DOI: 10.19884/j.1672-5220.202505009

Capability-Aware Fair Complex Task Assignment in Spatial Crowdsourcing

FANG Xiu, ZHU Tiancheng, YI Yuqiong, HUANG Xinwei, SUN Guohao*
School of Information and Intelligent Science, Donghua University, Shanghai 201620, China

Abstract: Spatial crowdsourcing (SC) requires efficient task assignment algorithms, particularly for complex tasks requiring sequential completion. Despite extensive research on complex task assignments, most studies overlook the importance of fairness and the worker capability, leading to resource wastage. Fairness means ensuring that each worker is treated justly in the task assignment process. The worker capability refers to the maximum number of tasks that they can complete. This paper defines a capability-aware fair complex task assignment (CAFCTA) problem in SC, aiming to maximize the number of completed tasks, the total utility of workers, and capability-aware fairness under worker capability, task dependency, and time constraints. To effectively address the CAFCTA problem, we propose two algorithms: a fast assignment algorithm using a greedy strategy to match tasks to the most suitable workers quickly; an enhanced greedy algorithm with simulated annealing (SA), introducing a random search mechanism to avoid local optima. Experimental results demonstrate the effectiveness and efficiency of our algorithms.

Keywords: spatial crowdsourcing (SC); task assignment; fairness; NP-hardness; greedy strategy; simulated annealing
CLC number: TP18 **Document code**: A
Article ID: 1672-5220(2026)05-0112-09

0 Introduction

Spatial crowdsourcing (SC)[1] assigns location-related tasks to workers through mobile Internet platforms, requiring workers to complete tasks at specified locations and time. It has been widely applied in logistics distribution, event reporting, and environmental monitoring. A task assignment is a core problem in SC, where spatial and temporal constraints of both tasks and workers must be considered.

Current task assignment methods primarily apply to two categories of tasks. The first category consists of simple and mutually independent tasks[2-15], where the assignment methods focus on efficiently assigning single and discrete jobs. The second category involves the assignment of complex tasks[16-17], which typically require multi-step operational processes. These tasks demand that multiple workers complete associated tasks in a specific order. Unlike task assignments for simple tasks, complex task assignments should also consider the dependencies between tasks. For example, logistics distribution includes several dependent subtasks, such as goods preparation, route planning, and delivery. Thus, a complex task can be viewed as a set of simple subtasks with dependencies[18]. Such complex tasks occur frequently in daily life, and several methods have been proposed to efficiently assign complex tasks to workers. A multi-stage complex task assignment (MSCTA) framework was introduced to propose a greedy algorithm and a game theory algorithm for task assignments to workers[6]. Yao et al.[7] divided task assignments into two stages to obtain a solution set. Tan et al.[8] proposed a greedy algorithm and a game algorithm for task assignments to maximize the number of assigned tasks. However, these methods do not balance the efficiency and fairness among workers. Fairness[18-19] aims to avoid overburdening some workers while leaving others underutilized, and is therefore essential in task assignments.

Fairness has also been taken into account in recent task assignment schemes. For instance, in Ref. [9], it is proposed that when a worker completes a certain number of tasks, they will not be assigned new tasks in the current round. Wu et al.[10] aimed to achieve group fairness by maintaining that each group completed the same number of tasks. Zhao et al.[11] sought to average individual earnings, and minimized the difference in rewards among workers. Chen et al.[12] reduced the gap in fair costs among workers. Although these methods take such fairness into account among workers, they ignore the capability differences among workers. They assume that all workers have the same capability, which leads to unfair assignments. Worker capability denotes the maximum number of tasks that a worker can complete. Ignoring capability differences may lead to unfair assignments: high-capability workers may be underutilized, whereas low-capability workers may be overloaded.

The following example illustrates the limitations mentioned above. Figure 1 shows the location information of workers w_j and tasks t_i at a certain time, and the dependencies of tasks are $t_1 \rightarrow t_2 \rightarrow t_3 \rightarrow t_4$. The capability of w_1 is 1, and that of w_2 is 3.

Received date: 2025-05-20
* Correspondence should be addressed to SUN Guohao, email: ghsun@dhu.edu.cn
Citation: Fang X, Zhu T C, Yi Y Q, et al. Capability-aware fair complex task assignment in spatial crowdsourcing[J]. *Journal of Donghua University (English Edition)*, 2026, 43(5): 112-120.

