# Gradient-Guided Credit Assignment and Joint Optimization for Dependency-Aware Spatial Crowdsourcing

**Yafei Li[1]*, Wei Chen[1]*, Jinxing Yan[1], Huiling Li[2], Lei Gao[1], Mingliang Xu[1]†**

[1]School of Computer Science and Artificial Intelligence, Zhengzhou University
[2]Department of Computer Science, Hong Kong Baptist University
{ieyfli, iexumingliang}@zzu.edu.cn, {chenweizzu, jxyan, leigao}@gs.zzu.edu.cn, cshlli@comp.hkbu.edu.hk

## Abstract

Dependency-aware spatial crowdsourcing (DASC) addresses the unique challenges posed by subtask dependencies in spatial task assignments. This paper investigates the task assignment problem in DASC and proposes a two-stage Recommend and Match Optimization (RMO) framework, leveraging multi-agent reinforcement learning for subtask recommendation and a multi-dimensional utility function for subtask matching. The RMO framework primarily addresses two key challenges: credit assignment for subtasks with interdependencies and maintaining overall coherence between subtask recommendation and matching. Specifically, we employ meta-gradients to construct auxiliary policies and establish a gradient connection between two stages, which can effectively address credit assignment and joint optimization of subtask recommendation and matching, while concurrently accelerating network training. We further establish a unified gradient descent process through gradient synchronization across recommendation networks, auxiliary policies, and the matching utility evaluation function. Experiments on two real-world datasets validate the effectiveness and feasibility of our proposed approach.

## Introduction

With the popularity of smart mobile devices, spatial crowdsourcing (SC) as a promising computing paradigm has become increasingly prosperous (Tong et al. 2020; Kang et al. 2023; Lyu et al. 2023), where workers perform spatial tasks assigned by the SC platform for payments. Recently, the increasing complexity and requirements of spatial tasks have induced the emergence of a sophisticated paradigm in SC, namely Dependency-Aware Spatial Crowdsourcing (DASC) (Ni et al. 2020; Liu et al. 2022). DASC is characterized by its intricate task structure and complex execution requirements, further enhancing the challenges and potential applications of traditional SC. In DASC, spatial tasks consist of multiple interdependent subtasks that require a specific execution order, and each subtask may require different worker skills. Task completion is achieved only when all constituent subtasks are finished, and the platform receives remuneration at this time.

To illustrate DASC, consider a telecommunications network maintenance scenario with three interdependent subtasks: on-site data collection, equipment preparation at a warehouse, and tower repair. Obviously, repair can only begin after data collection and equipment preparation are complete, and each subtask requires different skills and is indispensable to successful telecommunications network maintenance. As such, DASC diverges significantly from typical spatial crowdsourcing models because, regarding spatial tasks, it emphasizes subtask dependencies, skill-specific assignments, and comprehensive completion of all the corresponding subtasks. In this paper, we aim to maximize the platform's total revenue by optimizing task assignment policies. The optimization problem in DASC is inherently more complex than traditional spatial crowdsourcing due to the additional dimensions of inter-task dependencies and heterogeneous skill requirements.

Currently, there is a relative paucity of research focused on DASC, whereas standard SC problems have been extensively studied. Our approach involves converting the DASC problem into a standard SC problem, thereby leveraging established SC solutions for the matching process. However, this conversion presents unique challenges: if all subtasks are incorporated into the matching process, task completion efficiency would be substantially impeded as the start time of subtasks and their predecessors cannot be the same. Conversely, if the matching is limited to subtasks without prerequisites, it neglects the inherent interdependencies among tasks and may fail to utilize available workers fully. To this end, we propose a novel two-stage Recommend and Match Optimization (RMO) framework (Fig.1), which initially analyzes task graph structures and features to recommend subtasks and subsequently strategically matches these subtasks with appropriate workers.

Specifically, we design a multi-agent network employing reinforcement learning to explore optimal recommendation policies for subtasks in the first stage. Since the completion of a spatial task consists of the successful execution of all the corresponding subtasks with specific dependencies(Wang et al. 2020; Liu et al. 2022), the task recommendation network must learn to evaluate a subtask's contribution to both its associated spatial task and the crowdsourcing platform holistically. As such, we categorize the contribution assessment of subtasks as a credit assignment prob-

*These authors contributed equally.
†Corresponding author.

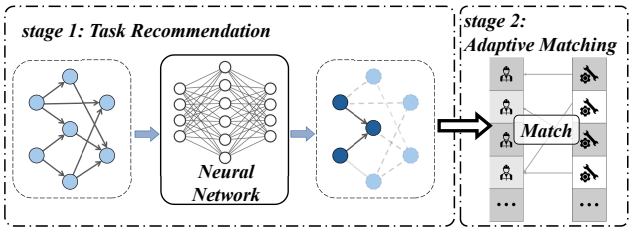



Figure 1: RMO Framework. Stage 1 employs reinforcement learning to build recommendation networks for task analysis and subtask recommendation; Stage 2 utilizes adaptive algorithms to match workers with the recommended subtasks.

lem(Chen et al. 2023). In the second stage of RMO, we develop a utility evaluation method that incorporates time, distance, and skill, where the challenge lies in determining the weight for each factor in evaluating utility.

