it possible to design a satisfactory STC-SoSs design scheme efficiently and accurately. The representative studies from the three domains of system-of-systems design, spatio-temporal crowdsourcing, and reinforcement learning, together with the sites of intersection, will be reviewed in the following.

2.1. System-of-systems design

The SoSs [17] are usually defined as composed of multiple systems with different required capabilities, and jointly collaborate to complete the designated complex tasks according to the corresponding logical relationship. For example, it is assumed that we have a task requirement for fitness exercise in daily life. If we want to complete this task, we need to decompose it into various steps such as publishing tasks on the network, selecting appropriate fitness coaches, and finding the nearest free fitness venues. In the whole process, it involves demand publishing system, coach matching system, venue management system and so on, which together constitute a specific STC-SoSs for completing fitness tasks. SoSs design [17] is a process of architecting the SoSs; hence, it is imperative for a SoS design solution to not only contemplate the inclusion of systems within the SoS, but also the specific tasks to be performed by each system. The SoSs design is an architecting process aiming at obtaining excellent solutions. This process must encompass an understanding of the interactions between individual systems to achieve enhanced overall effects.

The SoSs design scheme is a set of preference solutions generated by combining all candidate systems. Ridolfi et al. [18] introduced a SoSs design approach to support engineering teams by using a low-cost mixed-hypercube solver, and analyzed the results by global sensitivity analysis and response surface analysis. In the aspect of SoSs optimization of air transportation, Marwaha et al. [19] proposed a design approach based on nested optimization and direct search. In Ref. [20], the author formulated a SoSs architecture process that considers the distribution, heterogeneity and independence of information fusion system, and performed modeling and evaluation based on multi-agent and experimental design, respectively. In the industrial field, Carnevale et al. [21] constructed an optimal allocation SoSs for pollutant monitoring sensors, which was formalized as a variant of the well-known set coverage problem and solved by means of both a mixed-integer linear programming solver and genetic algorithms. Although the above method can quickly obtain several feasible alternatives, it heavily relies on the knowledge and experience of domain experts, resulting in more computational time and labor costs.

To improve the design accuracy of SoSs architecture, relevant researchers have simplified it into an NP-hard combinatorial optimization problem. Given multiple participating systems, heuristic algorithms are applied to determine the preferable design scheme by exploring the SoSs design space. In Ref. [22], the design of SoSs based on resilience heuristics was studied from the perspective of forestry cases. In the field of crowdsourcing, Ceschia et al. [12] used greedy local search algorithm to solve the problem of optimizing human intelligent task construction, and showed its scalability to complex crowdsourcing SoSs settings. These heuristic algorithms in the above literatures all acquire satisfactory design schemes through iterative solutions. The tediousness lies in the design of target coding and search structure for specific scenarios, and the construction of dynamic large-scale SoS designs still presents challenges in meeting the demands of timeliness and scalability. What 's more, a notable limitation of heuristic algorithms is that any change in problem elements necessitates starting the search process from scratch. This aspect can significantly limit their practicality and efficiency, especially in dynamic environments where frequent adjustments to the system elements are common.

The DRL algorithm has the advantages of autonomous online learning, complex task processing, efficient design optimization and dynamic large-scale transfer, improving the efficiency and performance of SoSs design. Raman et al. [23] combined reinforcement learning with evolutionary algorithms. On the one hand, DQN-based algorithms were used to solve task planning problems, and on the other hand, we employed improved genetic algorithms to optimize the architecture according to the output of task planning. Lin et al. [24] proposed a communication architecture design method based on deep reinforcement learning. Specifically, considering the characteristics of the communication architecture design itself, the attention mechanism and the dynamic embedding mechanism were introduced. Compared with the meta-heuristic algorithm, this method performed well in terms of generalization ability and can achieve the best compromise between solution speed and solution quality. In Ref. [25], a dynamic two-layer learning framework based on DRL was proposed to realize dynamic resource allocation while recognizing the autonomy of system components, and showed the excellent performance compared with the baseline on different SoSs key parameter sets. As the design grows in scale, it remains a formidable task to explore the design area efficiently.

2.2. Spatio-temporal crowdsourcing

Spatio-temporal crowdsourcing (STC) is widely active in different fields such as business operations, data processing, and computer programs. Bhatti et al. [26] systematically investigated and summarized the existing crowdsourcing models. The author briefly defined the concept of traditional crowdsourcing and proposed a basic framework consisting of three components in combination with emerging technologies. Based on this framework, the steps of task design, task setting and incentive mechanism were given, and the verification methods, reward strategies and reputation management under different technologies were discussed. The STC can be essentially understood as determining the

matching relationship between tasks and workers, and balancing execution efficiency and allocation cost to the greatest extent [27]. Therefore, how to improve the task allocation mechanism will become the key core of the design of STC-SoSs. Numerous scholars have carried out extensive theoretical research. Aiming at the most cutting-edge task allocation mechanism in the STC, Gong et al. [28] classified and explained the operation principle according to various design standards, and discussed their respective advantages and disadvantages. In Ref. [29], the author studied the obstacles classified according to the types of participants in STC, including resource, privacy, benefit and other obstacles, which will help to understand the challenges that the design crowdsourcing process should face, and then established a more complete framework. Zhang et al. [30] proposed a two-stage task allocation algorithm based on STC features to maximize the training quality and matching degree for the environment-driven task allocation of heterogeneous crowdsourcing.

Recent advancements in task allocation research have predominantly centered around dynamic online crowdsourcing scenarios. The article [31] proposed a spatial task allocation based on data-driven prediction, which allocates tasks to workers by considering the current and future situation that dynamically enter the STC-SoSs. Next, a greedy algorithm to efficiently assign tasks and a graph partition based decomposition algorithm were designed to find the global optimal. Zhang et al. [32] put forward a prediction-oriented cross-regional online task allocation algorithm, which is accelerated through multiple assignment rounds and optimized by combining offline guidance with online allocation strategy. To encourage crowd workers to complete crowd tasks across regions, an incentive strategy was designed to encourage crowd workers' movement. At the same time, in order to solve the time sensitivity problem of online allocation, Zhang et al. [33] also developed a two-stage task allocation algorithm under the online model, which can achieve the most suitable matching combination while satisfying the allocation efficiency. Jiang et al. [34] solved the photographing to make money problem by establishing a quasi-group role assignment model, which can effectively divide tasks to accelerate the solution, and improved the task completion rate to a certain extent. Moreover, it provided an agent satisfaction evaluation method to quantify the relationship between task completion rate and employee satisfaction. From the above literatures, we can find that most of the STC-SoSs design still only stays in the binary matching between the published tasks and the execution workers. The intricate problems related to the STC necessitate the consideration of the matching of multiple types of objects, including tasks, workers, and workplaces. For example, inspired by several emerging STC applications, a new dynamic matching problem was proposed in Ref. [7], which is called trichromatic online matching problem in real-time spatial crowdsourcing, and the formal definition was given. However, the existing crowdsourcing modeling is often too one-sided and not systematic. Therefore, on the basis of previous research, this paper innovatively organizes and proposes a ternary dynamic matching model for the design of STC-SoSs.

2.3. Deep reinforcement learning in STC

Deep reinforcement learning (DRL), a subfield of machine learning (ML), aims to optimize the expected cumulative return through exploratory learning. The agent, based on the current state, takes actions and iteratively updates the policy network to achieve comprehensive intelligent training. Due to the excellent efficiency and scalability of end-to-end training, as well as the characteristics of setting reward optimization without training samples, DRL is migrated to the design of STC-SoSs. Specifically, its advantage lies in the ability to trade-off indicators such as utility and cost by designing incentive mechanisms, and the model can swiftly generate effective crowdsourcing matching schemes, which can be extended to complex STC-SoSs scenarios. Moreover, such algorithms have also been successfully applied in path planning, group decision-making, cluster confrontation and other fields.

Some pioneering studies based on DRL provide new opportunities and intelligent methods for addressing dynamic challenges in the design of STC-SoSs. Wang et al. [35] introduced a dynamic bipartite graph matching that better reflects real-world applications, and designed an adaptive batch restricted Q-learning based on a constant competitive ratio solution framework. In Ref. [36], an end-to-end DRL framework for task scheduling was proposed, which used a deep Q network to estimate task 's long-term return to cope with dynamic changes. Liu et al. in Ref. [37] proposed a novel DRL method for curiosity-driven energy efficient personnel scheduling to maximize data collection and minimize overall worker energy consumption. Zhao et al. [38] constructed a multi-agent DRL framework, which applied the Advantage Actor-Critic to the multi-agent category and considered both local and global rewards in the setting of the reward function. To further improve the performance of the model, the attention mechanism was introduced to promote the information interaction between agents. To protect the large amount of privacy data generated by crowdsourcing services, Lin et al. [39] proposed a new mechanism based on DRL and blockchain in the STC-SoSs design. Ren et al. [40] proposed a reinforcement learning-based participants selection policy using Q-learning to select the appropriate participants without knowing the environment of the sensing model. These studies collectively demonstrate the versatility and potential of DRL in tackling various aspects of STC-SoSs design.

In addition, some researchers have combined DRL with other mature methods, further improving model accuracy and shortening training time. Considering the dynamic of task allocation scenarios, Quan and Wang [41] proposed a fusion method based on knowledge graph and reinforcement learning framework to optimize task allocation in a changing crowdsourcing environment. Liu et al. [42] proposed a distributed multi-agent DRL solution, using convolutional neural network to extract useful spatial features as input to the Actor-Critic network to generate real-time actions. This solution combined distributed priority experience replay for better exploration and utilization, and fully excavated the spatio-temporal characteristics of the scenarios considered to better match workers and tasks. Xu and Song [43] integrated graph attention network into DRL to solve the problem of low efficiency and long duration of task allocation in mobile crowdsensing. Piao and Liu [44] developed a new sequence deep model, called ''PPO+LSTM'', which contains a sequence model LSTM and uses proximal policy optimization (PPO) for allocating tasks and planning routes. In Ref. [45], the author proposed a task migration policy based on DRL for STC, which aims to minimize the average cost of task completion under the premise of meeting the task deadline. This policy combined long short-term memory, asynchronous deep Q-learning and ant colony optimization to realize the mobile prediction of crowdsourcing units and generate the optimal task migration scheme. These above studies showcase the potential of combining DRL with other sophisticated techniques, leading to more precise, efficient, and contextually adaptive solutions in dynamic crowdsourcing and task allocation scenarios.

3. The design of STC-SoSs based on IAM-PPO algorithm

This section first formally defines a ternary dynamic matching (TDM). Following this, we proceed to characterize the STC allocation as a Markov decision process, specifying the state space, action space, and reward mechanism required for the implementation of the DRL algorithm. Lastly, an invalid action masking (IAM) [46,47] method is introduced to update the policy-based network, and an improved PPO based on IAM (IAM-IPPO) is presented for the STC-SoSs design.

3.1. Problem definition

In this study, we first provide a comprehensive explanation of the fundamental concepts of task, worker, and workplace in the STC. Subsequently, we present a formal definition of the dynamic model and the

utility value associated with a single match. Additionally, we establish the triple matching relationship among tasks, workers, and workplaces, outline the necessary constraints, and provide a specific definition of the TDM problem.

Definition 1. 【**Crowd Task**】Crowd tasks are published by different task demanders. The task set in the STC can be written as $TS = \{t_1, t_2, \cdots, t_n\}$, and each crowd task t_i within this set is expressed as follows:

$$t_i = \langle tl_i, tc_i, td_i, tr_i, tb_i, tw_i \rangle \tag{1}$$

where, n denotes the total number of crowd tasks. tl_i is the geographical location of task t_i. tc_i is a category number of task t_i, which can belong to either maintenance task or haircut task. td_i is the difficulty of each crowd task, and its value is between 0 and 1. tr_i is the radius of the limited publishing range of task t_i, whose center is tl_i. tb_i is the budget of task t_i. tw_i is the time window for different task t_i. **Definition 2.** 【**Crowd Worker**】Different workers have the ability to complete specified tasks. The worker set in the STC can defined as $WS = \{w_1, w_2, \cdots, w_m\}$, and each crowd worker w_j is formalized as follows:

$$w_j = \langle wl_j, wc_j, we_{ij}, ws_{ij}, wv_j, wd_j \rangle \tag{2}$$

where, m denotes the total number of crowd workers. wl_j is geographical location of worker w_j. wc_j is a category number of worker w_j, which can belong to either maintenance worker or haircut worker. we_{ij} is the efficiency of worker w_j for different task t_i. ws_{ij} is the salary of worker w_j for different task t_i, and wage information is strictly confidential as privacy in actual daily life. wv_j is moving velocity of worker w_j. wd_j is the maximal moving distance of worker w_j, and crowd workers move dynamically in the STC platform at speed wv_j and upload current location wl_j to the platform. The workers have the autonomy to join and exit the platform independently, and only take tasks that fall within a radius of maximum movement distance wd_j. **Definition 3.** 【**Crowd Workplace**】As the execution place of workers and task publishers, the workplace set in the STC is defined as $PS = \{p_1, p_2, \cdots, p_s\}$, and each crowd workplace p_k is signified as follows:

$$p_k = \langle pl_k, pc_k, pa_k \rangle \tag{3}$$

where, s denotes the total number of crowd workplaces. pl_k is geographical location of workplace p_k. pc_k is a category of workplace p_k, which can belong to either maintenance workplace or haircut workplace, including large, medium and small workplaces. pa_k is the maximum limit of tasks that can be accommodated simultaneously in the workplace p_k. **Definition 4.** 【**Dynamic Model**】A ternary dynamic model is adopted, which mainly refers to the three types of objects of tasks, workers and workplaces that can dynamically appear or remove. Among them, the crowd tasks have a time window attribute and can only be matched within this period of time. Crowd workers move with a certain speed and random direction on the STC platform, and can join and leave independently. Whenever a crowd task is completed or mismatched, new task and worker are automatically added to the platform at random positions. When a crowd task is successfully matched, it will enter the execution phase. During this period, an accommodation space in the crowd workplace will be occupied. Only when this task is completed, the occupied space will be restored to idle. After each crowd task is published, the platform will proactively match it with worker and workplace. Once the task, worker and workplace are matched, they cannot be changed until the matching task is completed.

Definition 5. 【**Match Utility**】The Match utility refers to the positive and negative benefits obtained by a single match, which mainly consists of three parts: the work utility of workers completing tasks, the cost utility of workers moving to the task area and workplace, and the distance utility between the task publisher and the workplace.

Specifically, the work utility U_1 of crowd workers to complete tasks is defined as the product of task 's difficulty and worker 's efficiency in Eq. (4). The Euclidean distance is used to calculate the total distance d_{ijk} of workers moving to the task and workplace, and the cost utility U_2 of crowd worker is characterized according to the obtained d_{ijk}, as shown in Eq. (5). Similarly, the distance utility U_3 between the task publisher and the workplace can be expressed by Eq. (6). It is worth noting that both cost utility and distance utility are considered as negative values.

$$U_1(td, we) = td_i \times we_{ij} \tag{4}$$

$$\begin{aligned} U_2(d_{ijk}) &= -e^{d_{ijk}} \\ &= -exp\left(\sqrt{(tx_i - wx_j)^2 + (ty_i - wy_j)^2} + \sqrt{(wx_j - px_k)^2 + (wy_j - py_k)^2}\right) \end{aligned} \tag{5}$$

$$U_3(d_{ik}) = -e^{d_{ik}} = -exp\left(\sqrt{(tx_i - px_k)^2 + (ty_i - py_k)^2}\right) \tag{6}$$

where, tx_i and ty_i represent the position coordinates of task t_i; wx_j and wy_j represent the position coordinates of worker w_j; px_k and py_k represent the position coordinates of workplace p_k.

Definition 6. 【**Match Relation**】The task-worker-workplace matching set can be obtained in the design of STC-SoSs, which is denoted by $M = \{(task, worker, workplace) \mid task \in TS, worker \in WS, workplace \in PS\}$. It contains all possible matching relationships among tasks, workers, and workplaces. The task-worker-workplace matching relationship has two stages in its interpretation: Initially, the relationship is defined between tasks and workers. Here, the task is the entity requiring a match, while the worker is the entity being matched to the task. This matching can either be one-to-one (a single worker for each task) or one-to-many (multiple workers for a single task). Subsequently, the task and the worker as a whole becomes the matching party, and the workplace is the party being matched. In this scenario, when the sum of the distances from the workplace to both the worker and the task is smaller, the matching utility is higher.

Definition 7. 【**Satisfying Constraints**】In the STC-SoSs, constraints pertain to the guidelines that must be adhered to by the matching of task-worker-workplace, which include the following constraints:

Task constraint: A crowd task can be contracted by multiple workers. When the cumulative efficiency of multiple workers on a crowd task should not be less than the difficulty of this task. This determines the necessary number of workers for a crowd task.

$$\prod_{j=1}^{N} we_{ij} \geq td_i, \quad i = 1, 2, \cdots, n \tag{7}$$

where, N represents the number of workers required to contract each crowd task t_i.

Worker constraint: A crowd worker can only contract one task at the same time.

$$\sum_{i=1}^{n} x_{ij} \leq 1, \quad j = 1, 2, \cdots, m; \; x_{ij} \in \{0, 1\} \tag{8}$$

where, x_{ij} is the element of the allocation matrix X, when $x_{ij} = 1$, the task t_i can be contracted by the worker w_j; when $x_{ij} = 0$, the task t_i cannot be contracted by the worker w_j.