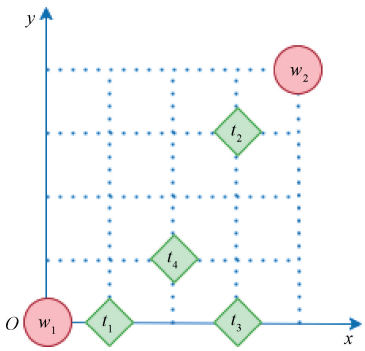


Fig. 1 A comparative example of SC allocation strategy under task dependency, fairness, and worker capability constraints

Three schemes can be derived from Fig. 1. If only the task dependency is considered, some tasks may be assigned to workers beyond their capability. If only absolute fairness is considered, downstream dependent tasks may remain incomplete. In contrast, considering both task dependency and capability-aware fairness enables all four tasks to be completed sequentially.

We formulate the problem of maximizing the number of completed tasks, the total utility of workers, capability-aware fairness under worker capability, task dependency, and time constraints in SC task assignments as a capability-aware fair complex task assignment (CAFCTA) problem. To solve this CAFCTA problem, we propose an iterative-based greedy algorithm that synchronously updates the target task set, task dependency set and assignment result set during each iteration. To avoid local optima, we further propose a simulated annealing (SA)-based greedy algorithm. This algorithm uses probabilistic search to avoid local optima and to improve the assignment result.

The main contributions are as follows.

1) We formulate the CAFCTA problem and prove its offline version is NP-hard.

2) We propose an iterative-based greedy algorithm and an SA-based greedy algorithm, and analyze their complexities.

3) We conduct experiments on two real-world datasets to validate their effectiveness and efficiency.

1 Related Work

This section reviews SC task assignments from three aspects: general task assignments, complex task assignments, and fairness-aware task assignments.

1.1 General task assignment methods in SC

For task assignments in SC[20-25], especially for simple tasks, several studies have proposed methods that address specific aspects of the assignment process. Mavridis et al.[3] studied skill-based task assignments by jointly considering worker skills and distance. Difallah et al.[4] focused on task scheduling to reduce latency and improve worker productivity. Celis et al.[5] studied privacy-aware task assignments under PUSH, PULL, and TUG-of-WAR settings. However, these methods do not consider the issue of dependencies between tasks.

1.2 Complex task assignment methods in SC

Recent studies have proposed various efficient algorithms for dependency-aware complex task assignments. Liu et al.[6] proposed the MSCTA framework and designed greedy and game-theoretic algorithms. Yao et al.[7] adopted a two-stage assignment strategy with greedy matching and the Hungarian algorithm. Tan et al.[8] considered the worker skill quality and task budget, and proposed pruning strategies to reduce ineffective computation. However, these methods mainly optimize the assignment efficiency and do not fully address fairness among workers.

1.3 Fairness-aware task assignment methods in SC

Zhang et al.[9] limited further assignments to workers who had completed too many tasks, aiming to balance workloads. Wu et al.[10] studied group fairness by balancing assignments among non-overlapping worker groups. References [11–12] focused on reducing earning differences among workers while maintaining the assignment efficiency. Although these studies consider fairness, they ignore worker capability and mainly focus on simple independent tasks. Therefore, they are not suitable for capability-aware complex task assignments with dependencies.

2 Problem Statement

In this section, we propose the CAFCTA problem in SC. Table 1 lists the commonly used notations in this paper.

Table 1 Symbol definitions

Symbol	Meaning
$T=\{t_i\}_{i=1}^{\lvert T\rvert}$	Set of spatial tasks, with $\lvert T\rvert$ denoting size of T
$W=\{w_j\}_{j=1}^{\lvert W\rvert}$	Set of crowd workers, with $\lvert W\rvert$ denoting size of W
$\Omega=\{(t_i,w_j)\}_{i=1,j=1}^{\lvert\Omega\rvert}$	Solution set of assigned task-worker pair, with $\lvert\Omega\rvert$ denoting size of Ω
D	Set of dependencies between tasks
T'	Set of executable tasks
W_i'	Set of possible workers for t_i
ρ	Distance limit parameter set by SC platform
$u(t_i, w_j)$	Utility that w_j can get by completing t_i
α	Consumption per unit distance
β	Weight parameter used to balance capability fairness index
γ	Cooling rate in SA
$I_{\min}$	Minimum temperature in SA
I_0	Initial temperature in SA
I_{cur}	Current temperature in SA
E	Energy in SA
ε	Weight parameter used to balance energy in SA

Definition 1 Spatial task

The ith spatial task (task for short) is represented as a tuple $t_i = \langle h_i, loc_i \rangle$, where h_i represents the time at which the ith task appears, and loc_i represents the location of the ith task.