To further address these challenges, we extend the application of meta-gradients (Xu, van Hasselt, and Silver 2018) to both stages of the RMO framework. This approach not only guides the policy in learning an effective credit assignment method but also bridges the two stages, facilitating joint optimization of the recommendation network and the utility function. By leveraging meta-gradients, we construct a highly robust unified decision system, effectively integrating credit assignment with the RMO framework's dual-stage process. Our contributions are summarized as follows:

- We investigate the DASC problem, where a spatial task consists of subtasks with specific dependencies. To address it, we propose the RMO framework integrating reinforcement learning-based subtask recommendation and utility-adaptive subtask assignment.
- We propose a new credit assignment strategy optimization method leveraging meta-gradients, effectively addressing the challenges of sparse rewards in reinforcement learning-based subtask recommendation. This approach can mitigate training difficulties and accelerate the learning process.
- We propose a new joint optimization method using meta-gradients, enabling unified training of interconnected components to mitigate inconsistent optimization directions and enhance model generalization and robustness.
- We conduct extensive experiments on two real-world datasets to validate the effectiveness of our proposed algorithms. The experimental results show that our proposed RMO exhibits $2.2\times$ to $6.6\times$ higher than baselines in total revenue.

## Related Work

In this section, we review the related works in spatial crowdsourcing regarding dependency awareness and the application of reinforcement learning.

### Dependency-Aware Spatial Crowdsourcing

Extant work on DASC has primarily focused on addressing the inherent challenges within this complex paradigm. Ni et al. (2020) conceptualized DASC as a strategic game, leveraging game theory principles to optimize task-worker matching policies. Similarly, Liu et al. (2022) employed game theory methods and introduced a filtering module to enhance matching effectiveness. Further advancing the field, Yao, Yang, and Xu (2022) incorporated worker preferences into their model, developing three algorithms with provable constant competitive ratios. Despite these significant contributions, current approaches exhibit limitations in their comprehensive analysis of task graph dependencies. For instance, Wang et al. (2020) proposed a BFS-based priority scheduling algorithm in conjunction with an evolutionary multitasking-based approach. However, this method is limited to managing dependency-aware tasks within a single static complex task. Consequently, much of the current research is often constrained to tasks with relatively simple structures or a limited number of complex tasks, failing to adequately address the diverse and intricate graph structures characteristic of more comprehensive DASC scenarios. Moreover, despite the inherent complexity of the DASC problem, there remains a notable absence of research exploring the application of Learning-Based Algorithms to this domain. This gap in the literature underscores the need for more sophisticated methodologies capable of handling the multifaceted nature of DASC, particularly in scenarios involving complex dependency structures and dynamic environments.

### Reinforcement Learning-based Task Assignment

Reinforcement learning (RL) is increasingly applied in crowdsourcing to optimize task assignment and scheduling through adaptive policy learning in dynamic environments. A common approach is to model task assignment as a Markov Decision Process. For instance, Zong et al. (2022) developed an end-to-end multi-agent system or pickup and delivery problems. In air-ground spatial crowdsourcing, Wang et al. (2023) introduced a multi-center attention-based graph convolutional network for communication. In contrast, Ye et al. (2023) proposed a heterogeneous multi-agent framework exploring both individual and collaborative environments. Jiang et al. (2023) designed a fairness-aware concurrent dispatch system for instant delivery services. For ride-hailing order dispatching, Zhang et al. proposed an offline deep reinforcement learning framework with a method for managing dynamic nondeterministic action spaces (Zhang et al. 2024)Beyond direct application in decision-making, RL enhances established matching methods. It can determine the sliding window size of decision time steps (Li et al. 2022b) and optimize existing algorithms based on game theory (Li et al. 2023), large-scale search (Li et al. 2022a), and combinatorial optimization (Tong et al. 2021), etc.

## Problem Definition

Generally, we define the DASC problem on a road network represented by a graph $G = (N, E)$, where $n \in N$ represents road intersections, and $e_{ij} \in E$ denotes roads connecting two intersections. Next, we define other entities related to the DASC problem.

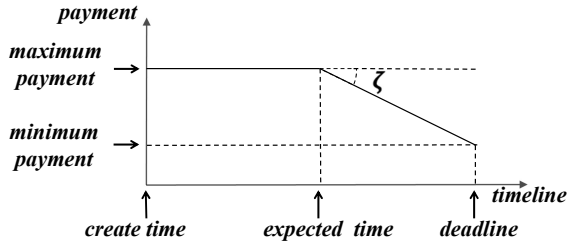



Figure 2: payment of spatial task.

**Worker.** A worker $w \in W$ is described by the tuple $w = (l, K)$, where $l$ denotes the worker's location and $K$ represents the set of skills possessed by the worker.

**Spatial Task.** Spatial task $q$ is denoted as a tuple $q = (\mathcal{T}, \mathcal{C}, t_c, t_d, t_e, \rho)$. Specifically, $\mathcal{T}$ represents the set of spatial subtasks, while $\mathcal{C}$ denotes the dependency constraints among these subtasks. Additionally, the temporal aspects of the task are captured by $t_c$, $t_d$, and $t_e$, which indicate the creation timestamp, deadline, and expected completion time, respectively. Furthermore, $\rho$ signifies the payment obtained upon successful completion of the task.

The platform's revenue is subject to reduction if spatial tasks are not completed before the expected completion time. This relationship is expressed as follows:

$$\varrho = (1 - \zeta \cdot \Delta t)\rho - \kappa \cdot \sum_{\tau \in \mathcal{T}} d^{\mathcal{M}(w,\tau)}. \tag{1}$$

In this equation, $\varrho$ represents the profit of an individual spatial task, $\zeta$ serves as a penalty factor for time, and $\Delta t$ indicates the time difference relative to the expected completion time. Additionally, $\kappa$ denotes the cost of worker movement per kilometer, while $d^{\mathcal{M}(w,\tau)}$ represents the distance that worker $w$ needs to travel to complete subtask $\tau$. It's important to note that if a spatial task is completed before the expected completion time, the time penalty does not apply, and consequently, $\Delta t$ is equal to 0. For a visual representation of task revenue, refer to Fig.2.