Workplace constraint: In the dynamic matching, any crowd workplace p_k cannot exceed the maximum limit pa_k of crowd tasks that can be accommodated simultaneously.

$$pt_k \leq pa_k, \quad k = 1, 2, \cdots, s \tag{9}$$

where, pt_k represents the number of tasks that have been assigned to the

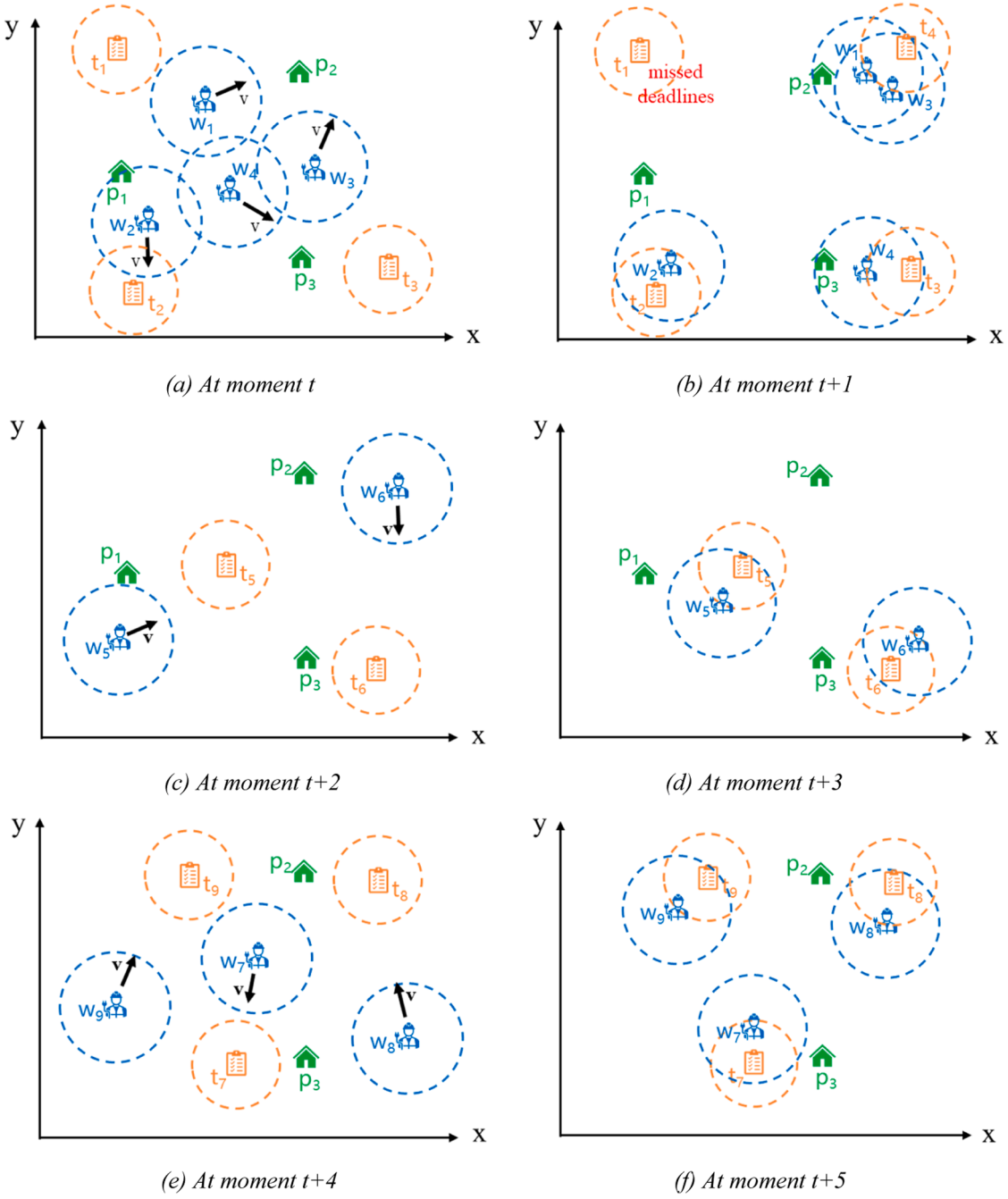


Fig. 2. The TDM problem is illustrated by an example.

workplace p_k

Skill constraint: Different crowd workers can only accomplish specific tasks, and the workers' skills should match those needed to complete the tasks. That is, the category of crowd task t_i should be the same as that of crowd workers w_j.

$$\mathscr{M}(tc_i) = \mathscr{M}(wc_j)) \tag{10}$$

where, $\mathscr{M}$ indicates the category of task or worker, such as the haircut task matches the haircut worker, and the maintenance task matches the maintenance worker.

Budget constraint: The total salary requirement ws_{ij} paid to workers for completing a crowd task t_i cannot exceed the task budget tb_i.

$$\sum_{j=1}^{N} ws_{ij} \le tb_i, \quad i = 1, 2, \cdots, n \tag{11}$$

Spatial constraint: The contracted crowd task t_i should be within the maximum moving distance wd_j of the worker w_j, and the assigned worker w_j should also be within the limited publishing range tr_i of the task t_i, that is, the spatial distance between the task t_i and the worker w_j needs to be less than the task 's limited publishing range tr_i and less than the worker 's maximum moving distance wd_j.

$$d_{ij} = \sqrt{(tx_i - wx_k)^2 + (ty_i - wy_k)^2} \le min(tr_i, wd_j), \quad i = 1, 2, \cdots, n;\ j = 1, 2, \cdots, m \tag{12}$$

where, d_{ij} represents the spatial distance between the crowd task t_i and

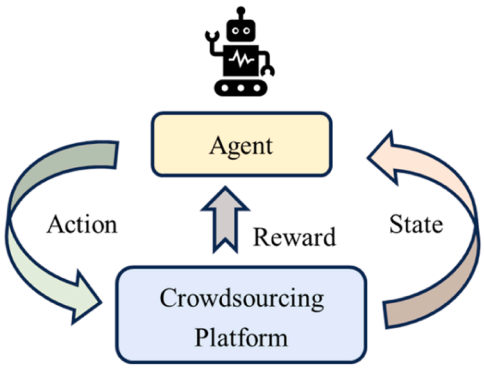


Fig. 3. The interaction between the agent and the crowdsourcing platform.

crowd worker w_j. tx_i and ty_i represent the task 's position coordinates; wx_j and wy_j represent the worker 's position coordinates.

Time window constraint: The task t_i must be matched within the designated time window tw_i, otherwise, this task is regarded as leaving the platform.

$$tw_i^l \le tw_i \le tw_i^u, \quad i = 1, 2, \cdots, n \tag{13}$$

where, tw_i^l and tw_i^u represent the lower and upper bounds of the time window of the task t_i, respectively.

Invariable constraint: Once a triplet $(t,\ w,\ p)$ of task t, worker w and workplace p is matched, it cannot be changed.

Definition 8. 【**TDM Problem**】Given a series of tasks *TS*, workers *WS*, workplaces *PS*, and a utility function $U(.,\ .,\ .)$ on the STC platform, which has no object initially and allows each object to arrive and leave at any time. The main core of the TDM problem is to find a matching relationship M among the tasks, workers and workplaces, so that the total utility under various constraints is maximized *MaxSum* $U(M)$.

We use a simple example to illustrate the TDM problem involved in this paper. At moment t, there are 3 tasks t_1−t_3, 4 workers w_1−w_4 and 3 workplaces p_1−p_3 in a STC platform. Each object has its own attributes, and its corresponding position are shown in Fig. 2(a). Each task has a limited publishing range, represented by an orange dashed circle in Fig. 2(a), and workers within this range are sought for co-contracting. Similarly, each worker also has a maximum moving range, represented by a blue dashed circle in Fig. 2(a), which can match the crowd tasks within this range. The dynamic of the TDM problem is mainly reflected as follows: ① The task has a certain time window, so the task will be removed if the deadline is exceeded. ② The worker move freely in 2D space at a certain speed, and can enter or leave the platform at random. ③ After the task matches the worker, it enters the working stage, which will occupy a space in the workplace that is closest to the average distance between the task and the worker, and this space will be restored after the task is completed. ④ After completing a working cycle, all matched objects are removed from the platform, unmatched objects are retained, and new tasks, workers, and workplaces will be added. Therefore, a new task t_4 is added to the platform at moment t+1 in Fig. 2 (b), and the task t_1 will not be matched due to missed deadlines, while the task t_2−t_4 is contracted by the worker w_1−w_4, forming a series of matching group of 't_2-w_2-p_1', 't_3-w_4-p_3', 't_4-w_1-p_2' and 't_4-w_3-p_2'. At moment t+2, 2 workers w_5−w_6 and 2 tasks t_5−t_6 are added to the platform in Fig. 2(c). At moment t+3, Fig. 2(d) realizes the matching between tasks t_5−t_6 and workers w_5−w_6, forming a series of matching group of 't_5-w_5-p_1' and 't_6-w_6-p_3'. At moment t+4, the platform adds 3 workers w_7−w_9 and 3 tasks t_7−t_9, and the number of workplaces was reduced from 3 to 2 because the space of workplace p_1 is occupied in Fig. 2(e). At moment t+5, Fig. 2(f) realizes the matching between workers w_7−w_9 and tasks t_7−t_9, forming a series of matching group of 't_7-w_7-p_3', 't_8-w_8-p_2' and 't_9-w_9-p_2'. This dynamic STC scenario will be the object of this paper and the case to verify the next proposed algorithm.

3.2. MDP formulation of STC

In this section, the TDM problem for STC-SoSs design is represented as a Markov decision process (MDP) [48]. By feeding the agent to be trained the present state of the crowdsourcing scenario, the matching pair of task-worker-workplace is output. Subsequently, the decision results are fed back to the crowdsourcing platform and interact continuously to achieve the purpose of updating the policy network.

3.2.1. Markov decision process

Given a sequence *Seq* involving the entire STC process, it contains five model elements such as state, action, policy, reward, and return, which can be expressed as $MDP = (S, A, P, R, G)$.

- $S = \{s_0, s_1, s_2, \cdots, s_\tau\}$ represents a state set, including the state of tasks, workers, and workplaces at each moment in the entire sequence *Seq*. (Note: The state space of this paper is discrete.)
- $A = \{a_0, a_1, a_2, \cdots, a_\tau\}$ represents an action set, including each match relation M of tasks, workers and workplaces. (Note: The action space in this paper is also discrete.)
- $P = \pi(a|s)$ represents the policy of executing action a in state s, which can be reflected by conditional probability distribution $p(a|s)$. (Note: In the context of DRL in this paper, it is a random policy.)
- $R = \{r_1, r_2, \cdots, r_\tau\}$ represents the reward feedback of the environment to the agent after the performs action a_t. It is a scalar function $r(s_t, a_t, s_{t+1})$ about the current moment state s_t, action a_t and the next moment state s_{t+1}, which is related to the previously defined match utility. The reward design in the MDP often depends on the specific STC problem.
- $G = \sum_{t=0}^{\tau-1} \gamma^t r_{t+1}$ represents the cumulative return of reward over time steps. After introducing the concept of trajectory, the return is also the sum of all rewards on the trajectory. (Note: Related to the match utility U previously defined in this paper) Among them, γ is the discount factor. The larger the value, the more attention is paid to the past experience; the smaller the value, the more attention is paid to the immediate interests.

In accordance with the Markov property definition, the matching result of the STC platform at the current moment is solely dependent on the state of standby workers, tasks and workplaces and the action set of matching at the previous moment, and is independent of the state and action conditions at other moments. As a result, the interaction depicted in Fig. 3 between the agent and the crowdsourcing platform can be regarded as an MDP.

On the basis of the above modeling elements, the MDP is organized as follows: the agent perceives the initial environment s_0, implements the action a_0 according to the policy π_0. This action, in turn, influences the environment, leading to a transition to a new state s_1, while also providing feedback in the form of a reward, denoted as $r_1(s_0, a_0, s_1)$. Subsequently, the agent adapts its policy, transitioning to π_1, grounded in the new state s_1, and continues to engage in ongoing interactions with the environment.

The graphical representation of the STC design process, underpinned by the Markov decision process (MDP), is illustrated in Fig. 4. This diagram comprises state nodes and action nodes [8]. The edge from state to action is defined by policy, and the edge from action to state is defined by environmental. Notably, with the exception of the initial state, each state returns a reward.

On this basis, a trajectory of the MDP can be defined as follows, that is, the environment evolves from the initial state s_0 to the set of all ac-

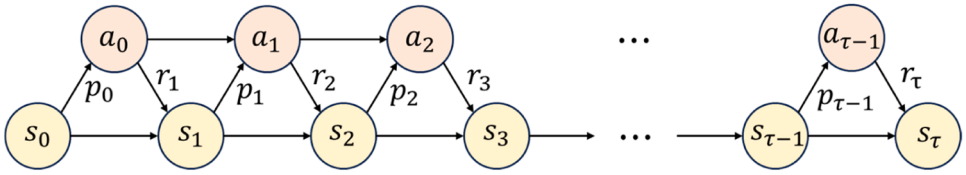


Fig. 4. The graphical model of STC design process based on MDP.

tions, states and rewards of the current state s_τ according to the given policy $\pi(a|s)$.

$$\tau = s_0,\ a_0,\ s_1,\ a_1,\ r_1, \cdots, s_{\tau-1},\ a_{\tau-1},\ r_{\tau-1}, s_\tau,\ a_\tau,\ r_\tau \tag{14}$$

Due to the randomness of policy and state transition, the trajectories obtained in each experiment are stochastic, resulting in different total return and total utility. The ultimate purpose of DRL is to find a set of policy parameters θ, that is, the NN 's weights, so that the mathematical expectation of the total return can be maximized.

$$U = \mathbb{E}_{\pi_\theta}[G(\tau)] = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\tau-1} \gamma^t r_{t+1}\right] \tag{15}$$

3.2.2. State, action, and reward

The state, action and reward are critical to solve the design problem of STC-SoSs by using DRL algorithm. In this section, the detailed introduction lays the foundation for the next writing.

State represents the current state of each element in the STC platform, including the task set $TS^{'}$ published by the demander, the worker set $WS^{'}$ to be allocated and the unoccupied workplace set $PS^{'}$, which is represented by the triples $\langle TS^{'}, WS^{'}, PS^{'}\rangle$. For the elements contained in $TS^{'}$, $WS^{'}$, and $PS^{'}$, see Section 3.1.

Action represents all possible matching combinations of task-worker-workplace scheme throughout the design process of the STC-SoSs. After performing an action, the tasks, workers and workplaces that have been successfully matched will not be considered in the next state. This decision is only related to the state of the previous moment, which is in line with the MDP model.

Reward represents the alteration of matching utility after an action is conducted. With the goal of maximizing match utility, the reward function r shown in Eqs. (16)–(18) is derived from Eqs. (4)–(6). Rewards can be categorized into positive and negative rewards. Positive reward refers to the benefits obtained after the implementation of actions in reinforcement learning, such as the sum of the work reward r_1 of the tasks completed by the workers. Conversely, negative rewards involve penalties incurred following the execution of actions, such as the sum of the cost reward r_2 of the area where the worker needs to move to the task and the workplace, and the sum of the distance reward r_3 between the task demander and the workplace.

$$r_1 = \sum_{i,j \in M} td_i \times we_{ij} \tag{16}$$

$$r_2 = -\sum_{i,j,k \in M} exp\left(\sqrt{\left(tx_i - wx_j\right)^2 + \left(ty_i - wy_j\right)^2} + \sqrt{\left(wx_j - px_k\right)^2 + \left(wy_j - py_k\right)^2}\right) \tag{17}$$

$$r_3 = -\sum_{i,k \in M} exp\left(\sqrt{\left(tx_i - px_k\right)^2 + \left(ty_i - py_k\right)^2}\right) \tag{18}$$

The DRL algorithm relies on the reward-penalty balance to control the exploration of the output scheme. When there are too many positive rewards, the policy tends to match more tasks to obtain more rewards, which may lead to the situation of falling into an unreasonable scheme. On the contrary, if the negative rewards are too much, it will result in the tendency of the agent to not match the task, thus avoiding punishment. In general, the total reward set for each episode should not be too large, and the positive reward is approximately twice as much as the negative reward. To achieve this, we introduce the reward adjustment factor, which meets the requirements of the above reward setting.

$$r = \omega_1 r_1 + \omega_2 r_2 + \omega_3 r_3 \tag{19}$$

where, ω_1, ω_2 and ω_3 are the adjustment factors of the three-part reward, which can be adjusted according to the design requirements.

3.3. Improved PPO algorithm based on an IAM

This section delves into an improved proximal policy optimization algorithm based on invalid action masking (IAM-IPPO) for the design of STC-SoSs. To begin with, a useful method known as IAM is introduced address the issue that, in large-scale situations, the algorithm could have trouble converging. Subsequently, the policy network structure and the update process are further elaborated upon following the introduction of IAM. Lastly, a systematic outline of the implementation of IAM-IPPO is presented, accompanied by the corresponding pseudo-code.