Definition 2 Crowd worker

The jth crowd worker (worker for short) is represented as a tuple $w_j = \langle h_j, loc_j, cnt_j, cap_j \rangle$, where h_j, loc_j, cnt_j, and cap_j represent the appearance time, location, number of completed tasks, and capability of w_j, respectively.

Without loss of generality, we assume that all workers have a constant movement speed, allowing the task time constraints to be represented by distance.

Definition 3 Distance between task and worker

Formally, the Euclidean distance $d(t_i, w_j)$ between t_i and w_j can be defined as

$$d(t_i, w_j) = \sqrt{(loc_{i,x} - loc_{j,x})^2 + (loc_{i,y} - loc_{j,y})^2}, \quad (1)$$

where $loc_{i,x}$ and $loc_{j,x}$ mean the x-coordinates of t_i and w_j, respectively; $loc_{i,y}$ and $loc_{j,y}$ mean the y-coordinates of t_i and w_j, respectively.

Definition 4 Task-worker pair

Task-worker pair (t_i, w_j) refers to a feasible match between t_i and w_j. A feasible match needs to meet the following constraints: $h_i == h_j$ meaning that t_i and w_j appear at the same time; $cnt_j < cap_j$ meaning that worker w_j still has the capability to complete a task; $d(t_i, w_j) < \rho$ meaning that $d(t_i, w_j)$ should be below a threshold ρ set by the SC platform.

Definition 5 Dependency set

Dependency set D is a dictionary where each key is t_i. The value associated with each key is a task list that t_i depends on. Formally, D can be represented as

$$D = \{t_i : \{\,|t_1, t_2, \cdots, t_{i-1}|\,\}\}. \quad (2)$$

D maps each task to its preceding tasks, ensuring the required execution order.

Definition 6 Utility

The utility of (t_i, w_j), namely $u(t_i, w_j)$, refers to the utility that w_j can get by completing t_i, defined as

$$u(t_i, w_j) = a_{t_i} - d(t_i, w_j) \cdot \alpha, \quad (3)$$

where a_{t_i} is the initial utility of task t_i.

Let Ω be the solution set of all assigned task-worker pairs. Then, the total utility *Utility_Sum* of workers can be defined as

$$Utility_Sum = \sum_{(t_i, w_j) \in \Omega} u(t_i, w_j). \quad (4)$$

Utility_Sum measures the assignment quality based on the task utility, worker-task distance, and completed assignments.

To avoid overloading or underutilizing workers, fairness is measured according to whether assigned task numbers match worker capabilities.

Definition 7 Capability fairness index

The capability fairness index *CFI* is a metric that measures the capability-aware fairness of a task assignment solution. Based on our analysis, capability-aware fairness depends on the data distribution of task assignments and the correlation between the number of assigned tasks and the worker capacity.

$$CFI = \beta \cdot \exp\left(-\frac{\sigma_\Omega^2}{\mu_\Omega}\right) + (1 - \beta) \cdot r, \quad (5)$$

where σ_Ω^2 is the variance of the number of completed tasks; μ_Ω is the mean of the number of completed tasks; r is the Pearson correlation coefficient between the *cap* matrix (which stores the capability of each worker) and the *cnt* matrix (which stores the count of tasks each worker has completed).

CFI is non-negative, continuous and scale-free, and bounded within $[0, 1]$. A higher *CFI* indicates better capability-aware fairness.

Definition 8 CAFCTA

Given a set of tasks T and a set of workers W, the CAFCTA problem is to find a solution set Ω that maximizes the number of tasks completed, *Utility_Sum*, and *CFI* under the worker capability, task dependency, and time constraints:

$$\begin{aligned} &\max\{\,|\Omega|, Utility_Sum, CFI\}, \\ &\text{subject to} \quad cnt_j < cap_j, \\ &\qquad\qquad\quad D, \\ &\qquad\qquad\quad d(t_i, w_j) < \rho. \end{aligned} \quad (6)$$

After formally defining the CAFCTA problem, we prove that its offline version is NP-hard.

Lemma 1 The CAFCTA problem is NP-hard.

Proof In the offline CAFCTA problem, all task and worker information is known in advance. For any instance of the traveling salesman problem (TSP), we can construct a related instance of the CAFCTA problem. By setting the number of workers to 1 and α to 0, the first part of the *CFI* becomes $\beta \cdot \exp(0) = \beta$. When there is only one worker, r is meaningless, so we only need to consider maximizing the number of completed tasks and *Utility_Sum*, which means maximizing the utilities in T. Solving the TSP to maximize the *Utility_Sum* in T is equivalent to solving the related instance of the CAFCTA problem to maximize the number of completed tasks, *Utility_Sum* and *CFI*. Since the TSP is NP-hard, the offline version of the CAFCTA problem is also NP-hard.