**Spatial Subtask.** A spatial subtask $\tau$ is denoted as a tuple $\tau = (l, K, t)$, where $l$ is the location of the task, $K$ represents the set of skills required to complete the task, and $t$ denotes the duration needed for task completion.

Spatial subtasks are completed by exactly one worker, and the worker must possess all the skills required by the subtask, denoted as $K^\tau \subseteq K^w$. Moreover, if a subtask does not require any specific work skills, i.e., $K^\tau = \phi$, then there are no restrictions on the workers for that subtask.

**DASC Problem.** Given a set of workers $W$ and a stream of tasks $Q$, the goal of the DASC problem is to find the best matching plan $\mathcal{M} \subseteq W \times \mathcal{T}$ such that the total revenue of platform $\mathrm{E}(\mathcal{M})$ is maximized,

$$\mathrm{E}(\mathcal{M}) = \sum_{q \in Q} \varrho^q, \tag{2}$$

where for each matching $(w, \tau) \in \mathcal{M}$, should satisfy the following constraints:

- **Skill Constraint.** A worker $w$ can only be assigned to subtask $\tau$ if $w$ has all the required skills.
- **Order Constraint.** Subtasks must satisfy the dependency constraints of the spatial task.
- **Deadline Constraint.** All subtasks of spatial tasks are required to be completed within the deadlines.

# Method

To address the DASC problem, we propose a new approach that leverages meta-gradients to guide credit assignment and joint optimization within the RMO framework. Our approach is summarized as follows: (1) In the subtask recommendation stage, we develop a multi-agent reinforcement learning model enhanced by meta-gradient methods, which optimizes policy updates and facilitates effective credit assignment among agents. (2) In the adaptive matching stage, we employ the Kuhn-Munkres (KM) algorithm for maximum weight matching and utilize meta-gradients to optimize the KM algorithm's hyperparameters which enable joint training of the entire framework.

## Task Recommendation Module

We establish a multi-agent recommendation network system based on the Multi-Agent Proximal Policy Optimization (MAPPO) algorithm (Yu et al. 2022), where each subtask functions as an autonomous agent. The task recommendation process is modeled using two distinct Markov Decision Processes (MDPs).

**MDP with Sparse Reward.** This MDP is formulated to capture the DASC's base attribute, where payment is received only upon completion of all subtasks It is represented by the tuple $(\mathcal{N}, \mathcal{S}, \mathcal{O}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma)$. Detailed descriptions of this MDP's components are provided below.

*Agent* $n \in \mathcal{N}$: Each spatial subtask $\tau$ is treated as an agent $n$, which appears and disappears based on task publication and completion. Moreover, all agents are assumed to share an identical network structure.

*State* $s \in \mathcal{S}$: The global state $s_t$ is represented by a vector $(Q, W)$, which includes both worker and spatial task real-time information.

*Observation* $o \in \mathcal{O}$: The observation $o_t^n$ for agent $n$ at time $t$ is represented as a triplet $(\mathcal{T}near, Wnear, q)$, denoting the conditions of surrounding workers, surrounding subtasks, and the agent's own real-time information.

*Action* $a \in \mathcal{A}$: The action space is defined as $\{0, 1\}$, where 0 indicates no recommendation and 1 indicates a recommendation.

*Reward* $r \in \mathcal{R}$: The reward is assigned only to the final subtask of a task upon successful matching, with the task's revenue being attributed as the reward for this subtask, represented as $r_q = \varrho$. If a subtask is not recommended or is recommended but is not the final subtask, then $r_q = 0$. It is critical to note that if the task recommendation fails to achieve a match, then $r_q = -1$.

*State transition distribution* is defined as $\mathcal{P} : \mathcal{S} \times \mathcal{A}_1 \times \ldots \times \mathcal{A}_n \times \mathcal{S} \to [0, 1]$. The function $\mathcal{P}(s, a, s')$ represents the probability of transitioning to state $s'$ given that action $a$ is executed in state $s$.

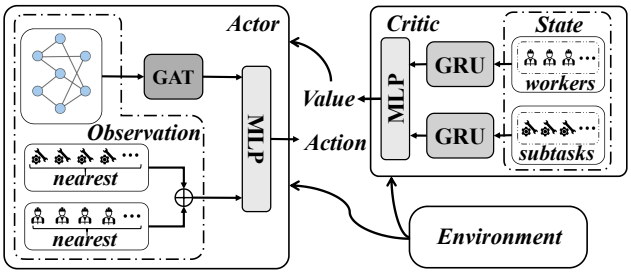



Figure 3: Recommendation network structure.

*Discount factor* $\gamma \in [0, 1]$: The discount factor balances the current reward with future rewards. A value closer to 1 emphasizes the importance of long-term rewards.

**MDP with Reward Shaping.** This MDP converts revenue into rewards for individual subtasks, differing from the MDP with sparse reward only in its reward structure.

*Reward* $r \in \mathcal{R}$: If agent $n$ does not recommend subtask $\tau$, the reward $r_\tau$ is 0. If subtask $\tau$ is recommended but fails to match with a worker, the reward is -1. Conversely, if the subtask is recommended and successfully matches with a worker, the reward is calculated as follows:

$$r_\tau = \begin{cases} \rho_{max}/|\mathcal{T}| - \kappa \cdot d^{\mathcal{M}}, & t_\tau \le t_\tau^e \\ \rho_{min}/|\mathcal{T}| - \kappa \cdot d^{\mathcal{M}}, & t_\tau^e < t_\tau < t_\tau^d \\ 0, & t_\tau \ge t_\tau^d \end{cases}. \quad (3)$$

The variables $t^d$ and $t^e$ signify a subtask's deadline and expected completion time, respectively. And $t_\tau$ is finish time. By applying the critical path method to the deadline $t_q^d$ and expected completion time $t_q^e$ of a crowdsourcing task $q$, we can infer the corresponding values for individual subtasks.