3.3.1. An invalid action masking

When using DRL to solve the design of STC-SoSs, due to the various types, the large number and the varying difficulty of tasks, the limited number and different efficiency of workers, and the different scale of the workplaces, etc., some actions cannot be selected in specific states at a certain time. For example, the action space is set as a matching pair of task, worker and workplace, and the number is *tasks* × *workers* × *workplaces* in TDM problem. As the dynamic matching progresses to subsequent states, certain workplaces may have already reached their capacity limits in accommodating tasks, rendering them unavailable for subsequent allocations. Furthermore, each crowd task is initially assigned a difficulty value. As tasks are allocated to workers, their difficulty values will decrease. Once a task 's difficulty value falls below zero, further worker assignments to that specific task are halted to prevent unnecessary wastage of human resources.

To solve the above problems, soft constraints (i.e., setting a large penalty) are used in the past to make the agent no longer choose actions with negative effects. However, this method has its limitations. Irrespective of the penalty magnitude, it cannot guarantee that the agent will completely avoid such actions, because the agent might still opt to undertake an invalid action in a certain step of episode, accepting the penalty as a calculated risk. As a result, the DRL algorithm will get a poor overall benefit reward. In contrast, in this paper, soft constraints are replaced to hard constraints to mask invalid action, so that it is mandatory to sample only from the effective action set, effectively eliminating the possibility of selecting invalid actions. Huang et al. [46] has proved that when the invalid action space is large, the invalid action masking exhibits superior scalability, and the usual method of giving negative rewards will fail.

The above IAM is a common strategy to avoid repeated generation of invalid actions in a large discrete action space, which is achieved by setting the sampling probability of the corresponding invalid action to approximately 0. In the design of STC-SoSs, the following actions are considered to be invalid and should be masked: ① Continuing to allocate excess workers to a crowd task that has already been contracted by a sufficient number of workers to complete it. ② Assigning a task to a crowd worker who does not have the ability or qualifications to handle

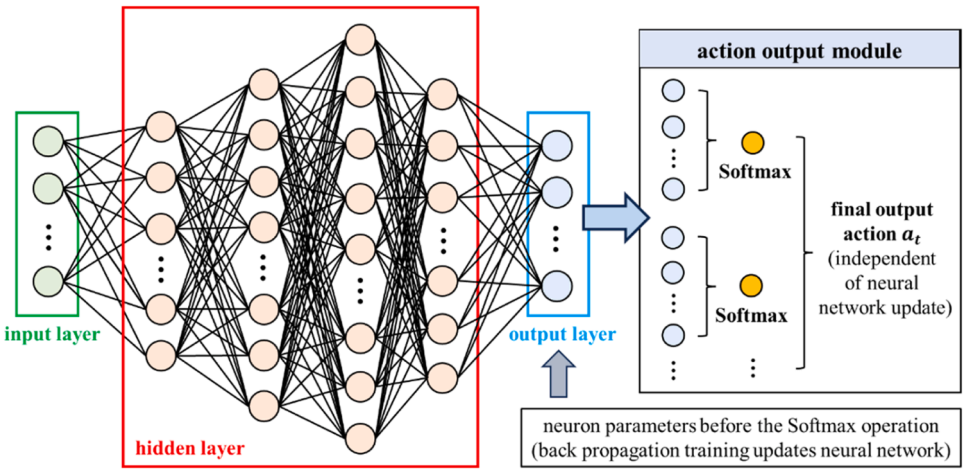


Fig. 5. The structure of policy/value network and action output module under the STC-SoSs.

that particular task. ③ The allocated workplace has already exceeded its maximum capacity limit, it still continues to be allocated. ④ When the number of available crowd tasks, workers, and workplaces is zero, it continues to participate in the subsequent matching. By applying IAM method, the agent is guided to focus on selecting only feasible, logical, and efficient actions, which can lead to better overall performance in the STC-SoSs design.

Since DRL is essentially training the gradient of the policy network, the base PPO used in this paper belongs to the policy gradient algorithm, and its policy is represented by using a neural network (NN). The policy network uses the softmax operation to convert the non-standardized scores (logits) into the action probability distribution, while the MASK operation is generally set on the value function before the action is selected, and the value after masking is returned to the NN training. Next, we will focus on a simple example of which worker should be contracted for a crowd task, and explain this process in detail by referring to the literature [49].

Assuming that a task requester needs to decide which of the four nearby workers to complete this task at a certain moment, the MDP can be understood as taking one of the four actions a_1, a_2, a_3, a_4 under the state s_0, and obtaining the corresponding reward r. Next, the state is transferred to s_1. To simplify the description, the policy π_θ is initialized to select actions with the same probability, that is, $\pi_\theta = [p_1, p_2, p_3, p_4] = [1.0, 1.0, 1.0, 1.0]$, and the output logit is

$$\begin{aligned}&[\pi_\theta(a_1|s_0), \pi_\theta(a_2|s_0), \pi_\theta(a_3|s_0), \pi_\theta(a_4|s_0)] = softmax([p_1, p_2, p_3, p_4])\\ &= [0.25, 0.25, 0.25, 0.25]\end{aligned} \tag{20}$$

$$\pi_\theta(a_i|s_0) = \frac{exp(p_i)}{\sum_j exp(p_j)} \tag{21}$$

Then, the policy gradient algorithm is used to calculate the sampling gradient of $\pi_\theta(\cdot|s_0)$.

$$\begin{aligned}&g = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \sum_{t=0}^{\tau-1} \log\pi_\theta(a_t|s_t) G_t\right] = \nabla_\theta \log\pi_\theta(a_0|s_0) G_0\\ &= [0.75, \ -0.25, -0.25, -0.25]\end{aligned} \tag{22}$$

$$\left(\nabla_\theta \log softmax(\theta)_j\right)_i = \begin{cases} 1 - \dfrac{-exp(l_j)}{\sum_j exp(l_j)} & if\ i = j \\ \dfrac{-exp(l_j)}{\sum_j exp(l_j)} & otherwise \end{cases} \tag{23}$$

Now assume that worker 2 has been assigned, then a_2 is invalid for the state s_0, and the only valid actions are a_1, a_3, a_4, a_5. A large negative number M (e.g., $M = -1 \times 10^8$) is used to replace the probability that the action needs to be masked. Let us use inv_s to represent the masking process and calculate the renormalized probability distribution-$\pi'_\theta(\cdot|s_0)$ as follows:

$$\begin{aligned}&\left[\pi'_\theta(a_1|s_0), \pi'_\theta(a_2|s_0), \pi'_\theta(a_3|s_0), \pi'_\theta(a_4|s_0)\right] = softmax(inv_s([p_1, p_2, p_3, p_4]))\\ &= softmax([p_1, M, p_3, p_4]) = \left[\pi'_\theta(a_1|s_0), \epsilon, \pi'_\theta(a_3|s_0), \pi'_\theta(a_4|s_0)\right]\\ &= [0.33, 0.00, 0.33, 0.33]\end{aligned} \tag{24}$$

where, ϵ is the mask invalid action outcome probability, which ought to be a small number.

If the larger negative value is selected, the probability of selecting the masked invalid action a_2 is actually zero. The following gradient, also known as the invalid policy gradient, will be used to update the policy following the conclusion of the episode:

$$\begin{aligned}&g' = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \sum_{t=0}^{\tau-1} \log\pi'_\theta(a_t|s_t) G_t\right] = \nabla_\theta \log\pi'_\theta(a_0|s_0) G_0\\ &= [0.67, \ 0, -0.33, -0.33]\end{aligned} \tag{25}$$

This example highlights the notion that IAM goes beyond mere renormalization of the probability distribution. In fact, it effectively renders the logit gradient of the corresponding invalid action as zero.

3.3.2. *Policy-based network model*

Most of the RL algorithms in the research are built upon the Actor-Critic framework or its improvements. *Actor* pertains to the policy function primarily employed for action generation and interaction with the environment. *Critic* pertains to the value function responsible for assessing the performance of the Actor and guiding its subsequent action.

The core concept of DRL is to approximate arbitrary function by using the powerful nonlinear fitting ability of deep neural network. For specific practical problems, the training optimization and policy update of neural network in DRL have always been the focus and difficulty. Hence, this paper improves the algorithm from the perspective of network structure and network update, making it more suitable for the design of large-scale STC-SoSs.

3.3.2.1. Structural design. In the structural design of the policy network, considering that the output matching actions of crowd tasks, workers and workplaces are multiple discrete sequences, the standard PPO 's policy network structure is modified to satisfy the action space requirements and prepare for subsequent policy update. Meanwhile, the designed action output module makes the policy network output meet the reasonable requirements of the matching scheme.

We employ a feedforward NN to construct a policy-based network model. According to the content described in Section 3.2.2, the input layer of the NN is the state space composed of the unmatched task set $TS^{'}$, the unallocated worker set $WS^{'}$ and the unoccupied workplace set $PS^{'}$ at the current time, that is, the number of neurons in the input layer is $6TS^{'} + 6WS^{'} + 3PS^{'}$. What 's more, we combine the unallocated worker set and the unoccupied workplace set into a set of alternative schemes (dimension $WS^{'} \times PS^{'}$), which is used as the matching object of crowd tasks. Then the action space can be expressed as $TS^{'} \times WS^{'} \times PS^{'}$. Since each crowdsourcing task can select a match from this set, or may not select anyone, the action space can be expressed as $TS^{'} \times (WS^{'} \times PS^{'} + 1)$, that is, the number of neurons in the output layer.

Suppose a simple STC contains 4 tasks, 6 workers, and 2 workplaces at a certain time. The hidden layer network structure of the standard PPO algorithm is often constructed as a two-layer fully connected layer with 64 neurons, which is obviously difficult to achieve better learning ability. Therefore, we can modify the structure of the policy network to four hidden layers, and the number of neurons is 512, 1024, 2048 and 1024, respectively. The input and output of the policy network are calculated to be $4 \times 6+6 \times 6+3 \times 2=66$ and $4 \times (6 \times 2+1)=52$. The number of hidden layers of the Critic value network is consistent with that of the policy network, but the number of neurons in each layer is different, which is 256, 128, 128 and 64, respectively. The input and output dimensions of the value network are 70 and 1.

Besides, compared with the value network, the policy network adds an action output module after the output layer. For the action output module, the Softmax function is applied to group the neurons and finally output the required action. Compared with the traditional policy network that can only output a single action, the modified policy network can output multiple integer action sequences. It mainly collects experience by continuously interacting with the environment through action sampling, and uses the neuron parameters before the Softmax operation in the action output module to learn multiple distribution results, so as to achieve the purpose of updating the policy network. Fig. 5 illustrates the aforesaid process 's structure.

3.3.2.2. Policy update. The policy update of DRL is a learning process in which the agent gradually has human-like intelligence by accumulating experience over and over again in training. The updating of policy network is the key to the knowledge learning of the agent. However, due to the large action space in the early stage of training, the lack of information in the agent experience pool leads to the slow learning efficiency of the NN, requiring a long training period when tackling the large-scale SoSs design issue.

In the design of STC-SoSs, with the continuous dynamic matching of crowd tasks, workers and workplaces, the invalid action space will become larger and larger, which will lead to the increase of the probability of agent 's error, making it difficult for agent to obtain complete successful experience. To solve this problem, the following rules can be formulated to avoid agent mistakes through empirical analysis:

- Ø **Rule 1:** When the difficulty of a crowd task is less than zero, because it is contracted by multiple workers, no additional workers are assigned to this task.
- Ø **Rule 2:** When a crowd worker does not have the ability to perform a task, such worker is not assigned to this task.
- Ø **Rule 3:** When a workplace exceeds its maximum capacity limit, it will not be allocated further.
- Ø **Rule 4:** When the number of crowd tasks, workers, and workplaces is 0, no such crowd elements will appear in the subsequent matching process.

Based on the above rules, the IAM operation can constrain the output by masking specific neurons, which is divided into the following modifications:

(1) **Modification of policy network.** First, the invalid actions inv_a_t at each moment in the STC-SoSs are analyzed, and then input into the policy network with the state s_t at the same time. Finally, the policy network $\pi_\theta(a_t|s_t)$ is modified to $\pi_\theta(a_t|s_t, inv\_a_t)$.

(2) **Storage of memory library.** The invalid action inv_a_t at the moment t and inv_a_{t+1} at the next moment are stored in the memory library for the update training of the policy network.

(3) **Replacement of neuron parameters.** In the policy network, the corresponding neuron parameters are replaced with maximum negative values by inputting an invalid action inv_a_t, so that the selected actions after the Softmax operation are reasonable and the invalid action masking is realized.

The introduction of rules greatly reduces the action space and limits the policy network 's output to guarantee the feasibility of each step of the output action in a complete episode, so that the agent can quickly accumulate successful and experience. Each update is in the optimization of the network, and the reduction of the action space greatly saves the training time that is constantly trial and error in the early stage of training.

In the design of STC-SoSs, the matching action output by the policy network is required to be multiple discrete integer sequences. The dimension of the final output action is equal to the number of targets, and the generation of each discrete sequence is selected by batch through the Softmax operation. It is evident that the purpose of policy network update is to learn multiple distributions to output multiple integer sequences of the final action. Therefore, the improved policy network $\pi_\theta(a_t|s_t)$ can be expressed as the product of each batch 's policy function, that is:

$$\pi_\theta(a_t|s_t) = \prod_{i=1}^{n} \pi_\theta\left(a_t^i|s_t\right) \tag{26}$$

where, a_t is the agent 's action at the current moment t (i.e., crowdsourcing matching scheme); s_t is the agent 's state at the current moment t (i.e., the attributes of crowdsourcing tasks, workers and workplaces); n denotes the number of crowdsourcing tasks; a_{it} denotes the action a in the ith batch at the current time t.

By constraining the ratio of new and old policy and using importance sampling to control the change of the ratio, the standard PPO algorithm achieves stable and effective policy optimization iteration, which greatly avoids the performance collapse caused by step size selection. It can be well applied to large-scale discrete or continuous concerns with high-dimensional state space and action space, and has the advantages of fast output response and convenient parameter adjustment. The objective function can be expressed as Eq. (27).

$$J^{CPI}(\theta) = E_t[r_t(\theta)\widehat{A}_t] = E_t\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta old}(a_t|s_t)}\widehat{A}_t\right] \tag{27}$$

where, E_t is the mean value of the agent 's reward at the current moment t; $\widehat{A}_t$ is the advantage function representing the importance sampling; r_t is the ratio between the current policy and the original policy; θ is the trainable parameter of the policy function; $\pi_\theta(a_s|s_t)$ is the current policy function; $\pi_{\theta old}(a_s|s_t)$ is the original policy function.

The advantage function is expressed by the difference between the Q value and the V value of the agent at the current time t. The advantage function with a length of sequence T can be expressed as

$$\widehat{A}_t = \sum_{t<T} \gamma^t r_t - V(s_t) \tag{28}$$

where, γ is the discount factor; the first item on the right represents the actual total reward obtained in the sequence T; the second term represents the estimated reward value in the value network.

However, the policy mutation still occurs only by the above method.

Ref. [50] proposed the PPO_clip algorithm that artificially sets the ratio of the old and new policy $r_t(\theta)$ within a fixed clip range, which can prevent the performance fluctuation caused by the large gap between the old and new strategies.

In summary, the improved objective function in this paper is shown in Eqs. (29) and (30).

$$J^{CPI}(\theta) = E_t[min(r_t(\theta)\widehat{A}_t, clip(r_t(\theta), 1-\varepsilon, 1+\varepsilon)\widehat{A}_t)] \tag{29}$$

$$r_t(\theta) = \frac{\sum_{i=1}^{n}\pi_\theta(a_t^i|s_t, inv_{at})}{\pi_{\theta old}(a|s, inv_{at})} \tag{30}$$

where, $\pi_\theta(a_{it}|s_t, inv_{at})$ is the current policy after masking; $\pi_{\theta old}(a|s, inv_{at})$ is the original policy after masking; ε is the truncation coefficient, which can usually be set to 0.15 ~ 0.25.

3.3.2.3. Optimizing tricks. The above operation is the most important optimization modules in the proposed IAM-IPPO. Moreover, this paper applies some tricks such as state normalization, reward scaling, orthogonal initialization, and gradient clipping to the NN in another module to ensure the convergence of the policy network output, and further improve the model accuracy and shorten the training time.

(1) State normalization

State normalization is a trick for preprocessing the state space in DRL. Since different state variables may have different dimensions and ranges, direct use of these variables may cause the learning process to become unstable or inefficient. The main implementation is that all the current states are normalized to a normal distribution with a mean of 0 and a variance of 1 in the process of interacting with the environment. In the subsequent training process, whenever a state is obtained, the mean and variance of all states are updated again, and the normalized state is returned as the NN 's input (Algorithm 1).

(2) Reward scaling

Reward scaling [51] is used to adjust the efficiency, stability and convergence performance in the DRL training. For example, if a reinforcement learning algorithm performs poorly on some tasks, it may be necessary to adjust the learning direction and goal by increasing or decreasing the reward. Reward scaling can also be used to balance exploration and exploitation, that is, while pursuing high rewards, you also explore all possible actions and states to find the best policy (Algorithm 2).