3 Proposed Algorithms

This section presents an iterative-based greedy algorithm and an SA-based greedy algorithm for CAFCTA.

3.1 Iterative-based greedy algorithm

Since the CAFCTA problem is NP-hard, we design an iterative-based greedy algorithm to obtain feasible

approximate solutions efficiently. The core idea of this greedy algorithm is to make locally optimal choices at each decision step to build the final solution. Specifically, for every task without prerequisite tasks, the iterative-based greedy algorithm selects the worker who is closest to the task location, achieves the maximum utility and the highest *CFI* under the worker capability, task dependency, and time constraints. The pseudocode is shown in Algorithm 1.

Algorithm 1 Iterative-based greedy algorithm

Input: Task set $T = \{t_i\}_{i=1}^{|T|}$, $t_i = \langle h_i, loc_i \rangle$; worker set $W = \{w_j\}_{j=1}^{|W|}$, $w_j = \langle h_j, loc_j, cnt_j, cap_j \rangle$; dependency set D; distance limit parameter ρ

Output: A task-worker pair solution set Ω

```
Initialize T' ← ∅, Ω ← ∅
Determine whether t_i in T can be added to T' based on D
while T' is not empty do
    Get a task t_* from T'
    Initialize W'_* ← ∅
    for all w_j ∈ W do
        if d(t_*, w_j) < ρ and cnt_j < cap_j then
            W'_* ← W'_* ∪ w_j
            Calculate u(t_*, w_j)
        end if
    end for
    Find w_* in W'_* with the highest u(t_*, w_*)
    Ω ← Ω ∪ {t_*, w_*}
    T' ← T'/{t_*}
    D ← D/{t_*}
    Check if t_i in T can be added to T' based on D
end while
return Ω
```

Algorithm 1 first identifies executable tasks whose prerequisites have been satisfied. For each executable task, it constructs the feasible worker set under distance, capability, and status constraints, and then selects the worker that maximizes the utility while maintaining capability-aware fairness. After assignments, the dependency set and executable task set are updated until no executable task remains.

Feasible workers are selected under three constraints: distance limit, capability limit, and current status.

Algorithm 1 is correct because it only assigns executable tasks whose dependencies have been satisfied, and only selects workers satisfying distance, capability, and status constraints. After each assignment, newly executable tasks are added according to D. Therefore, the final solution satisfies task dependency and worker capability constraints. For each task, Algorithm 1 scans all workers to construct the feasible worker set and select the best worker. Thus, its time complexity is $O(|T||W|)$.

3.2 SA-based greedy algorithm

Algorithm 1 may fall into local optima because of its greedy strategy. Therefore, we introduce simulated annealing to accept worse solutions with a certain probability, improving global exploration at the cost of additional time. The pseudocode is shown in Algorithm 2.

The parameters required for Algorithm 2 include I_0, $I_{\min}$, γ, and the maximum number of iterations N. These additional parameters are used in the SA part of Algorithm 2. The change in energy ΔE quantifies the difference in the objective function value between the current solution and a neighboring solution. Specifically, we first calculate $d(t_i, w_j)$. Then, we calculate the difference between cap_j and cnt_j of w_j, which we define as cap_j^*:

$$cap_j^* = cap_j - cnt_j. \tag{7}$$

After defining $d(t_i, w_j)$ and cap_j^*, we propose

$$E = \varepsilon d(t_i, w_j) - (1 - \varepsilon) \cdot cap_j^*. \tag{8}$$

A lower energy value indicates a better assignment, with a shorter travel distance and better use of worker capability. ΔE is calculated as the difference between the energy E_{new} of the new solution Ω' and the energy E_{cur} of the current solution Ω:

$$\Delta E = E_{\text{new}} - E_{\text{cur}}. \tag{9}$$

After defining ΔE, we introduce the conditions under which the new solution replaces the current solution in the SA-based greedy algorithm. The acceptance of a new solution is based on ΔE and I_{cur}. Specifically, if $\Delta E \leqslant 0$, the new solution is accepted unconditionally. This is because the new solution provides a lower energy value, thus improving the overall solution quality. However, when $\Delta E > 0$, a probability function determines whether to accept the new solution. The probability function p is

$$p = \exp\left(-\frac{\Delta E}{I_{\text{cur}}}\right). \tag{10}$$

This probability function ensures that even when the new solution is not as good as the current solution, there is still a certain probability of accepting it. This mechanism helps Algorithm 2 to avoid getting trapped in a local optimum and explore a broad solution space.