**Recommendation Network.** We have implemented a task recommendation network using the Actor-critic framework, depicted in Fig.3.

*Actor*: The Actor network uses local observations to decide which actions to take. To effectively handle the dependency information of subtasks, a Graph Attention Network (GAT) (Veličković et al. 2017) is employed. The processed node features, along with the top-k selected subtasks and worker information, are then input into a Multi-Layer Perceptron (MLP) for further processing.

*Critic*: The Critic network assesses global information to inform the decision-making processes of the Actor network. To process worker and subtask information separately, two distinct Gated Recurrent Units (GRU) (Cho et al. 2014) are utilized. The outputs from these GRUs are then combined and fed into an MLP to evaluate the current state.

## Collaborative Policy Optimization

This section introduces a meta-gradient method to link two policies derived from distinct MDPs, enabling simultaneous parameter updates in a single gradient step. We begin with a brief review of policy network updates in MAPPO.

The objective function of the policy in MAPPO is to maximize the expected reward while constraining the extent of

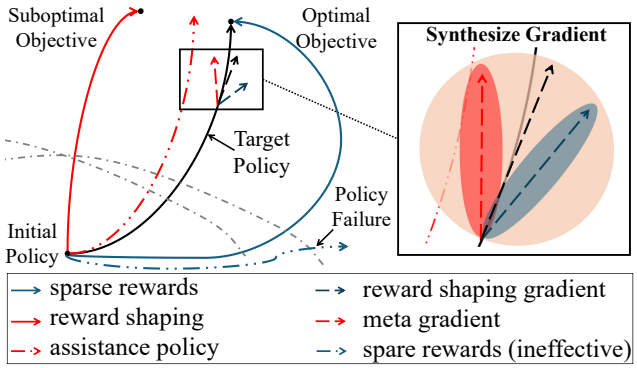



Figure 4: Collaborative policy optimization employs the target policy to constrain the update of the policy derived from reward shaping, then applying the meta-gradient obtained from the reward shaping process to refine the target policy.

policy updates. MAPPO employs a clipped probability ratio loss function to ensure that updates stay within a reasonable range. The specific formula is as follows:

$$J(\theta) = \mathrm{E}_{a\sim\pi_\theta(\cdot|o)} \min\left[\psi A^n, \mathrm{clip}\left(\psi, 1-\epsilon, 1+\epsilon\right) A^n\right], \quad (4)$$

where $\psi = \frac{\pi_\theta(a_t|o_t)}{\pi_{\theta_{old}}(a_t|o_t)}$ is the probability ratio between the current policy and the old policy and $\epsilon$ is a hyperparameter that controls the clipping range.

Our primary focus is the policy network $\theta$ trained by the MDP with sparse reward. However, the sparse rewards lead to protracted training periods and challenges in credit assignment (Lin et al. 2024). In extreme cases, policies derived from these sparse rewards may impede the convergence of network training. The policy network $\hat{\theta}$ trained by the MDP with reward shaping can mitigate the drawbacks above. However, $\hat{\theta}$ may diverge from the core DASC optimization objectives. Thus, we propose a gradient-guided method to train the network, leveraging the benefits of both MDPs.

First, $\theta$ regulates $\hat{\theta}$ to prevent excessive deviation from the target policy(Wang et al. 2022). The objective function for $\hat{\theta}$ has been restructured by replacing the similarity ratio $\psi$ with the policy similarity between $\theta$ to $\hat{\theta}$, as follows:

$$\psi_{\hat{\theta}} = \frac{\pi_{\hat{\theta}}(a_t^n|o_t^n)}{\pi_\theta(a_t^n|o_t^n)}. \quad (5)$$

To construct gradient connections from $\hat{\theta}$ to $\theta$, we utilize the meta-gradient method to optimize $\theta$. We treat the parameter of $\theta$ as hyperparameters in the trainer of policy $\hat{\theta}$. This perspective enables us to compute the meta-gradient of $\theta$ with respect to $\hat{\theta}$, effectively creating a feedback loop between the two policy networks. The meta-gradient can be expressed as:

$$\nabla_\theta J(\hat{\theta}) = \nabla_{\hat{\theta}} J(\hat{\theta}) \cdot \nabla_\theta f(\omega, \hat{\theta}, \theta), \quad (6)$$

while $f(\cdot)$ denotes the update step size for $\theta$, with $\omega$ representing a sequence of experiences, such as $\omega =$

$(S_t, O_T, A_t, \ldots)$. For clarity, we use the Stochastic Gradient Descent (SGD) optimizer as an example to illustrate the construction of $f(\cdot)$, hence $\nabla_\theta f(\cdot)$ is defined as follows:

$$\nabla_\theta f(\omega, \hat{\theta}, \theta) = -\frac{\mu_f}{2} \frac{\partial^2 J(\hat{\theta})}{\partial \hat{\theta} \cdot \partial \theta}. \tag{7}$$

while $\mu_f$ is the learning rate of the SGD. Based on the formula above, we compute the meta-gradient $\nabla_\theta J(\hat{\theta})$ for the parameter $\theta$. Additionally, $\nabla_\theta J(\theta)$ represents the gradient of the policy $\theta$. These two gradients are combined to determine the update gradient for $\theta$:

$$\nabla J = \nabla_\theta J(\theta) + \xi \cdot \nabla_\theta J(\hat{\theta}). \tag{8}$$

Here, $\xi$ is used to scale the meta-gradient. As iterations progress, $\xi$ asymptotically approaches zero, enabling rapid initial updates to $\theta$ while reducing divergence from the objective function in later stages.