(3) Orthogonal initialization

Orthogonal initialization [52] is a neural network initialization method proposed to prevent gradient disappearance and gradient explosion at the beginning of training, which is divided into two steps: ① The weight matrix is initialized by a Gaussian distribution with a mean of 0 and a standard deviation of 1. ② The singular value decomposition of this weight matrix is carried out to obtain two orthogonal matrices, and one of them is taken as the weight matrix of the neural NN (Algorithm 3).

Algorithm 1
State normalization.

1. **Input:** state $\boldsymbol{s_t}$ // Array of all states
2. **Output:** normalized_states $\boldsymbol{s'_t}$ // Normalized array of states
3. for $\boldsymbol{t} = \mathbf{1}, \mathbf{2}, \ldots$ epoch do
4. mean ← mean($\boldsymbol{s_t}$) // Calculate the mean value of the state array
5. std ← std($\boldsymbol{s_t}$) // Calculate the standard deviation of the state array
6. normalized_states ← (states - mean) / std // Normalize the state array
7. $\boldsymbol{s'_t}$ ← ($\boldsymbol{s_t}$ - mean) / std // Normalize the state array
8. $\boldsymbol{s_t} \leftarrow \boldsymbol{s'_t}$
9. end for

Algorithm 2
Reward scaling.

1. **Input:** reward $\boldsymbol{r_t}$, state $\boldsymbol{s_t}$
2. **Output:** scaled_reward $\boldsymbol{r'_t}$
3. $\boldsymbol{R_0} \leftarrow 0$
3. Define $\boldsymbol{\gamma}$ // Discount factor
4. $\boldsymbol{R_0} \leftarrow 0$
6. mean ← mean($\boldsymbol{R_{t-1}}$) // Calculate the mean value
7. std ← std($\boldsymbol{R_{t-1}}$) // Calculate the standard deviation
8. $\boldsymbol{s'_t}$ ← ($\boldsymbol{s_t}$ - mean) / std // Normalize the state array
9. $\boldsymbol{R_t} = \boldsymbol{\gamma R_{t-1}} + \boldsymbol{r_t}$
10. $\boldsymbol{r'_t} \leftarrow \boldsymbol{r_t / std}(\boldsymbol{s'_t})$ // Only divided std
11. $\boldsymbol{r_t} \leftarrow \boldsymbol{r'_t}$
12. end for

Algorithm 3
Orthogonal initialization.

1. **Input:** weights $\boldsymbol{\theta}$ // Initial weight matrix
2. **Output:** initialized_weights $\boldsymbol{\theta'}$ // Weights after orthogonal initialization
3. $\boldsymbol{\theta} \leftarrow$ Normal(mean=0, std=1) // Gain the initial weight matrix using a Gaussian distribution
4. U, s, VT ← svd($\boldsymbol{\theta}$) // Perform singular value decomposition
5. $\boldsymbol{\theta'} \leftarrow$ U // Select U or VT as the weight matrix, here take U as an example

Algorithm 4
Gradient clipping.

1. **Input:** learning rate $\boldsymbol{lr}$, gradient clipping threshold $\boldsymbol{\sigma}$, weights $\boldsymbol{\theta'}$
2. **Output:** updated_weights $\boldsymbol{\theta''}$ // Weights after gradient clipping
3. for $\boldsymbol{epochs} = \mathbf{1}, \mathbf{2}, \ldots$ epoch do
4. gradient = derivative_of_loss($\boldsymbol{\theta'}$) // Calculate the gradient
5. if L2-norm(gradient) > $\boldsymbol{\sigma}$
6. gradient ← $\boldsymbol{\sigma}$ * gradient / L2-norm(gradient)
7. $\boldsymbol{\theta''} \leftarrow \boldsymbol{\theta'} - \boldsymbol{lr}$ * gradient // Update network parameters
8. $\boldsymbol{\theta'} \leftarrow \boldsymbol{\theta''}$
9. end for

(4) Gradient clipping

Gradient clipping [50] is a trick introduced to prevent gradient explosion during training, which can also play a role in stabilizing the training process. Its essence is to clip the gradient during the NN 's back propagation so that this gradient will never reach a certain threshold. Usually, the gradient clipping can be set in the Actor and Critic network (Algorithm 4).

3.3.3. Algorithm implementation of the IAM-IPPO

The IAM-IPPO algorithm proposed in this paper is based on the standard PPO, which not only retains the advantages of the original PPO algorithm, but also improves the algorithm by introducing two optimization modules.

The core pseudo-code of IPPO policy network training is shown in Algorithm 5. Lines 1 to 4 represent the process of environment modeling and parameter preparation at the initial stage of training. The total number of training rounds is indicated in line 5. Lines 6 to 9 represent

Algorithm 5
Pseudo-code of IPPO policy network training.

Input: old_policy $\pi_{\theta old}$
Output: updated_policy π_{θ}
Define state space S, action space A.
Define the hyperparameters of PPO algorithm: learning rate, discount factor, etc.
for $epochs = 1, 2, \ldots$ epoch do
  for $steps = 1, 2, \ldots$ step do
    Input state s_t and invalid action inv_a_t, and obtain action a_t by the policy.
    Perform action a_t and get reward r_t, next state s_{t+1} and invalid action inv_a_{t+1}.
    Add $(s_t, inv\_a_t, a_t, r_t, s_{t+1}, inv\_a_{t+1})$ to the experience pool
    if the experience pool is full
      for $k = 1, 2, \ldots$ do
        Update the policy π by maximizing the PPO-clip target.

$$r_t(\theta) = \frac{\sum_{i=1}^{n} \pi_\theta(a_{it}|s_t, inv\_a_t)}{\pi_{\theta old}(a|s, inv\_a_t)}$$

$$J^{CPI}(\theta) = E_t\left[min\left(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_t\right)\right]$$

        Update θ by Random gradient ascent Adam.

$$\pi_{\theta old}(a|s, inv_{a_t}) = \pi_\theta(a_{it}|s_t, inv_{a_t})$$

      end for
  end for
end for

the process by which agents interact in the environment and store the gained experience. Lines 10 to 16 show the updating training process of the policy network, while line 11 indicates that the policy network collects one experience and updates several times, speeding up the learning efficiency, which is also one of the advantages of PPO compared with other algorithms.

Fig. 6 shows the overall flow chart of the algorithm studied in this paper. At the beginning of the algorithm, prior to providing the MDP elements—such as state space, action space, and reward function—the STC interactive environment required for training is first built. In this stage, we use tricks 1 and 2 to normalize the state and scale the reward respectively. Subsequently, the structural framework of the policy network and the value network are defined as depicted in Fig. 5, Section 3.3.2.1, and the training weights are orthogonally initialized. The input of the policy network is state s_t, and the output is reward r_t and action a_t. The input of the value network is the state s_t and a_t, and the output is the value q_t. Most importantly, when the policy network outputs actions, we introduce the module 1 invalid action masking as the key innovation point of this paper to improve the rationality of the work output. Then, we will perform the action a_t in the interactive environment to transfer the state to s_{t+1}, and store a series of experiences such as s_t, r_t, a_t, and s_{t+1} into the playback pool for subsequent updates by executing the trajectory process shown in Fig. 4. Finally, in order to ensure the training and training of the network, we calculate the loss

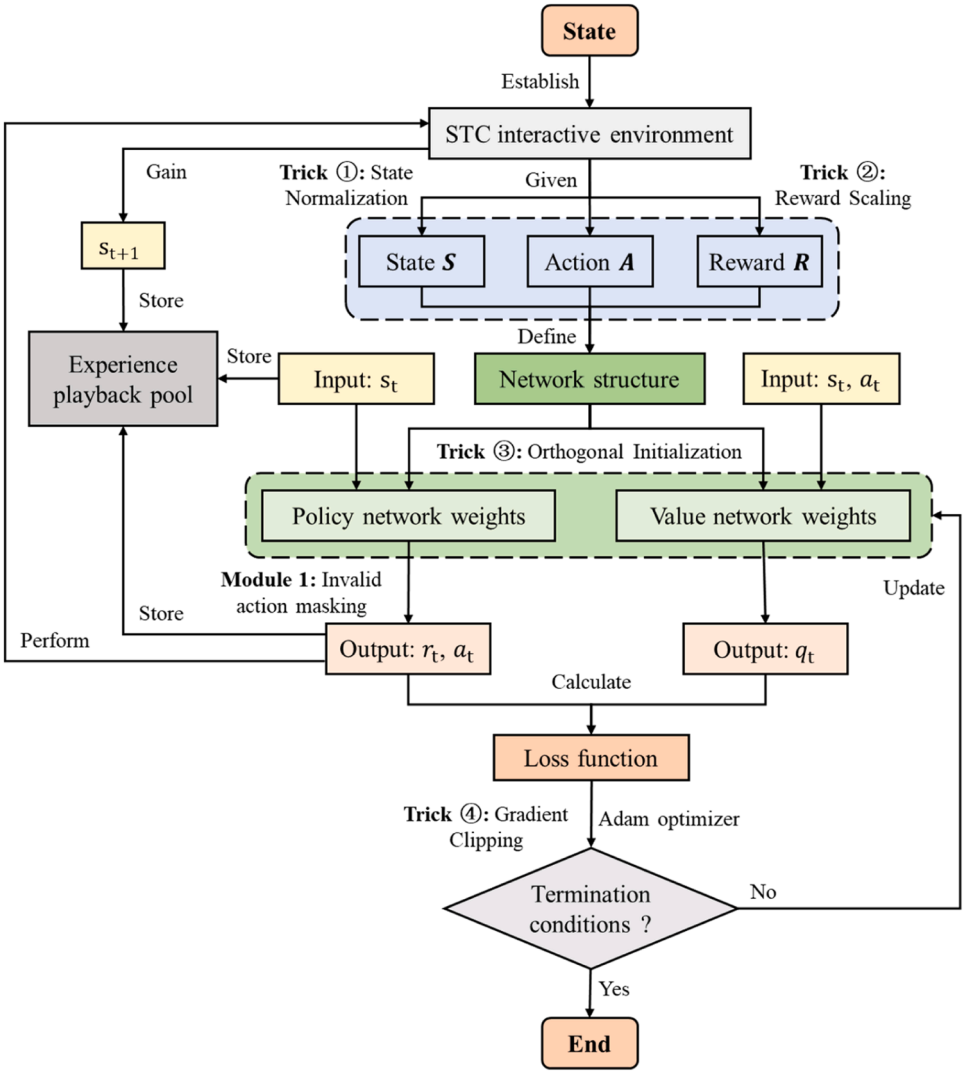


Fig. 6. Flow chart of the algorithm proposed in this paper.

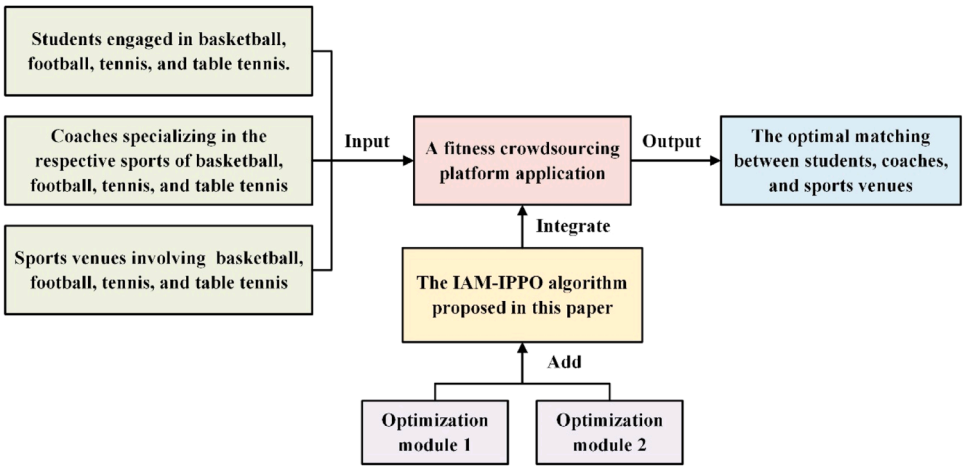


Fig. 7. The application interface diagram of the algorithm in the fitness crowdsourcing.

Table 1
Main parameter setting of four algorithms.

Algorithm	Parameter	Value
Rule-based	Convergence condition	0.05
	Penalty factor	0.1
	Iteration times	100
Heuristic NSGA-II	Population size	100
	Selection operator	Tournament selection
	Crossover operator	Two-point crossover
	Mutation operator	Breeder-GA mutation
	Crossover probability	0.5
	Mutation scaling factor	0.5
	Mutation probability	0.5
	Max generation	200
Value-based DQN	Network structure	66-256-128-128-52
	Learning rate of network	5e-5
	Adam optimizer epsilon parameter	1e-8
	Batch size	64
	Discount factor	0.99
	Frequency of training	50,000
Policy-based IAM-IPPO	Policy network structure	66-512-1024-2045-1024-52
	Value network structure	70-256-128-128-64
	Learning rate of policy network	5e-5
	Learning rate of value network	4.5e-5
	Adam optimizer epsilon parameter	1e-5
	Batch size	64
	Discount factor	0.99
	GAE parameter	0.95
	Truncation coefficient	0.2
	Entropy coefficient	0.01
	Gradient clip parameter	0.5
	Frequency of training	200,000

function and use the Adam optimizer to perform gradient clipping update. When the termination condition is met, the algorithm ends, otherwise the algorithm loop continues.

4. Case study: spatio-temporal crowdsourcing

To validate the feasibility and superiority of the algorithm proposed in this paper through experimental setting and result analysis, this section focuses on a specific case study involving a spatio-temporal crowdsourcing platform for fitness.

Within the realm of spatio-temporal crowdsourcing, the entities responsible for issuing contracts are students engaged in basketball, football, tennis, and table tennis. These students express their requirements for seeking guidance from coaches through a fitness crowdsourcing platform application. On the other hand, the parties who receive these contracts are coaches specializing in the respective sports of basketball, football, tennis, and table tennis. At the same time, it also includes the corresponding basketball, football, tennis, table tennis four kinds of sports venues.

To obtain optimal matching between students, coaches, and sports venues in this particular scenario, our proposed algorithm has been integrated into the fitness crowdsourcing platform application. Due to the time-limited nature of tasks and the constant movement of coaches, the platform encounters a dynamic matching challenge within the constraints of spatial and temporal. Additionally, as the number of platform users grows, the increase of matching time and the decline of matching success will be one of the main technical problems we face. To address these issues, the platform has introduced two optimization modules. Fig. 7 displays the application interface diagram of the algorithm in this scenario.

4.1. Experimental setup

4.1.1. Datasets

Aiming at the ternary dynamic matching problem for STC-SoSs design proposed in this paper, we use a policy-based DRL improved PPO algorithm based on invalid action masking to train the network model. According to the characteristics of DRL, it does not need to mark data, but only needs to define state space, action space and reward function. However, due to the lack of an existing public training environment required for the simulation experiment, we draw on the STC mode of Yelp [53] and gMission [54] platform to construct a verification scenario suitable for evaluating our proposed algorithm, and carry out numerous comparative tests and exploratory experiments.

The gMission and Yelp platforms allocate various crowd tasks from the collected and acquired information, which contain three distinct entities: workers, tasks, and workplaces. By reflecting the actual situation, the worker sets include different information such as quantity, type, location, efficiency, moving speed, maximum moving distance and salary requirement. The task sets include many attributes such as quantity, type, location, budget, publishing range, and time window. The workplace sets include quantity, type, location, maximum limit of tasks that can be accommodated simultaneously, etc. (Note: In this paper, the average score of platform consumers is used as the worker 's efficiency, and the task 's rating reflects different budgets. The salary requirement of workers and the capacity limit of the workplaces are expressed by the average level directly given by the platform.)