## Matching Utility

In the context of a given subtask $\tau$, where multiple available workers may exist for potential matching, the introduction of a matching utility function facilitates the evaluation of the quality of worker-subtask alignment. It's essential to underscore that utility assessments are restricted to feasible matches. Should a worker $w$ fail to satisfy the criteria for subtask $\tau$, their matching utility is automatically assigned a value of zero. We formulate utility functions by considering multiple dimensions of matching.

**Skill Compatibility Utility.** The skill compatibility utility is employed to evaluate the compatibility of skills. Its calculation formula is as follows:

$$u_{skill} = \frac{|K^\tau|}{|K^w|}, \tag{9}$$

where $|K^\tau|$ represents the number of skills required by subtask $\tau$, and $|K^w|$ denotes the skills possessed by worker $w$.

**Distance Utility.** The Distance Utility function can be computed using the following formula:

$$u_{dist} = \frac{L^w - \Delta_{dist}(l^w, l^\tau)}{L^w}, \tag{10}$$

where $L^w$ represents the maximum travel distance and $\Delta_{dist}(\cdot)$ represents the distance between $w$ and $\tau$.

**Time Remaining Utility.** The time remaining utility is utilized to evaluate the urgency level of a task. Its computational formula is expressed as follows:

$$u_{time} = \frac{1}{e} \cdot e^{(t-t^e)/(t^d-t^e)}. \tag{11}$$

Here, $t$ denotes the current timestamp. As time progresses, when $t$ is less than $t^e$, $u_{time}$ increases gradually. However, once $t$ exceeds $t^e$, it adversely affects the payment of task $q$. Therefore, it is crucial for $u_{time}$ to exhibit a more rapid increase thereafter.

**Matching Utility.** Given a worker set $W = \{w_1, \cdots w_n\}$ and a subtask set $\mathcal{T} = \{\tau_1, \cdots \tau_n\}$, If worker $w$ can be matched with subtask $\tau$, the matching utility of a matching instance $(w, \tau) \in W \times \mathcal{T}$ is defined as

$$u = v_1 \cdot u_{skill} + v_2 \cdot u_{dist} + v_3 \cdot u_{time}, \tag{12}$$

where $v_i$ are hyperparameters used to balance multiple utilities, each in the range $[0, 1]$ and satisfying $\sum v_i = 1$.

## Utility Combination Parameter Optimization

To streamline the computation of the hyperparameter $v_x$ of utility, we initially replace them with an auxiliary parameter $\eta = \{\alpha, \beta\}$, where each component ranges from $[0, 1]$:

$$\begin{cases} v_1 & = \alpha \cdot \beta \\ v_2 & = \alpha \cdot (1 - \beta) \\ v_3 & = 1 - \alpha \end{cases}. \tag{13}$$

The meta-gradient for parameter $\eta$ is constructed similarly to collaborative policy optimization. Specifically, it is derived from the policy $\hat{\theta}$ trained using reward shaping. The update formula is given by:

$$\Delta\eta = -\mu_\eta \cdot \nabla_{\hat{\theta}} J(\hat{\theta}) \cdot \nabla_\eta f(\omega, \hat{\theta}, \eta), \tag{14}$$

where $\mu_\eta$ is the learning rate for updating meta parameter $\eta$. In Eq.(14), the primary challenge is computing $\nabla_\eta f(\cdot)$, We can use the chain rule to relate $f(\cdot)$ to the utility function $u$: Specifically, the gradient of $f(\cdot)$ with respect to $\eta$ can be expressed as:

$$\nabla_\eta f(\omega, \hat{\theta}, \eta) = \frac{\partial f}{\partial u} \frac{\partial u}{\partial \eta}. \tag{15}$$

The derivative $\partial u / \partial \eta$ is easily solvable, whereas $\partial f / \partial u$ remains computationally infeasible. However, in practice, a high match utility $u$ results in the environment rewarding the agent with a relatively high reward $r$, indicating a positive correlation between $r$ and $u$. Therefore, we substitute $\partial f / \partial r$ with $\partial f / \partial u$. To clarify, we use the SGD optimizer as an example to illustrate $f(\cdot)$. Thus, $f(\cdot)$ can be expressed as $f(\cdot) = -\frac{1}{2}\mu_f \nabla_{\hat{\theta}} J(\hat{\theta})$. And the computation method for $\partial f / \partial r$ is as follows:

$$\frac{\partial f}{\partial r} = -\frac{\mu_f}{2} \frac{\partial^2 J}{\partial r \partial \hat{\theta}} = -\frac{\mu_f}{2} \frac{\partial}{\partial \hat{\theta}} \left( \frac{\partial J}{\partial A} \cdot \frac{\partial A}{\partial r} \right). \tag{16}$$

Generalized Advantage Estimation (GAE) is employed to compute the advantage function $A$:

$$A = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}, \tag{17}$$

where $\delta_t$ denotes the Temporal Difference (TD) error (Mnih et al. 2015) at each time step, $\gamma$ is the discount factor, and $\lambda$ modulates the weighting of different time steps. The TD error $\delta_t$ can be calculated using the reward $r_t$ and the value function $V$ as $\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$, $V(s_t)$ represents the expected value of long-term returns, encapsulating the overall policy's sustained gains. It is assumed that the value function is dependent on the state $s_t$ rather than being directly influenced by the immediate reward $r_t$. Therefore, we can further simplify Eq.(16):

$$\frac{\partial A}{\partial r_t} = \frac{\partial \delta_t}{\partial r_t} = 1. \tag{18}$$