4.1.2. Parameter setting

Table 1 shows the initial parameter settings of four algorithms

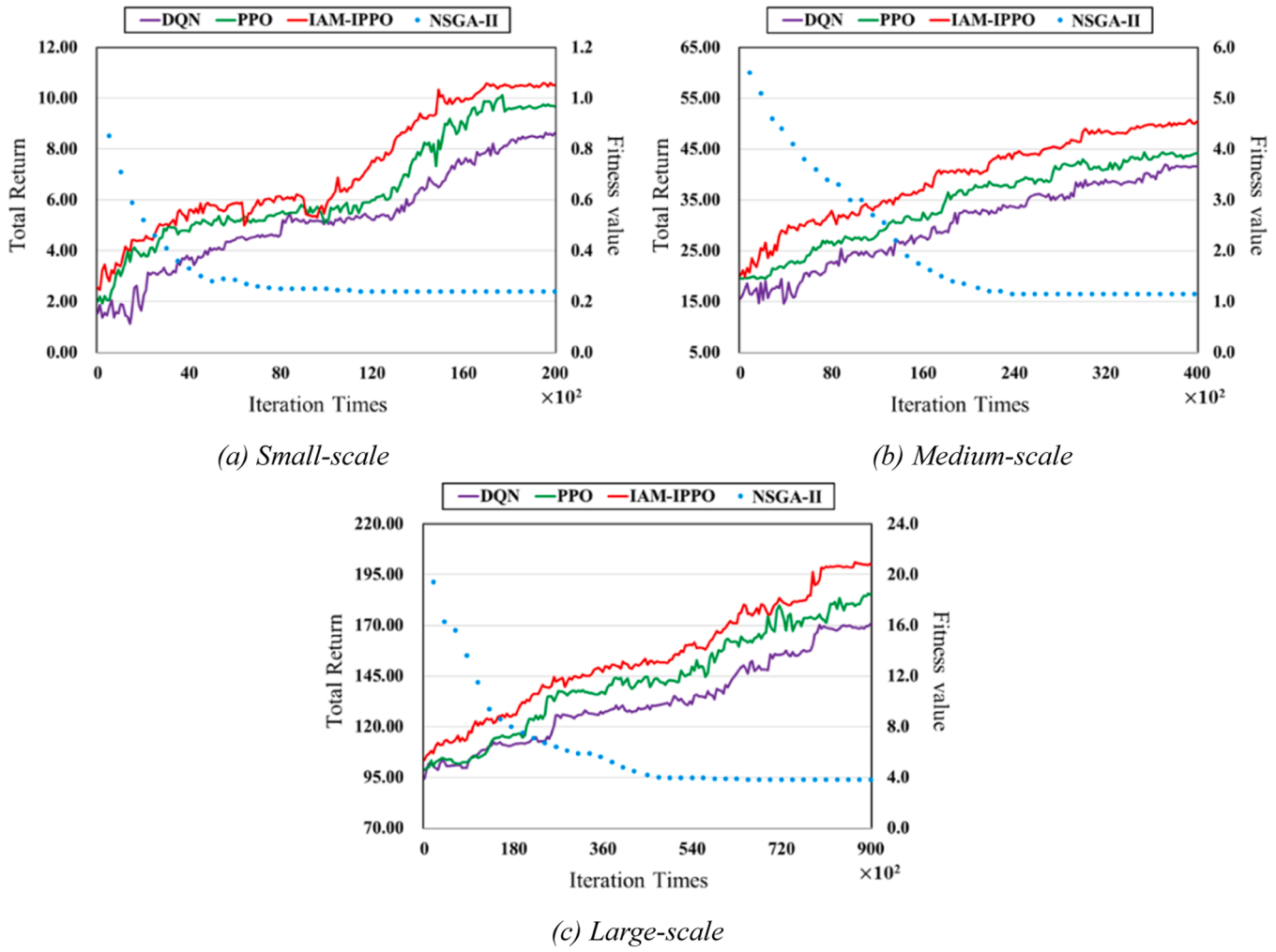


Fig. 8. Convergence performance of NSGA-II, DQN and IAM-IPPO.

including rule-based, heuristic NSGA-II, value-based DQN and policy-based IAM-IPPO. In addition to the IAM-IPPO proposed in this paper, the specific implementation steps of the other three algorithms are shown in the Appendixes A,–C.

4.1.3. Evaluation metrics

We comprehensively compare the performance of the presented IAM-IPPO experimentally by taking into account its convergence performance, allocation accuracy, matching total utility, and running total duration. The experiment is carried out on the Windows 10 operating system configured with i5-12400F @ 2.8GHz CPU, RTX 3060 GPU and 16G memory, and implemented with python 3.8 as programming language.

Convergence performance. Convergence is the most direct manifestation of judging whether different algorithms are feasible. In the comparison algorithm of this paper, the rule-based algorithm has no convergence index due to its characteristics. The NSGA-II algorithm judges whether it converges by observing the fitness function, while the reinforcement learning model gives the convergence of the reward after 50,000 episode training.

Allocation accuracy. Since different tasks in the crowdsourcing allocation process can only be contracted by specific workers and completed in specific workplaces, the allocation accuracy is the premise of the STC-SoSs design to ensure the correctness of the allocation.

Matching total utility. Definition 5 has given the positive and negative utility obtained by a single match in Section 3.1. The matching total utility refers to the sum of work utility, cost utility and distance utility brought by all matching in the whole crowdsourcing process.

$$U = \sum_{M} U_1 + U_2 + U_3 \tag{31}$$

where, U_1 represents the work utility; U_2 represents cost utility; U_3 represents distance utility; M denotes all matching pairs in the whole STC process.

Running total duration. Crowdsourcing tasks, workers, and workplaces are constantly dynamically allocated in a spatio-temporal environment. The running total duration refers to the time required from the start of crowdsourcing behavior to the end of all crowdsourcing tasks assigned throughout the STC-SoSs design process.

In addition, the metrics of crowdsourcing systems include data quality, system cost, task coverage rate and so on, but these are not the evaluation metrics studied in this paper.

4.2. Experimental simulation analysis

To eliminate the contingency in the process of experimental simulation, we construct three kinds of crowdsourcing verification scenarios of small, medium and large scale respectively. Among them, the small-scale scene contains a total of 10 tasks, 10 workers and 4 workplaces; the medium-scale scene contains 50 tasks, 60 workers and 20 workplaces; the large-scale scene consists of 200 tasks, 250 workers and 100 workplaces. Each scale is repeated 30 times and the experimental results

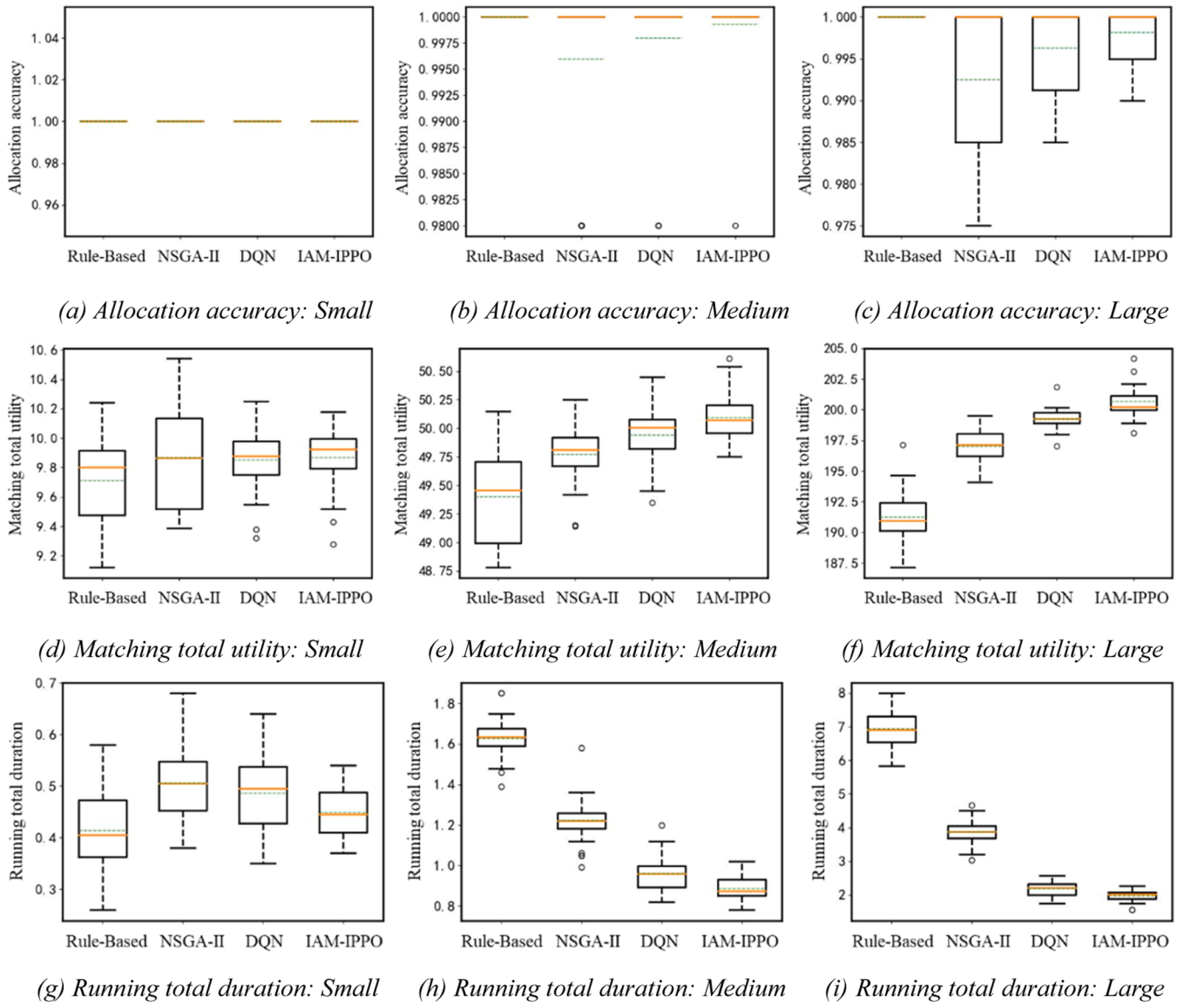


Fig. 9. Contrast experiments of evaluation indexes using various algorithms under different crowdsourcing scales.

are recorded.

4.2.1. Contrast experiment

The contrast experiments evaluate the performance by comparing different algorithms, models, or systems under the same experimental conditions. Usually, contrast experiments select a set of performance evaluation indicators and use statistical methods to analyze the significance of the experimental results. In this paper, we have compared the rule-based algorithm, heuristic search algorithm (NSGA-II [55]), and reinforcement learning algorithm (value-based DQN [36] and policy-based IAM-IPPO).

Convergence performance. As shown in Fig. 8, the convergence performance of different algorithms in small, medium and large-scale crowdsourcing scenarios are illustrated. The left ordinate of each subgraph represents the total return value of the reinforcement learning algorithm, which the right ordinate represents the fitness value of the heuristic search algorithm, and the abscissa represents the number of iterations. Note that heuristic NSGA-II method iterates hundreds of times, while the reinforcement learning iterates tens of thousands of times.

Firstly, let us delve into the NSGA-II algorithm and its distinct convergence curve. It is evident from the convergence curve that the fitness curve of NSGA-II tends to stabilize around the 80th, 200th, and 400th generations in small, medium, and large-scale scenarios, respectively. This can be attributed to the fact that as the complexity of the crowdsourcing scale increases, the algorithm 's exploration space expands, necessitating a greater number of iterations for optimization. Secondly, the other three reinforcement learning algorithms DQN, PPO and IAM-IPPO can also maintain convergence. Among them, the total return of IAM-IPPO algorithm is higher than that of DQN and PPO. Specifically, IAM-IPPO converges to about 2.0 $\times$ 102, PPO converges to about 1.8 $\times$ 102, and DQN converges to about 1.7 $\times$ 102 in Fig. 8(c). Furthermore, it is worth mentioning that the value-based DQN algorithm demonstrates slower convergence speed and greater fluctuation when compared to the policy-based PPO and IAM-IPPO algorithms. This discrepancy primarily arises from the fact that the DQN algorithm necessitates the comparison of values across various behaviors upon obtaining the value function. When the spatial dimension is large, this process is quite time-consuming. The reason why the IAM-IPPO has better convergence than the PPO is also to reduce the spatial dimension through the invalid action masking used in this paper, so as to improve the convergence performance.

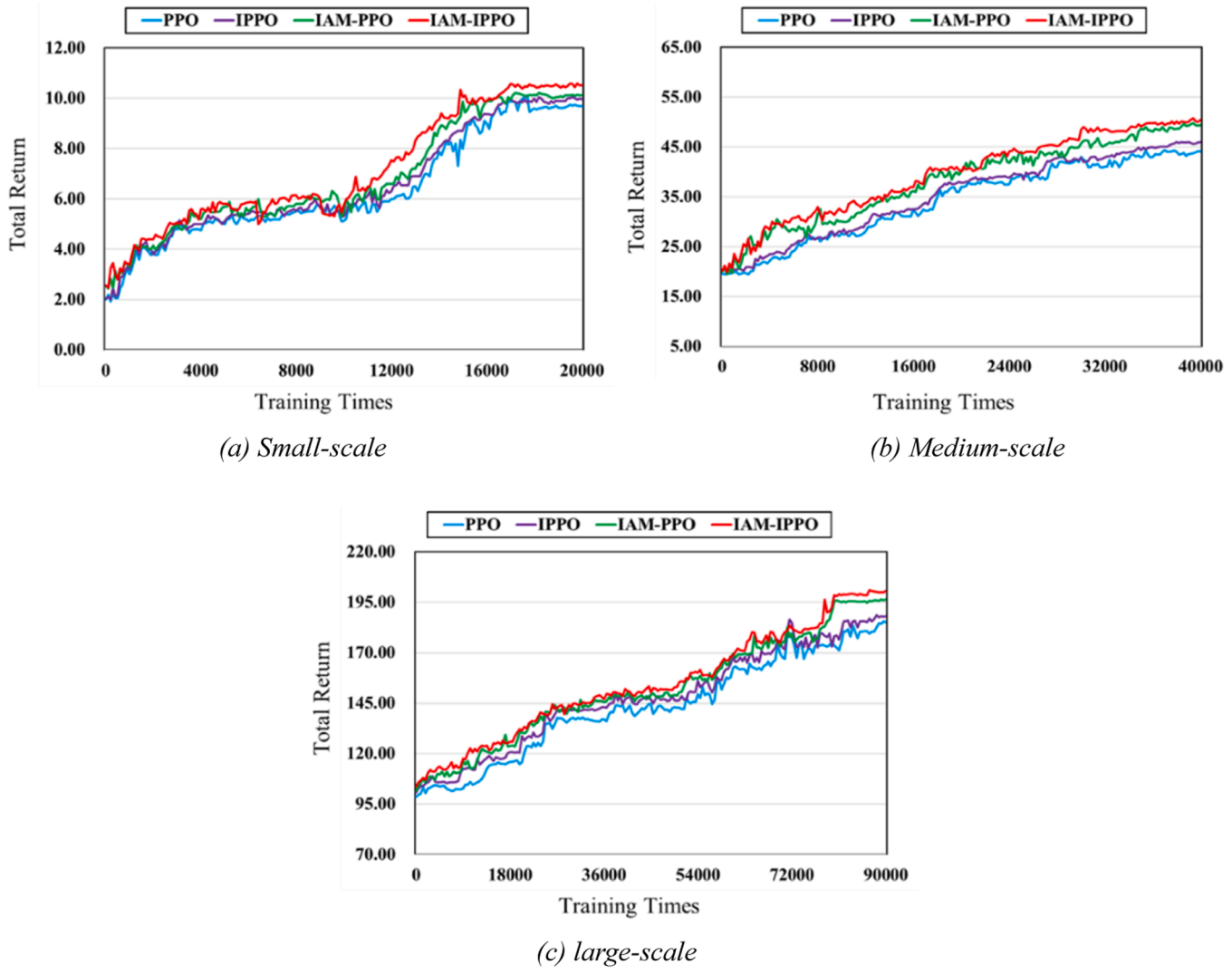


Fig. 10. Convergence performance of PPO, IPPO, IAM-PPO and IAM-IPPO.

Fig. 9 is a set of box plots used to show the comparison of the three indicators under different crowdsourcing scales. The abscissa of each subgraph represents four algorithms, and the ordinate represents the evaluation indicators. In these box plots, we marked the upper bound, 3/4 quantile, 1/2 quantile (yellow solid line), 1/4 quantile, lower bound, mean (green dotted line) and abnormal point (black circle).

Allocation accuracy. Fig. 9(a)–(c), respectively, show the comparative results of the allocation accuracy of different algorithms in small, medium and large scale crowdsourcing scenarios. Among them, only multiple lines are shown in Fig. 9(a), which is because the upper bound, 3/4 quantile, 1/2 quantile, 1/4 quantile, lower bound of the box plot coincide, indicating that the allocation accuracy of the four algorithms reaches 100 % in small-scale scenes. What 's more, we can find that the allocation accuracy of rule-based algorithms is 100 % in small, medium or large scale. In small-scale scenarios, due to the limited number of crowdsourcing elements, the algorithm can often effectively guarantee the allocation accuracy. However, this statement does not hold true in more complicated scenes - though there are exceptions. We can find that the allocation accuracy of the rule-based algorithm is 100 % on small scale, medium scale and large scale. This is mainly because we have set the rules specifying which tasks should be assigned to which workers. Hence, the algorithm only needs to consider which task belongs to which category. In Fig. 9(b), the solid line above NSGA-II, DQN and IAM-IPPO algorithms indicates that more than half of the 30 simulation experiments have achieved 100 % allocation accuracy, while the green dotted line below indicates the average allocation accuracy of different algorithms, namely NSGA-II < DQN < IAM-IPPO. By further observing Fig. 9(c), we can clearly see that the proposed IAM-IPPO algorithm has obvious advantages over DQN and IAM-IPPO algorithms in 30 simulation experiments. In the analysis of convergence performance, we have mentioned that the allocation accuracy of both NSGA-II and DQN decreases significantly with the increase of crowdsourcing elements. The reason why the IAM-IPPO algorithm has higher matching accuracy is that the masking operation is introduced into the invalid action.

Matching total utility. Fig. 9(d)–(f), respectively, show the comparative results of the matching total utility of different algorithms in small, medium and large scale crowdsourcing scenarios. It is evident that the IAM-IPPO algorithm presented in this paper performs well in crowdsourcing scenarios of any scale, with the total utility ranking as follows: Rule-Based < NSGA-II < DQN < IAM-IPPO. Moreover, the vertical height of the box plot involving the rule-based and NSGA-II algorithm is higher than that of the other two DRL methods, indicating that the DQN and IAM-IPPO have more concentrated results in 30 simulation experiments, which also reflects the stability of these algorithms. In general, matching total utility is related to the allocation accuracy. The higher the allocation accuracy, the higher the matching total utility, but this is not absolute. For example, the rule-based allocation accuracy is very high, but often the result is not the optimal solution.