Through the above derivation process, we can use the following formula to differentiate $f(\cdot)$ with respect to $\eta$:

$$\nabla_\eta f(\omega, \hat{\theta}, \eta) = -\frac{\mu_f}{2} \frac{\partial}{\partial \hat{\theta}} \left( \frac{\partial J}{\partial A} \right) \cdot \frac{\partial u}{\partial \eta}. \tag{19}$$

Algorithm 1: Meta training algorithm

**Input**: episode $E$, iteration $M$, soft update coefficient $\chi$
**Output**: policy network $\theta$, value network $\varphi$, hyperparameters $\eta$

Initialize policy network $\theta$ and $\hat{\theta}$ value network $\varphi$, hyperparameters $\eta$.
Initialize replay buffer $D$
**for** episode $= 1, \ldots, E$ **do**
  Use $\theta$ to execute actions and collect experience, then store transitions $(s_t, \{o_t^n\}, \{a_t^n\}, \{r_t^n\}, s_{t+1})$ in the data buffer $D$
  **for** $iteration \leftarrow 1$ **to** $M$ **do**
    Sample mini-batch from $D$
    **for** each agent $n \in N$ **do**
      Compute $\psi_{\hat{\theta}}$ and update $\hat{\theta}^n$ by Eq.(4)
      update $\theta^n$ by Eq.(8)
      **if** $\eta$ can update **then**
        Update $\eta$ by Eq.(14)
      **end if**
      Soft update: $\begin{cases} \theta \leftarrow \chi\theta^n + (1-\chi)\theta \\ \hat{\theta} \leftarrow \chi\hat{\theta}^n + (1-\chi)\hat{\theta} \end{cases}$
    **end for**
    Update $\varphi$ by Eq.(20);
  **end for**
**end for**
**return** $\theta, \varphi, \eta$

## Meta Training

Building upon the aforementioned methods, we can utilize meta-gradient-guided policies and hyperparameter optimization for training. Algorithm 1 illustrates the pseudocode for our proposed solution.

We employ the policy network $\theta$ to execute actions and utilize the experience gathered under this policy to jointly train the network parameters $\theta$, $\hat{\theta}$, $\varphi$, and the hyperparameter $\eta$. For the value function, we aim to minimize its prediction error using the mean squared error (MSE) as the loss function:

$$\mathcal{L} = \mathrm{E}_t[(V_\varphi(s_t) - V^{target})^2], \tag{20}$$

where $V_\phi(s_t)$ is the current estimate of the value function. $V^{target}$ is the target value, usually derived from the return and the advantage function.

# Experiments

## Experimental Setup

**Datasets.** We extract road network data from OpenStreetMap[1] for two cities: Chengdu (36,630 nodes, 50,786 edges) and Haikou (20,592 nodes, 49,549 edges). Regarding spatial tasks, we first extracted orders from DiDi[2], comprising 7,065,937 orders for Chengdu and 14,160,170 orders for Haikou, where the pick-up location of the order corresponds to the location of the subtask. Then, spatial tasks were constructed using random Directed Acyclic Graph structures. The task skill model was developed based on Point-of-Interest (POI) types from AutoNavi Maps[3], classified into 10 distinct categories. Each subtask derives 1-3 skill requirements from nearby POIs, following a right-skewed normal distribution ($\mu = 1.5, \sigma^2 = 0.5, \gamma = 10$). Worker skills were modeled according to a normal distribution ($\mu = 3.5, \sigma^2 = 1$), while worker initial locations were randomly initialized using DiDi order data. Code is available at [https://github.com/kiren-costello/dasc].

| Parameter | Value | Parameter | Value |
|---|---|---|---|
| $\mu_\eta$ | $1 \times 10^{-4}$ | $\mu_\theta$ | $1 \times 10^{-3}$ |
| $\mu_\varphi$ | $1 \times 10^{-3}$ | $\chi$ | $1 \times 10^{-3}$ |
| $\gamma$ | 0.99 | $L^w$ | 5 km |
| $\epsilon$ | 0.2 | $\zeta$ | 0.02 |
| $v_w$ | 10 m/s | $\kappa$ | $1 \times 10^{-3}$ |
| $N^w$ | 5 | $N^\tau$ | 5 |
| $\lambda$ | 0.95 | $\xi_0$ | 1 |
| $\lvert\mathcal{T}\rvert_{max}$ | 16 | $\lvert\mathcal{T}\rvert_{min}$ | 6 |

Table 1: Simulation Settings.

**Implementation Details.** We employ the SGD optimizer for training all models, utilizing a learning rate of $1 \times 10^{-4}$. The soft update parameter is set at $1 \times 10^{-3}$, and the hyperparameter update rate is maintained at $1 \times 10^{-3}$. The key parameters for our simulation experiments are presented in Table 1. All experimental procedures are executed on a computational system operating Ubuntu 22.04.2 LTS with Python 3.8, equipped with Intel Core i9-13900K CPU @ 5.80 GHz, NVIDIA GeForce RTX 3090 GPU, and 32 GB RAM.

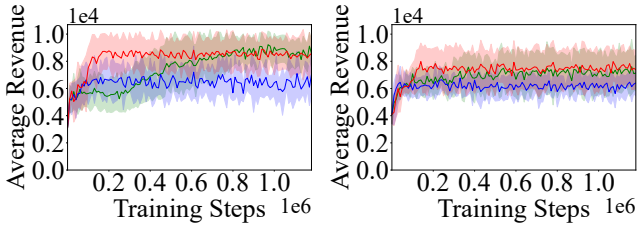
**Baselines.**

- **GREEDY**: greedily assigns each independent subtask (those without precedence constraints) to the nearest available worker.
- **KM** (Kuhn 1955; Munkres 1957): employs the Kuhn-Munkres algorithm to optimally match independent subtasks with workers, maximizing overall assignment efficiency.
- **REC**: within the RMO framework, recommending subtasks but matching them with workers greedily.
- **GAME** (Ni et al. 2020): formulates task assignment as a game-theoretic problem, employing an approximation algorithm that yields provable bounds on dependency-aware task assignment quality.
- **ODTA** (Yao, Yang, and Xu 2022): is a threshold-based randomized and two-stage algorithm enhancing efficiency and effectiveness in dynamic environments.