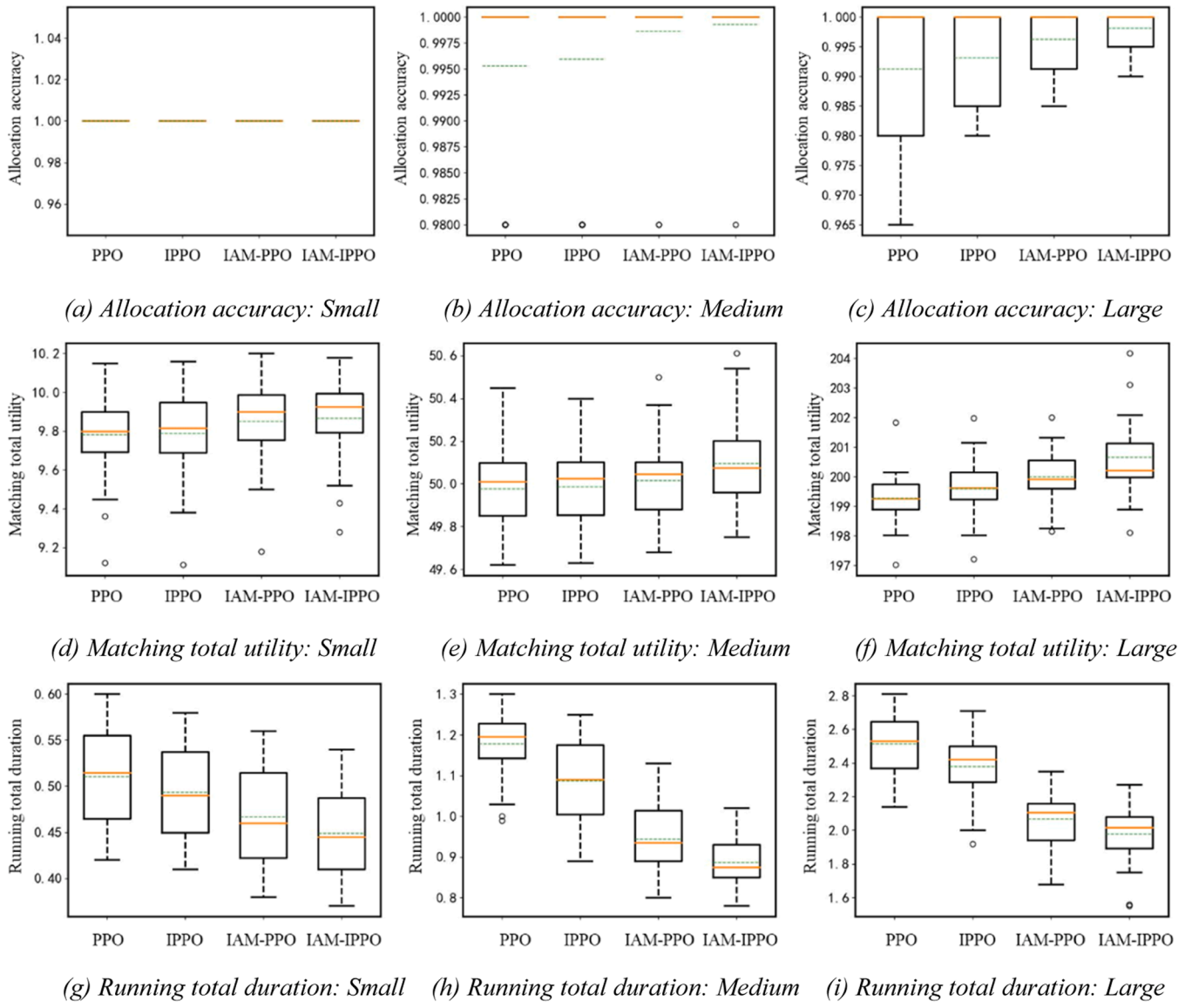


Fig. 11. Ablation experiments of evaluation indexes using various modules under different crowdsourcing scales.

Table 2
Nonparametric test *p*-value statistics of algorithms using different modules.

Evaluating Indicators	Algorithm	Crowdsourcing scale		
		Small	Medium	Large
Allocation accuracy	PPO	7.95×10^{-2}	4.99×10^{-2}	8.52×10^{-3}
	IPPO	8.15×10^{-2}	6.27×10^{-2}	1.62×10^{-2}
	IAM-PPO	8.25×10^{-2}	7.45×10^{-2}	4.64×10^{-2}
	IAM-IPPO	\	\	\
Matching total utility	PPO	6.21×10^{-2}	3.33×10^{-2}	6.97×10^{-3}
	IPPO	6.33×10^{-2}	4.85×10^{-2}	9.45×10^{-3}
	IAM-PPO	6.45×10^{-2}	5.94×10^{-2}	3.15×10^{-2}
	IAM-IPPO	\	\	\
Running total duration	PPO	5.35×10^{-2}	3.01×10^{-2}	7.17×10^{-3}
	IPPO	5.49×10^{-2}	3.89×10^{-2}	8.64×10^{-3}
	IAM-PPO	5.61×10^{-2}	4.66×10^{-2}	2.11×10^{-2}
	IAM-IPPO	\	\	\

Running total duration. Fig. 9(g)–(i), respectively, show the comparative results of the running total duration of different algorithms in small, medium and large scale crowdsourcing scenarios. Fig. 9(g) illustrates that while the rule-based algorithm responds quickly in small-

Table 3
The analysis of time complexity and the comparison of calculation response using different algorithms.

Algorithm	Time complexity	Average calculating response / s		
		Small scale	Medium scale	Large scale
Rule-based	$O(NMS)$	0.419	1.638	7.775
NSGA-II	$O(N_p N_q^2)$	0.541	1.204	3.980
DQN	$O(N_t N_m)$	0.434	0.872	2.133
PPO	$O(N_t N_m)$	0.453	0.883	2.157
IPPO	$O(N_t N_m)$	0.460	0.906	2.174
IAM-PPO	$O(N_t N_m)$	0.466	0.923	2.180
IAM-IPPO	$O(N_t N_m)$	0.479	0.941	2.205

Note: For rule-based algorithm, N, M, and S represent the number of tasks, workers, and workplaces traversed, respectively. While N_p and N_q are respectively the number of targets and individuals set in NSGA-II. In reinforcement learning algorithms, N_t refers to the number of time steps in each episode, and N_m refers to the number of iterations of the episode.

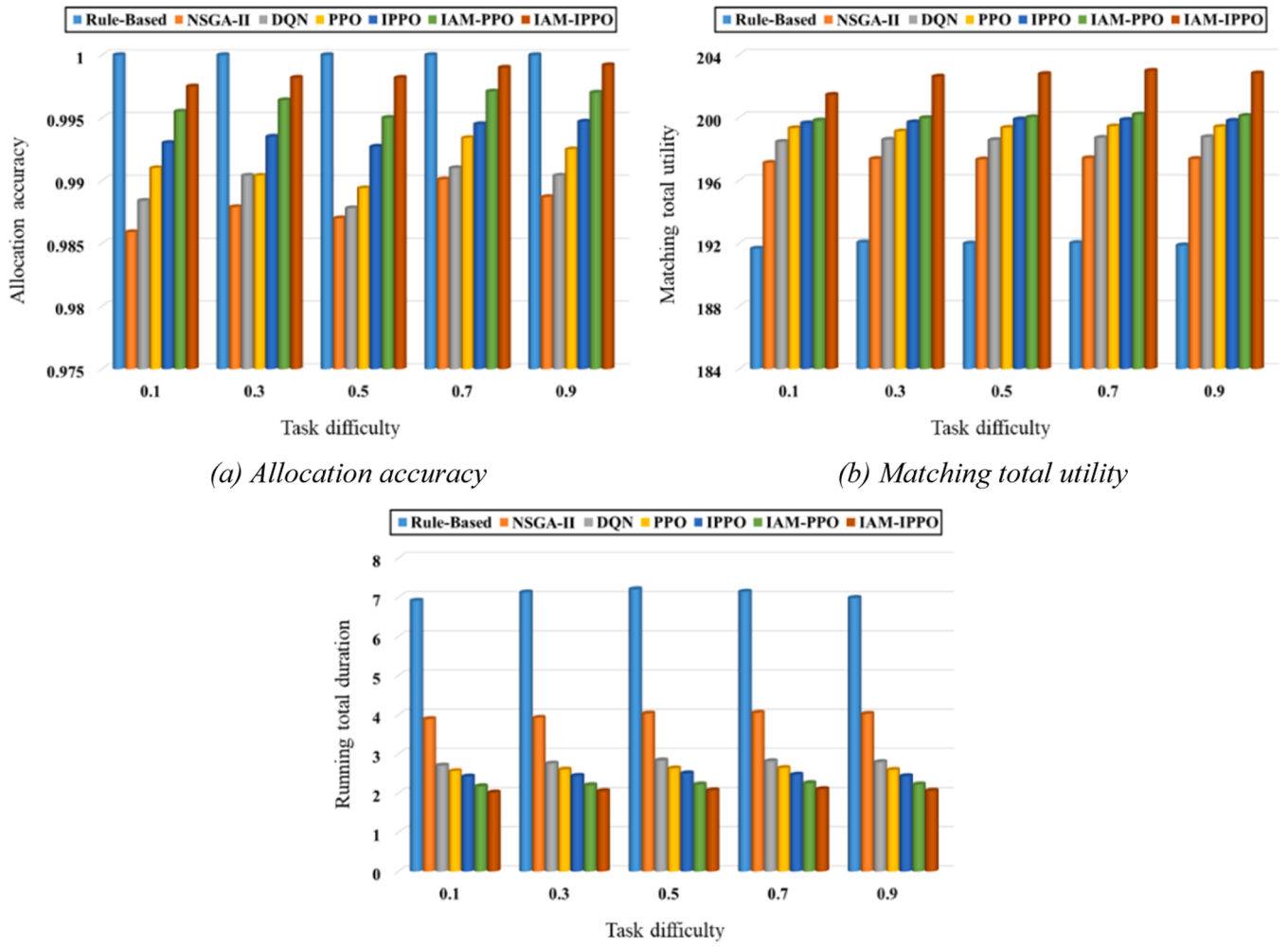


Fig. 12. Influence of task 's difficulty.

scale settings, it gradually loses its advantages and takes much longer to run than other algorithms as crowdsourcing grows in scale. The rule-based algorithm specifies mandatory constraints to ensure the allocation accuracy, but this is at the expense of the running time of the algorithm. Compared with the other three algorithms, since NSGA-II needs to iterate multiple times to obtain enough alternatives, the running total duration is the longest among the three. The policy-based IAM-IPPO algorithm develops a set of action policy and directly optimizes this policy to gain the maximum reward. In contrast, the value-based DQN method does not require an explicit policy formulation. It constructs a value function to select the most valuable action. Therefore, the total running time of IAM-IPPO is less than that of value-based DQN.

4.2.2. Ablation experiment

The ablation experiments [56] evaluate the impact of a component or module on the overall performance by systematically removing or modifying an algorithm, model, or system. In general, ablation experiments are often performed after contrast experiments. Therefore, we further set up ablation experiments to explore the influence of two optimization modules in PPO, IPPO, IAM-PPO and IAM-IPPO.

Convergence performance. Fig. 10 shows the convergence of different optimization modules in small, medium and large-scale crowdsourcing scenarios. From Fig. 10(a)–(c), we can intuitively observe that PPO, IPPO, IAM-PPO and IAM-IPPO gradually converge after 80,000, 35,000 and 16,000 times of training in large, medium and small scales, respectively. Notably, the total return of IAM-IPPO is slightly higher than that of other baseline algorithms regardless of the scale. In large-scale, even though PPO and IPPO still tend to converge as a whole, the error experience may be generated due to invalid actions, which makes the convergence process more volatile than IAM-PPO and IAM-IPPO algorithms. The incorporation of the Gradient Clipping trick in Module 2 renders the gradient more stable during the convergence process for IAM-IPPO.

Similar to Fig. 9, Fig. 11 presents the comparison results of evaluation indicators in small, medium and large-scale crowdsourcing scenarios using different optimization modules. In this evaluation, the standard PPO serves as the control group in the ablation experiment. The IPPO has added the tricks (Module 1) described in Section 3.3.2.3. On the basis of PPO, The IAM-PPO has integrated the invalid action masking (Module 2) introduced in Section 3.3.1. The IAM-IPPO algorithm has the above two optimization modules simultaneously.

Allocation accuracy. In small-scale scenarios, the allocation accuracy of all four methods is 100 %, as seen by Fig. 11(a). This reason has been clearly explained above. In Fig. 11(b), the allocation accuracy of the PPO, IPPO, IAM-PPO and IAM-IPPO is 100 % in more than half of the 30 experiments. The average allocation accuracy (indicated by the green dotted line) is 99.53 %, 99.60 %, 99.87 % and 99.93 %, respectively. As shown in Fig. 11(c), the allocation accuracy from large to small is as follows: IAM-IPPO > IAM-PPO > IPPO > PPO. Moreover, it can be seen that Module 2 has a higher impact on algorithm optimization than

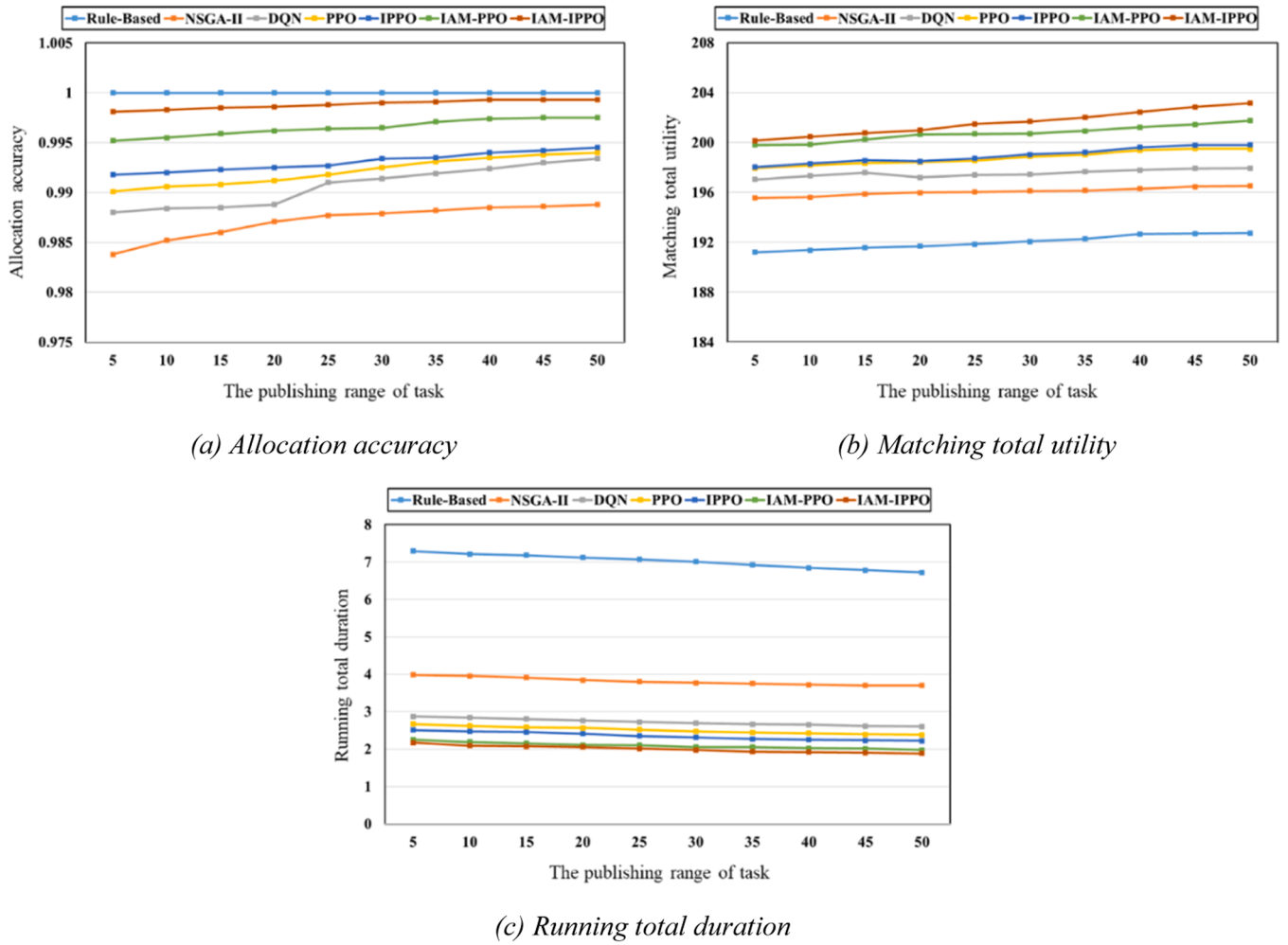


Fig. 13. Influence of the publishing range of task.

Module 1, because Module 2 sets the probability of selecting the wrong action to a smaller value by using hard constraints, thereby improving the allocation accuracy. Module 1 only makes the network training more stable through some tricks, which can improve the allocation accuracy to a small extent.

Matching total utility. As shown in Fig. 11(d)–(f), the matching total utility of IAM-PPO with Module 2 is higher than that of IPPO with Module 1. The order of matching total utility from small to large is PPO, IPPO, IAM-PPO and IAM-IPPO. We can conclude that the introduction of the optimization module has less influence on the total utility of the matching than on the other two evaluation indexes, but it can still prove that the two optimization modules have a positive effect on the improvement of the PPO algorithm. This may be attributed to the factors that impact the allocation accuracy.

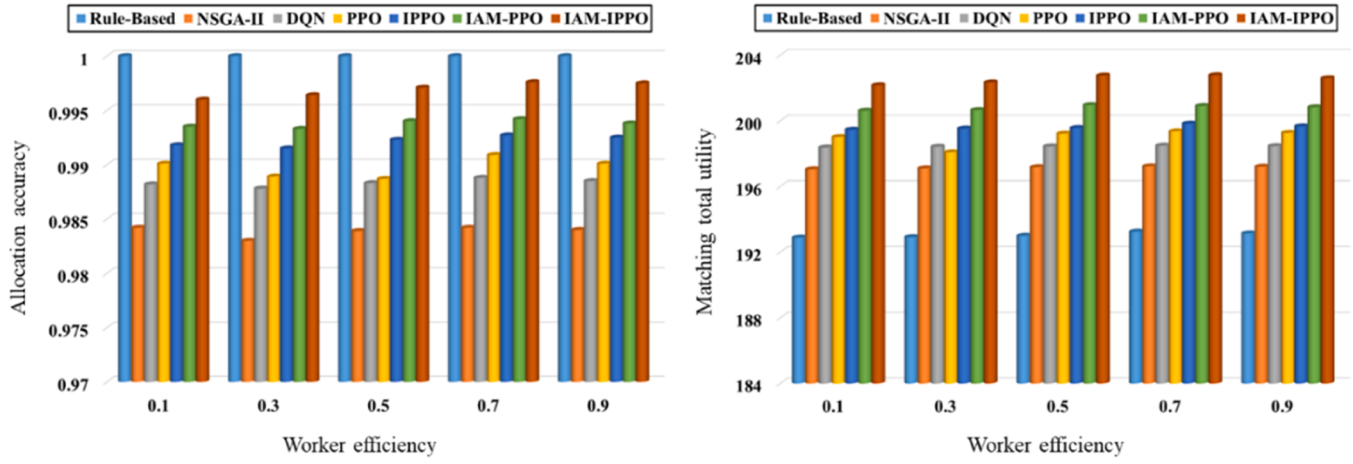
Running total duration. We draw box plots using different modules under different crowdsourcing scales and mark the average running total duration, as shown in Fig. 11(g)–(i). Module 2 simplifies the action space by masking invalid actions, thereby shortening the running time. On the other hand, Module 1 improves computational efficiency through NN 's orthogonal initialization. With the expansion of crowdsourcing scale, the running total duration of these algorithms has also increased. It is evident that, provided the matching scheme is satisfied, the IAM-IPPO algorithm 's running total duration is noticeably less than that of other baselines.

To comprehensively demonstrate the superiority of the proposed approach, we adopt the Wilcoxon signed rank for non-parametric testing and the IAM-IPPO with the greatest performance across all scales as the reference group. The corresponding results have been summarized in Table 2. Based on the above results, the four algorithms perform similarly in small-scale situations, and when the crowdsourcing scale becomes more complex, the proposed IAM-IPPO is significantly superior to other baselines.

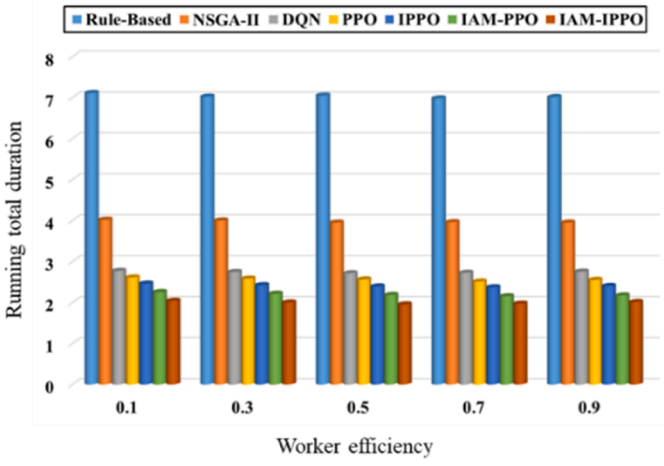
In Table 3, we compare the time complexity of different algorithms, from which we can see that as follows. For rule-based algorithm, the time complexity is affected by the number of tasks, workers and workplaces traversed. The time complexity of NSGA-II mainly comes from the number of targets and individuals. The time complexity of algorithms involving reinforcement learning is the same, depending on the total number of iterations of the episode and the number of time steps per episode. Additionally, we also give the average calculating response of different algorithms at different scales. We can find that in small-scale scenarios, the rule-based algorithm exhibits the shortest response time; however, this advantage diminishes as the scale increases. By comparing all reinforcement learning algorithms, we find that even if the PPO is continuously optimized to obtain IAM-IPPO, the calculating response time is not greatly increased. Combined with the comprehensive analysis of the experiment in Section 4.2, within the range of time acceptable changes, the allocation accuracy and matching total utility have been greatly improved.

4.3. Generalization ability analysis

The work of Figs. 9 and 11 in Section 4.2 has analyzed the evaluation metrics of various models with the quantity of tasks, workers and



(a) Allocation accuracy *(b) Matching total utility*



(c) Running total duration

Fig. 14. Influence of worker 's efficiency.

workplaces change (i.e., small, medium, and large scale). To further explore the impact of other crowdsourcing attributes on evaluation indicators, this section takes large-scale scenarios as an example to change the different attribute values of tasks, workers and workplaces, which proves the robustness of the proposed algorithm.

4.3.1. *Influence of task 's attributes*

Among all the task 's attributes of STC, we focus on selecting the most representative task 's difficulty and publishing range for generalization research. As shown in Fig. 12, the ordinate of each subgraph is the evaluation indicators, and the abscissa divides the task 's difficulty into five levels of 0.1, 0.3, 0.5, 0.7 and 0.9. The comparison results show that our proposed IAM-IPPO has more accurate allocation accuracy, higher matching total utility, and shorter total running duration than other baselines. Specifically, when the task 's difficulty is 0.7 and 0.9, the allocation accuracy of each algorithm is the highest in Fig. 12(a). Similarly, seen from Fig. 12(b) and (c), even if the task 's difficulty changes, the influence of the matching total utility and the total running duration is not as obvious as the allocation accuracy, but we can still see that when the task 's difficulty is equal to 0.7, the performance is the best. By comparing various models when the task 's difficulty is 0.7, it is concluded that the allocation accuracy and matching total utility of IAM-IPPO are improved by 0.56 % and 1.78 %, and the total running duration is reduced by 25.71 %. The reason for this may be that the higher task difficulty requires more workers to match.

What 's more, we also observe the performance changes of all algorithms as the publishing range increases in the form of a line chart. Fig. 13 shows that the publishing range is directly proportional to the allocation accuracy and the matching total utility, while inversely proportional to the running total duration. Further analysis shows that when the publishing range reaches its maximum value of 50, the three indicators of various algorithms perform best. This reason may be that the increase in the publishing range brings the possibility of selecting more workers and workplaces.

4.3.2. *Influence of worker 's attributes*

Similarly, we focus on the worker 's efficiency and maximal moving distance in the worker 's attributes of STC. As shown in Fig. 14, the worker 's efficiency is categorized into five levels of 0.1, 0.3, 0.5, 0.7 and 0.9, and the performance comparison experiments are carried out at different levels. Similar to the task 's difficulty attribute, when the worker efficiency is 0.7, IAM-IPPO is compared with PPO in the design of STC-SoSs. The allocation accuracy and matching total utility of the former are higher by 0.68 % and 1.86 %, and the total running time is shorter by 23.64 %. Consequently, we can infer that higher the efficiency of workers makes it easier to match more tasks effectively.

Fig. 15 shows that as the maximum moving distance of worker gradually increases, the allocation accuracy and matching total utility of IAM-IPPO increase, because the farther the moving distance is, the more tasks are successfully matched. In the case of a certain amount of

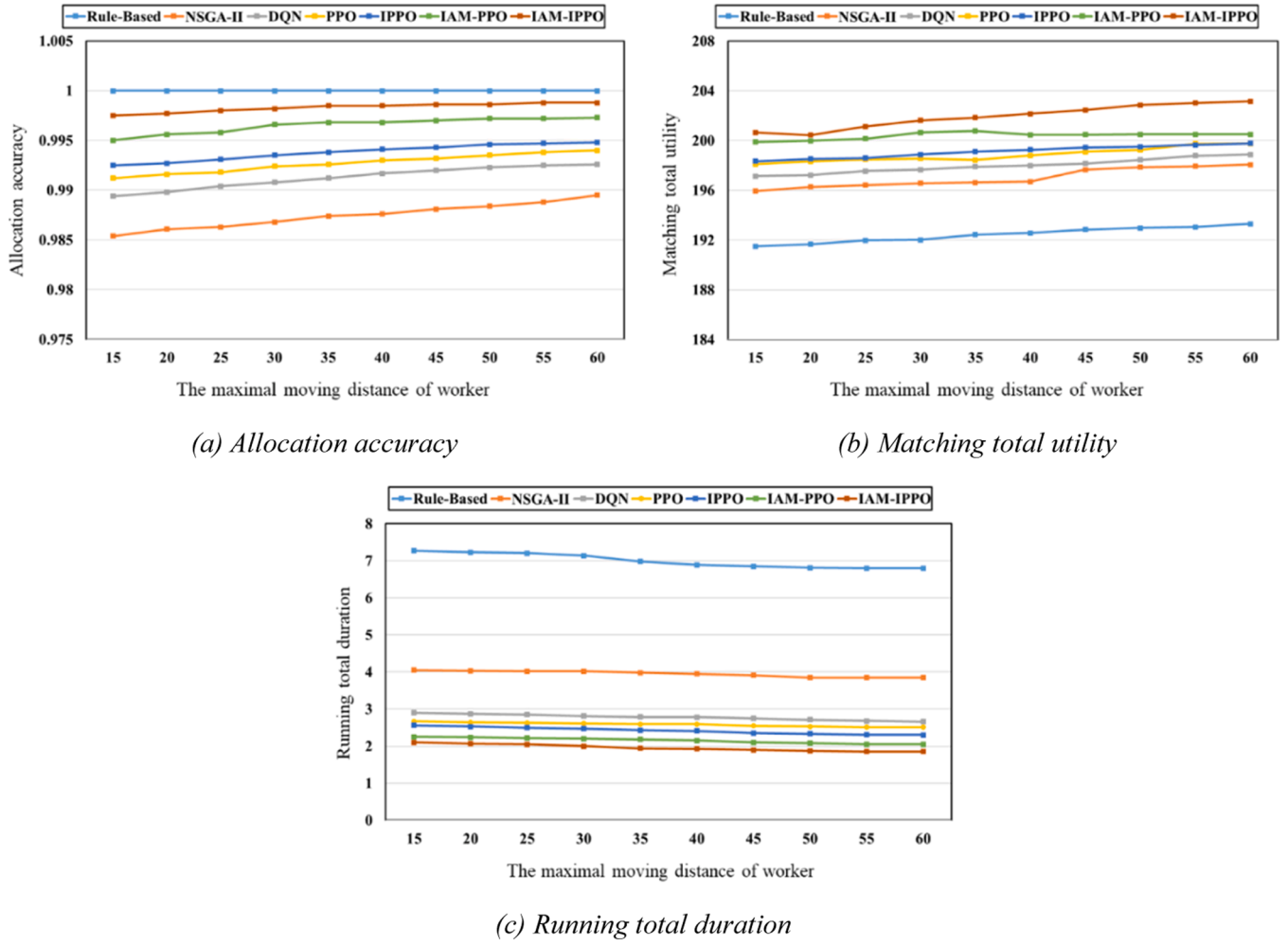


Fig. 15. Influence of the maximal moving distance of worker.

crowdsourcing total elements, the running total duration is reduced. When the worker 's maximum moving distance is 60, the allocation accuracy and matching total utility of all algorithms in the STC are the highest, while the running total duration is the shortest. Similarly, the increase of workers' moving distance will also bring the possibility of selecting more tasks and workplaces.

4.3.3. *Influence of workplace 's attributes*

The maximum limit of tasks that can be accommodated simultaneously in the workplace (i.e., workplace 's capacity) is one of the most important attributes, and we conduct comparative experiments by changing this value.

The ordinates in each subgraph of Fig. 16 shows the allocation accuracy, the matching total utility and the running total duration respectively, and the abscissa represents the workplace 's capacity. With the exception of the rule-based approach, Fig. 16(a) demonstrates that IAM-IPPO 's allocation accuracy is greater than that of the other comparisons. Fig. 16(b) shows that the matching total utility increases with the increase of the workplace 's capacity, because the publisher and executor of the tasks have more choice of workplace. The crowdsourcing platform seeks to produce this result. Compared with IPPO, IAM-IPPO reduces the decision action space due to the introduction of IAM module. The use of tricks such as forward initialization accelerates network updates in comparison to IAM-PPO. The smaller the workplace 's capacity, the more obvious the above results, as shown in Fig. 16(c).

6. Conclusion and future work

In order to fully utilize the concept of synergy in crowdsourcing, this study presents a system-of-systems (SoSs) approach to intelligently design spatio-temporal crowdsourcing (STC). Building upon the standard proximal policy optimization (PPO), this paper proposes an improved proximal policy optimization based on invalid action masking (IAM-IPPO) for the intelligent design of spatio-temporal crowdsourcing system-of-systems (STC-SoSs). For the STC scenario in real-world life, a novel ternary dynamic matching (TDM) model is constructed, encompassing tasks, workers, and workplaces. To address the problem of unreasonable action matching caused by spatio-temporal complexity, an invalid action masking (IAM) method that violently removes such matches through a hard constraint is introduced, which ensures the convergence of the proposed algorithm and the feasibility of the output scheme. Additionally, certain tricks are applied to the Actor-Critic network framework to further improve the model accuracy and shorten the training time. Contrast experiments and ablation experiments demonstrate the superiority of IAM-IPPO over other baselines in generating reasonable solutions that fulfill the design requirements of STC-SoSs, particularly in scenarios involving larger-scale crowdsourcing. In forthcoming research, we intend to investigate the potential of employing multi-agent deep reinforcement learning techniques to address these challenges.

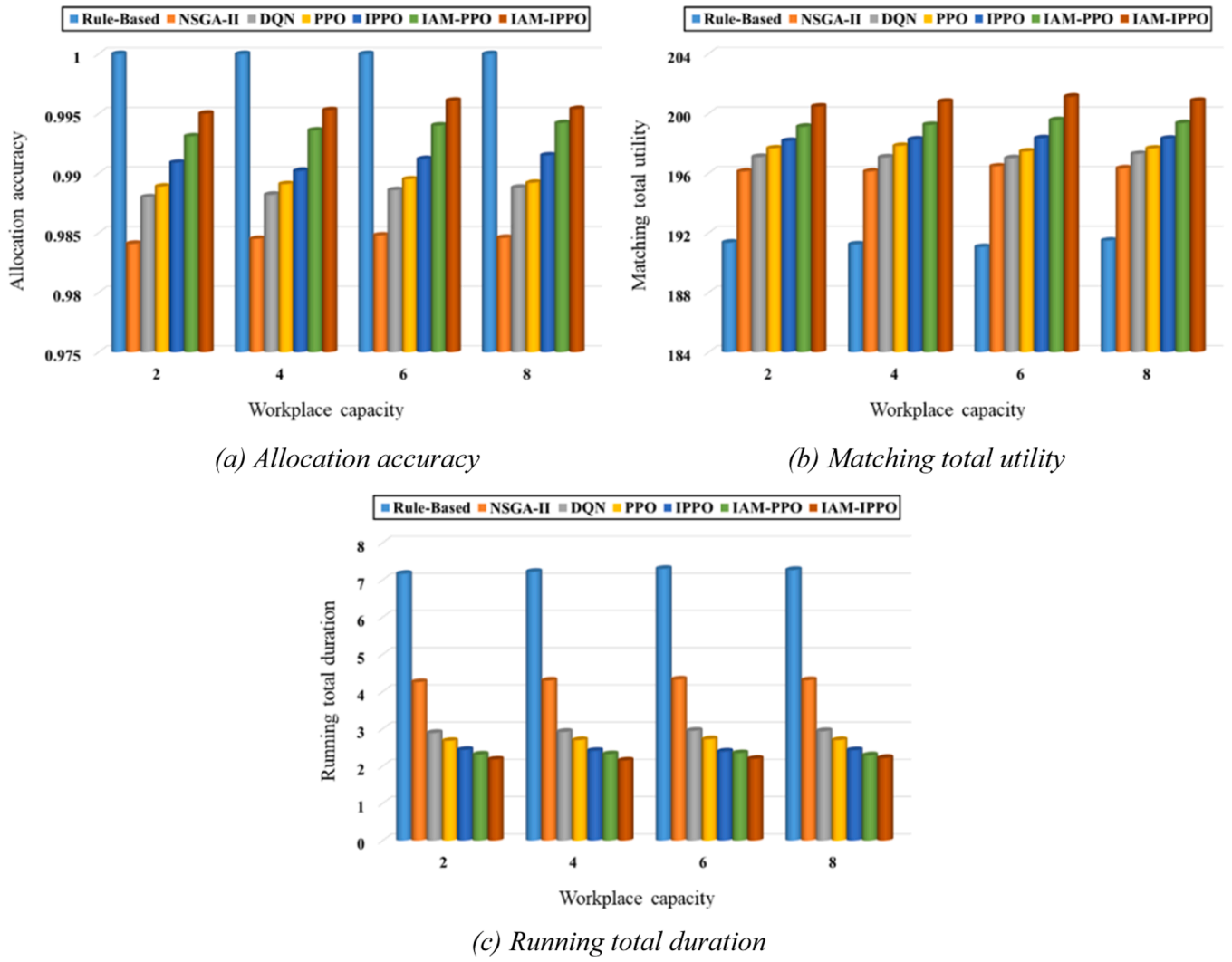


Fig. 16. Influence of workplace 's capacity.