## Experimental Results

**Analysis of Different Training Approaches.** To evaluate the effectiveness of our proposed approach, we conducted a comparative analysis against two baseline methods: (1) policies trained using only sparse rewards and (2) policies

[1]https://www.openstreetmap.org/
[2]https://www.didiglobal.com/
[3]https://lbs.amap.com/




(a) Chengdu datasets (b) Haikou datasets

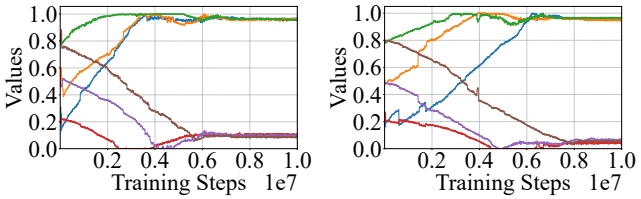
Figure 5: Comparative analysis of different training methods (Green: sparse reward, Blue: reward shaping, Red: meta-gradient-guided).

trained using only reward reshaping. We retained policies at different training steps and tested them on the same experimental dataset across 3 groups, measuring the crowdsourcing platform's average revenue. The experimental results are shown in Fig.5.

During the early training phase, the policy trained with reward shaping achieves higher revenue than others. However, as training progresses, policies trained with meta-gradient-guided and sparse rewards tend to surpass it. Comparing sparse rewards to meta-gradient-guided training, experiments results show that on the Chengdu dataset, the sparse rewards policy achieves performance comparable to the meta-gradient-guided method after $6\times10^5$ training steps. However, on the Haikou dataset, the sparse rewards policy underperforms compared to the meta-gradient-guided training policy, even after $1.0 \times 10^6$ training steps. These experiments show that the meta-gradient-guided method improves policy optimization and accelerates training.

**Analysis of Utility Hyperparameter.** We conducted experiments on two distinct datasets using initial $\eta = \{\alpha, \beta\}$ values of $\{0.2, 0.2\}$, $\{0.5, 0.5\}$, and $\{0.8, 0.8\}$. The results are presented in Fig.6. Despite some variations between different experimental groups, the outcomes converged within a similar range. Based on these results, we selected the hyperparameters for subsequent experiments: $\{0.95, 0.10\}$ for the CD dataset and $\{0.96, 0.06\}$ for the HK dataset.

During the training process, hyperparameters exhibit sudden jumps. caused by factors like suboptimal subtask recommendations or increased platform task traffic. Meta-gradients involve second-order derivatives, which result in heightened sensitivity compared to conventional gradients.

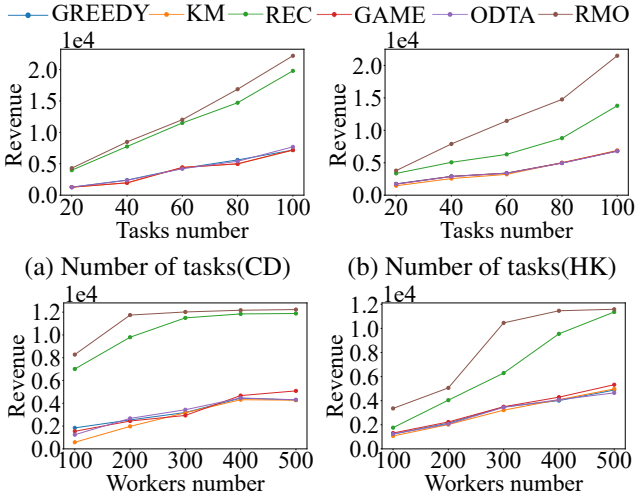



(a) Chengdu datasets (b) Haikou datasets

Figure 6: Training process analysis with varying initial parameters $\eta = \{\alpha, \beta\}$.

| Parameter | Value |
|---|---|
| # of tasks | 20, 40, **60**, 80, 100 |
| # of workers | 100, 200, **300**, 400, 500 |

Table 2: Parameter Settings.




(a) Number of tasks(CD) (b) Number of tasks(HK)

(c) Number of workers(CD) (d) Number of workers(HK)

Figure 7: Effect of different parameter settings on Chengdu and Haikou datasets.

**Evaluation and Comparative Analysis.** To evaluate the performance of the proposed algorithms, we conducted experiments by varying the parameters outlined in Table 2. We used the platform's total revenue as the primary evaluation metric. All experimental results are the average of 5 runs.

Fig.7 (a) and (b) illustrate the superior performance of our proposed RMO framework, with the task recommendation approach substantially outperforming alternative algorithms. Notably, as the number of tasks increases, our method employing joint optimization proves more effective than the REC method, which relies solely on task recommendation. Fig.7 (c) and (d) indicate that as the worker count rises, total revenue for all methods increases due to enhanced task completion efficiency. However, the RMO method distinguishes itself by maintaining high completion efficiency and platform revenue even with fewer workers.

## Conclusion

This paper presents a novel two-stage framework, the Recommend and Match Optimization (RMO) Framework, for dependency-aware spatial crowdsourcing. Additionally, We propose a joint optimization method that leverages meta-gradients to guide credit assignment across multi-agent policies under sparse rewards, complemented by an external hyperparameter synchronization mechanism for neural networks. This unified approach enables concurrent training of both stages through a single gradient descent process. Extensive experiments conducted on two datasets verified that our proposed method outperforms all compared approaches.