CRediT authorship contribution statement

Wei Ding: Conceptualization, Methodology, Software, Validation, Visualization, Writing – original draft. **Zhenjun Ming:** Conceptualization, Investigation, Funding acquisition, Supervision, Writing – review & editing. **Guoxin Wang:** Project administration, Supervision. **Yan Yan:** Resources, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

Zhenjun Ming acknowledges support from the National Natural Science Foundation of China (62373047), and Beijing Municipal Science & Technology Foundation (3222020). Guoxin Wang acknowledges support from the National Natural Science Foundation of China (51975056).

Appendix A. Rule-based algorithm

The rule-based algorithm realizes the matching of tasks, workers and workplaces by means of cyclic traversal. The specific steps are as follows:

1) According to the difficulty of crowdsourcing tasks, it is preliminarily judged that at least several workers are needed to complete.
2) Within the publishing range of tasks, traverse the workers closest to each crowdsourcing task, and meet the budget less than the worker 's salary requirement.
3) It is determined that there is currently no matchable object for this task if the worker being traversed fails to fulfill the requirements due to the maximum moving distance restriction.
4) Observe whether there is a worker performing multiple tasks at the same time, and if so, assign the exceeded tasks to the worker who is second nearest, and so on, so that all worker constraints are met.
5) The matching combination of tasks and workers is taken as a whole, and the workplace closest to the sum of the two is traversed.
6) Determine whether the assigned workplace exceeds its capacity simultaneously. If it exceeds, the excess part is arranged to the second nearest workplace, and so on, and finally all matching pairs of tasks-workers-workplaces are obtained.

Appendix B. Heuristic NSGA-II algorithm

The heuristic algorithm adopts the typical NSGA-II optimization algorithm, which explores the non-dominated solution set through

continuous iteration to obtain the matching scheme of tasks, workers and workplaces in the STC-SoSs. The following are the precise steps:

1) Considering the three STC elements of tasks, workers and workplaces, a multi-objective optimization model of STC is established, which takes the maximum work utility, the minimum cost utility and the minimum distance utility involved in Definition 5 as the objective function, and the various constraints given in Definition 7 as the constraints.
2) The matching scheme of STC elements is taken as the optimization variable, and the initial population of N is randomly generated. In this paper, the binary coding method is adopted.
3) Using selection, crossover and mutation operations to generate subpopulation Q, this paper adopts tournament selection, two-point crossover and Breeder-GA mutation respectively.
4) To create the new population Q-P, which has a size of 2 × N, the progeny population Q is joined with the existing population P.
5) By fast non-dominated sorting of the new population, the resulting population is divided into non-dominated layers, and normalized by scaling the target values of all solutions on the non-dominated front in the range of [0,1].
6) Repeat Steps 3-5 until the termination condition is satisfied.

Appendix C. Value-based DQN algorithm

The DQN algorithm aims to maximize the value function and continuously update the network parameters during the learning period, so as to gain the optimum policy. Specifically, the main steps are as follows:

In order to determine the best course of action, the DQN algorithm continually updates the network parameters while attempting to maximize the value function. In particular, the following are the primary steps:

1) Initialize the experience pool, and two neural networks (i.e., the evaluation network and the target network) with the same structure and parameters.
2) According to the current state to select the action, and execute this action to obtain the corresponding reward and the next moment of state.
3) The current state, action, reward and next state are stored in the experience pool, and the new memory covers the old memory.
4) The network parameters are updated according to the TD error of the two networks.
5) Repeated iterations make the DQN network parameters gradually stable.

References

[1] Y. Lai, Y. Xu, D. Mai, Y. Fan, F. Yang, Optimized large-scale road sensing through crowdsourced vehicles, IEEE Trans. Intell. Transp. Syst. 23 (4) (2022) 3878–3889.
[2] G. Marques, R. Pitarma, Noise mapping through mobile crowdsourcing for enhanced living environments, in: Proceedings of the International Conference on Computational Science, 2019, pp. 670–679.
[3] M. Chen, W. Yuan, C. Cao, C. Buehler, D.R. Gentner, X. Lee, Development and performance evaluation of a low-cost portable PM2.5 monitor for mobile deployment, Sensors 22 (7) (2022) 2767.
[4] B. Wu, K. Han, E. Zhang, On the task assignment with group fairness for spatial crowdsourcing, Inf. Process. Manag. 60 (2) (2023) 103175.
[5] W. Huang, P. Li, B. Li, L. Nie, H. Bao, Towards stable task assignment with preference lists and ties in spatial crowdsourcing, Inf. Sci. 620 (2023) 16–30.
[6] M.D. Simoni, M. Winkenbach, Crowdsourced on-demand food delivery: an order batching and assignment algorithm, Transp. Res. Part C Emerg. Technol. 149 (2023) 104055.
[7] T. Song, Y. Tong, L. Wang, J. She, B. Yao, L. Chen, K. Xu, Trichromatic online matching in real-time spatial crowdsourcing, in: Proceedings of the 2017 IEEE 33rd International Conference on Data Engineering (ICDE), 2017, pp. 1009–1020.
[8] B. Zhao, H. Dong, Y. Wang, T. Pan, PPO-TA: adaptive task allocation *via* proximal policy optimization for spatio-temporal crowdsourcing, Knowl. Based Syst. 264 (2023) 110330.
[9] F. Song, T. Ma, A location privacy protection method in spatial crowdsourcing, J. Inf. Secur. Appl. 65 (2022) 103095.
[10] Z. Liu, K. Li, X. Zhou, N. Zhu, Y. Gao, K. Li, Multi-stage complex task assignment in spatial crowdsourcing, Inf. Sci. 586 (2022) 119–139.
[11] L. Gao, Y. Gan, B. Zhou, M. Dong, A user-knowledge crowdsourcing task assignment model and heuristic algorithm for expert knowledge recommendation systems, Eng. Appl. Artif. Intell. 96 (2020) 103959.
[12] S. Ceschia, K. Rioters, G. Demartini, S. Mizzaro, L. Di Gaspero, A. Schaerf, Task design in complex crowdsourcing experiments: item assignment optimization, Comput. Oper. Res. 148 (2022) 105995.
[13] Z. Wu, L. Peng, C. Xiang, Assuring quality and waiting time in real-time spatial crowdsourcing, Decis. Support Syst. 164 (2023) 113869.
[14] Y. Sun, M. Liu, L. Huang, N. Xie, L. Zhao, W. Tan, An embedding-based deterministic policy gradient model for spatial crowdsourcing applications, in: Proceedings of the 2021 IEEE 24th International Conference on Computer Supported Cooperative Work in Design (CSCWD), 2021, pp. 1268–1274.
[15] L. Sun, X. Yu, J. Guo, Y. Yan, X. Yu, Deep reinforcement learning for task assignment in spatial crowdsourcing and sensing, IEEE Sens. J. 21 (22) (2021) 25323–25330.
[16] C. Wu, W. Bi, H. Liu, Proximal policy optimization algorithm for dynamic pricing with online reviews, Expert Syst. Appl. 213 (2023) 119191.
[17] M. Lin, T. Chen, H. Chen, B. Ren, M. Zhang, When architecture meets AI: a deep reinforcement learning approach for system of systems design, Adv. Eng. Inf. 56 (2023) 101965.
[18] G. Ridolfi, E. Mooij, D. Cardile, S. Corpino, G. Ferrari, A methodology for system-of-systems design in support of the engineering team, Acta Astronautica 73 (2012) 88–99.
[19] G. Marwaha, M. Kokkolaras, System-of-systems approach to air transportation design using nested optimization and direct search, Struct. Multidiscip. Optim. 51 (4) (2014) 885–901.
[20] A.K. Raz, C.R. Kenley, D.A. DeLaurentis, A System-of-systems perspective for information fusion system design and evaluation, Inf. Fusion 35 (2017) 148–165.
[21] C. Carnevale, L. Sangiorgi, E. De Angelis, R. Mansini, M. Volta, A system of systems for the optimal allocation of pollutant monitoring sensors, IEEE Syst. J. 16 (4) (2022) 6393–6400.
[22] B.C. Watson, Z.B. Morris, M. Weissburg, B. Bras, System of system design-for-resilience heuristics derived from forestry case study variants, Reliab. Eng. Syst. Saf. 229 (2023) 108807.
[23] Y. Huang, A. Luo, T. Chen, M. Zhang, B. Ren, Y. Song, When architecture meets RL +EA: a hybrid intelligent optimization approach for selecting combat system-of-systems architecture, Adv. Eng. Inf. 58 (2023) 102209.
[24] M. Lin, T. Chen, B. Ren, H. Chen, M. Zhang, D. Guo, CADer: a deep reinforcement learning approach for designing the communication architecture of system of systems, IEEE Trans. Intell. Veh. 8 (5) (2023) 3405–3417.
[25] D. Lee, S. Lee, N. Masoud, M.S. Krishnan, V.C. Li, Digital twin-driven deep reinforcement learning for adaptive task allocation in robotic construction, Adv. Eng. Inf. 53 (2022) 101710.
[26] S.S. Bhatti, X. Gao, G. Chen, General framework, opportunities and challenges for crowdsourcing techniques: a comprehensive survey [J], J. Syst. Softw. 167 (2020) 110611.
[27] Y. Tong, L. Wang, Z. Zhou, B. Ding, L. Chen, J. Ye, K. Xu, Flexible online task assignment in real-time spatial data [J], in: Proceedings of the VLDB Endowment 10, 2017, pp. 1334–1345.
[28] W. Gong, B. Zhang, C. Li, Task assignment in mobile crowdsensing: present and future directions [J], IEEE Netw. 32 (4) (2018) 100–107.
[29] L. Islam, S.T. Alvi, Obstacles of mobile crowdsourcing a survey, in: Proceedings of the 2019 IEEE Pune Section International Conference (PuneCon), 2019, pp. 1–4.
[30] X. Zhang, H. Cui, Z. Yu, B. Guo, Environment-driven task allocation in heterogeneous spatial crowdsourcing, in: Proceedings of the GLOBECOM 2022 - 2022 IEEE Global Communications Conference, 2022, pp. 3569–3574.
[31] Y. Zhao, K. Zheng, Y. Cui, H. Su, F. Zhu, X. Zhou, Predictive task assignment in spatial crowdsourcing: a data-driven approach, in: Proceedings of the 2020 IEEE 36th International Conference on Data Engineering (ICDE), 2020, pp. 13–24.
[32] Q. Zhang, Y. Wang, Z. Cai, X. Tong, Multi-stage online task assignment driven by offline data under spatio-temporal crowdsourcing, Digital Commun. Netw. 8 (4) (2022) 516–530.
[33] Q. Zhang, Y. Wang, G. Yin, X. Tong, A.M.V.V. Sai, Z. Cai, Two-stage bilateral online priority assignment in spatio-temporal crowdsourcing, IEEE Trans. Serv. Comput. (2022) 1–14, https://doi.org/10.1109/tsc.2022.3197676.
[34] Q. Jiang, D. Liu, H. Zhu, Y. Qiao, B. Huang, Quasi Group role assignment with role awareness in self-service spatiotemporal crowdsourcing, IEEE Trans. Comput. Social Syst. 9 (5) (2022) 1456–1468.
[35] Y. Wang, Y. Tong, C. Long, P. Xu, K. Xu, W. Lv, Adaptive dynamic bipartite graph matching: a reinforcement learning approach, in: Proceedings of the 2019 IEEE 35th International Conference on Data Engineering (ICDE), 2019, pp. 1478–1489.
[36] C. Shan, N. Mamoulis, R. Cheng, G. Li, X. Li, Y. Qian, An end-to-end deep RL framework for task arrangement in crowdsourcing platforms, in: Proceedings of the 2020 IEEE 36th International Conference on Data Engineering (ICDE), 2020, pp. 49–60.
[37] C.H. Liu, Y. Zhao, Z. Dai, Y. Yuan, G. Wang, D. Wu, K.K. Leung, Curiosity-driven energy-efficient worker scheduling in vehicular crowdsourcing: a deep reinforcement learning approach, in: Proceedings of the 2020 IEEE 36th International Conference on Data Engineering (ICDE), 2020, pp. 25–36.
[38] P. Zhao, X. Li, S. Gao, X. Wei, Cooperative task assignment in spatial crowdsourcing *via* multi-agent deep reinforcement learning, J. Syst. Archit. 128 (2022) 102551.

[39] H. Lin, S. Garg, J. Hu, G. Kaddoum, M. Peng, M.S. Hossain, Blockchain and deep reinforcement learning empowered spatial crowdsourcing in software-defined internet of vehicles, IEEE Trans. Intel. Transp. Syst. 22 (6) (2021) 3755–3764.
[40] Y. Ren, W. Liu, A. Liu, T. Wang, A. Li, A privacy-protected intelligent crowdsourcing application of IoT based on the reinforcement learning, Future Gen. Comput. Syst. 127 (2022) 56–69.
[41] J. Quan, N. Wang, An optimized task assignment framework based on crowdsourcing knowledge graph and prediction, Knowl. Based Syst. 260 (2023) 110096.
[42] C.H. Liu, Z. Dai, Y. Zhao, J. Crowcroft, D. Wu, K.K. Leung, Distributed and energy-efficient mobile crowdsensing with charging stations by deep reinforcement learning, IEEE Trans. Mobile Comput. 20 (1) (2021) 130–146.
[43] C. Xu, W. Song, Intelligent task allocation for mobile crowdsensing with graph attention network and deep reinforcement learning, IEEE Trans. Netw. Sci. Eng. 10 (2) (2023) 1032–1048.
[44] C. Piao, C.H. Liu, Energy-efficient mobile crowdsensing by unmanned vehicles: a sequential deep reinforcement learning approach, IEEE Internet Things J. 7 (7) (2020) 6312–6324.
[45] Y. Gao, Z. Wang, Z. Li, Z. Li, Task migration based on deep reinforcement learning in mobile crowdsourcing, in: Proceedings of the 2022 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom), 2022, pp. 410–417.
[46] E. Yuan, S. Cheng, L. Wang, S. Song, F. Wu, Solving job shop scheduling problems *via* deep reinforcement learning, Appl. Soft Comput. 143 (2023) 110436.
[47] Y. Liu, X. Zuo, G. Ai, Y. Liu, A reinforcement learning-based approach for online bus scheduling, Knowl. Based Syst. 271 (2023) 110584.
[48] M. Wang, Y. Wang, A.M.V.V. Sai, Z. Liu, Y. Gao, X. Tong, Z. Cai, Task assignment for hybrid scenarios in spatial crowdsourcing: a Q-learning-based approach, Appl. Soft Comput. 131 (2022) 109749.
[49] S. Hang, S. Ontaon, A closer look at invalid action masking in policy gradient algorithms, (2020) arXiv preprint. doi:10.48550/arXiv.2006.14171.
[50] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal Policy Optimization Algorithms, 2017, https://doi.org/10.48550/arXiv.1707.06347.
[51] L. Engstrom, A. Ilyasl, S. Santurkar, D. Tsipras, F. Janoos, L. Rudolph, A. Madry, Implementation matters in deep policy gradients: a case study on PPO and TRPO. (2020) arXiv preprint. doi:10.48550/arXiv.2005.12729.
[52] P.S. Mashhadi, S. Nowaczyk, S. Pashami, Parallel orthogonal deep neural network, Neural Netw. 140 (2021) 167–183.
[53] D. Wang, K. Liu, H. Xiong, Y. Fu, Online POI recommendation: learning dynamic geo-human interactions in streams, IEEE Trans. Big Data 9 (3) (2023) 832–844.
[54] Z. Chen, R. Fu, Z. Zhao, Z. Liu, L. Xia, L. Chen, P. Cheng, C.C Cao, Y. Tong, C. J. Zhang, gMission: a general spatial crowdsourcing platform, in: Proceedings of the VLDB 7, 2014.
[55] A. Yadav, S. Mishra, A.S. Sairam, A multi-objective worker selection scheme in crowdsourced platforms using NSGA-II, Expert Syst. Appl. 201 (2022) 116991.
[56] X. Wu, L. Jiang, W. Zhang, C. Li, Three-way decision-based noise correction for crowdsourcing, Int. J. Approx. Reason. 160 (2023) 108973.