## Acknowledgments

This work is supported by the following grants: NSFC Grants 62372416, 61972362, 62036010, and 62325602; HNSF Grant 242300421215.

## References

Chen, W.; Li, W.; Liu, X.; Yang, S.; and Gao, Y. 2023. Learning explicit credit assignment for cooperative multi-agent reinforcement learning via polarization policy gradient. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, 11542–11550.

Cho, K.; Van Merriënboer, B.; Bahdanau, D.; and Bengio, Y. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Jiang, L.; Wang, S.; Guo, B.; Wang, H.; Zhang, D.; and Wang, G. 2023. FairCod: a fairness-aware concurrent dispatch system for large-scale instant delivery services. In *Proceedings of the 29th ACM SIGKDD Conference on knowledge discovery and data mining*, 4229–4238.

Kang, X.; Yu, G.; Wang, J.; Guo, W.; Domeniconi, C.; and Zhang, J. 2023. Incentive-boosted federated crowdsourcing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, 6021–6029.

Kuhn, H. W. 1955. The Hungarian method for the assignment problem. *Nav. Res. Logist.*, 2(1-2): 83–97.

Li, Y.; Li, H.; Huang, X.; Xu, J.; Han, Y.; and Xu, M. 2022a. Utility-aware dynamic ridesharing in spatial crowdsourcing. *IEEE Transactions on Mobile Computing*, 23(2): 1066–1079.

Li, Y.; Li, H.; Mei, B.; Huang, X.; Xu, J.; and Xu, M. 2023. Fairness-Guaranteed Task Assignment for Crowdsourced Mobility Services. *IEEE Transactions on Mobile Computing*.

Li, Y.; Wu, Q.; Huang, X.; Xu, J.; Gao, W.; and Xu, M. 2022b. Efficient adaptive matching for real-time city express delivery. *IEEE Transactions on Knowledge and Data Engineering*, 35(6): 5767–5779.

Lin, H.; Wu, H.; Zhang, J.; Sun, Y.; Ye, J.; and Yu, Y. 2024. Episodic Return Decomposition by Difference of Implicitly Assigned Sub-trajectory Reward. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 13808–13816.

Liu, Z.; Li, K.; Zhou, X.; Zhu, N.; Gao, Y.; and Li, K. 2022. Multi-stage complex task assignment in spatial crowdsourcing. *Information Sciences*, 586: 119–139.

Lyu, W.; Wang, H.; Song, Y.; Liu, Y.; He, T.; and Zhang, D. 2023. A Prediction-and-Scheduling Framework for Efficient Order Transfer in Logistics. In *IJCAI*, 6130–6137.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *nature*, 518(7540): 529–533.

Munkres, J. 1957. Algorithms for the Assignment and Transportation Problems. *J. Soc. Ind. App. Math.*, 5(1): 32–38.

Ni, W.; Cheng, P.; Chen, L.; and Lin, X. 2020. Task allocation in dependency-aware spatial crowdsourcing. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, 985–996.

Tong, Y.; Shi, D.; Xu, Y.; Lv, W.; Qin, Z.; and Tang, X. 2021. Combinatorial optimization meets reinforcement learning: Effective taxi order dispatching at large-scale. *IEEE Transactions on Knowledge and Data Engineering*, 35(10): 9812–9823.

Tong, Y.; Zhou, Z.; Zeng, Y.; Chen, L.; and Shahabi, C. 2020. Spatial crowdsourcing: a survey. *The VLDB Journal*, 29: 217–250.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Wang, L.; Yu, Z.; Han, Q.; Yang, D.; Pan, S.; Yao, Y.; and Zhang, D. 2020. Compact scheduling for task graph oriented mobile crowdsourcing. *IEEE Transactions on Mobile Computing*, 21(7): 2358–2371.

Wang, L.; Zhang, Y.; Hu, Y.; Wang, W.; Zhang, C.; Gao, Y.; Hao, J.; Lv, T.; and Fan, C. 2022. Individual reward assisted multi-agent reinforcement learning. In *International Conference on Machine Learning*, 23417–23432. PMLR.

Wang, Y.; Wu, J.; Hua, X.; Liu, C. H.; Li, G.; Zhao, J.; Yuan, Y.; and Wang, G. 2023. Air-ground spatial crowdsourcing with uav carriers by geometric graph convolutional multi-agent deep reinforcement learning. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*, 1790–1802.

Xu, Z.; van Hasselt, H. P.; and Silver, D. 2018. Meta-gradient reinforcement learning. *Advances in Neural Information Processing Systems*, 31.

Yao, J.; Yang, L.; and Xu, X. 2022. Online dependent task assignment in preference aware spatial crowdsourcing. *IEEE Transactions on Services Computing*, 16(4): 2827–2840.

Ye, Y.; Liu, C. H.; Dai, Z.; Zhao, J.; Yuan, Y.; Wang, G.; and Tang, J. 2023. Exploring both individuality and cooperation for air-ground spatial crowdsourcing by multi-agent deep reinforcement learning. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*, 205–217.

Yu, C.; Velu, A.; Vinitsky, E.; Gao, J.; Wang, Y.; Bayen, A.; and Wu, Y. 2022. The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in Neural Information Processing Systems*, 35: 24611–24624.

Zhang, H.; Wang, G.; Wang, X.; Zhou, Z.; Zhang, C.; Dong, Z.; and Wang, Y. 2024. NondBREM: Nondeterministic Offline Reinforcement Learning for Large-Scale Order Dispatching. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 401–409.

Zong, Z.; Zheng, M.; Li, Y.; and Jin, D. 2022. Mapdp: Cooperative multi-agent reinforcement learning to solve pickup and delivery problems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 9980–9988.