At the beginning of Algorithm 2, we generate a solution set Ω using Algorithm 1, as described above. Then, we start the SA loop. In each iteration, we first randomly select a task-worker pair (t_*, w_*) from Ω, and find a set of matching workers W'_* for the task t_*. If W'_* is empty, we proceed to the next iteration; otherwise, we randomly select a worker w'_* from W'_*. Next, we calculate ΔE. According to the SA concept, if $\Delta E \leqslant 0$, we accept the new pair (t_*, w'_*) directly. If $\Delta E > 0$, we adopt the new pair (t_*, w'_*) with a certain probability. After adopting the new pair, we update the information for both

w_* and w'_*. Finally, we update I_{cur} and k. When I_{min} or N is reached, the while loop exits; otherwise, it continues. Through this process, the SA-based greedy algorithm balances exploration and exploitation to improve task completion, utility, and capability-aware fairness.

Algorithm 2 SA-based greedy algorithm

Input: Task set $T = \{t_i\}_{i=1}^{|T|}$, $t_i = \langle h_i, loc_i \rangle$; worker set $W = \{w_j\}_{j=1}^{|W|}$, $w_j = \langle h_j, loc_j, cnt_j, cap_j \rangle$; dependency set D; distance limit parameter ρ; initial temperature I_0; minimum temperature I_{min}; cooling rate γ; maximum number of iterations N

Output: A task-worker pair solution set Ω

Initialize the solution set Ω using Algorithm 1
Set current temperature $I_{cur} \leftarrow I_0$
Initialize iteration counter $k \leftarrow 0$
while $I_{cur} > I_{min}$ or $k < N$ **do**
  Randomly select a (t_*, w_*) pair from Ω
  Initialize possible worker set $W'_* \leftarrow \varnothing$
  for all $w_j \in W$ **do**
    if $d(t_*, w_j) < \rho$ and $cnt_j < cap_j$ **then**
      $W'_* \leftarrow W'_* \cup w_j$
      Calculate $u(t_*, w_j)$
    end if
  end for
  if $W'_* == \varnothing$ **then**
    Update the temperature: $I_{cur} \leftarrow I_{cur} \cdot \gamma$
    Increment iteration counter: $k \leftarrow k + 1$
    Continue
  end if
  Randomly select a worker w'_* from W'_*
  Calculate energy change ΔE and p
  Generate a random number $\xi \sim U(0, 1)$
  if $\Delta E \leqslant 0$ or $\xi < p$ **then**
    $\Omega \leftarrow \Omega / (t_*, w_*)$
    $\Omega \leftarrow \Omega \cup (t_*, w'_*)$
    Update the information of w_* and w'_*
  end if
  Update the temperature: $I_{cur} \leftarrow I_{cur} \cdot \gamma$
  Increment iteration counter: $k \leftarrow k + 1$
end while
return Ω

Algorithm 2 is correct since it starts from a feasible solution produced by Algorithm 1 and only replaces workers with feasible alternatives. Since the task set and dependency-respecting order remain unchanged, each iteration preserves feasibility. Thus, the final Ω satisfies all constraints.

According to the analysis in Algorithm 1, we know that the time complexity of obtaining the initial solution set Ω is $O(|T||W|)$. Next, we analyze the time complexity of the added content. In the while loop, the time complexity of finding the set of possible workers W'_* for task t_* is $O(|W|)$. Other operations have a constant time complexity of $O(1)$. The while loop has two conditions: $I_{cur} > I_{min}$ and $k < N$. Therefore, the while loop executes at most $\min(\log_{I_{min}/I_0}\gamma, N)$ times. Consequently, the overall time complexity is $O(|T||W| + \min(\log_{I_{min}/I_0}\gamma, N)|W|)$.

4 Experimental Study

This section evaluates the proposed algorithms on two real-world datasets. The experiments are implemented in Python 3.9.5 on an Intel Core i7-10750H CPU with 16 GB RAM, and each result is averaged over 50 independent runs.

4.1 Experimental setup

In this study, two real-world datasets were employed to validate the proposed algorithms: the gMission dataset[2] and the Didi Chuxing dataset from Chengdu, China (Didi for short). gMission[2] contains over 7 000 tasks and 500 workers. Didi contains about 3 000 tasks and 500 workers per day. Both datasets provide location information for tasks and workers.

For the above two datasets, we normalize the location information of each task and worker, mapping them into a $[0, 1]^2$ space. This setting is consistent with the experimental setup in Ref. [6]. Since the worker capability is unavailable in both datasets, we estimate it by repeatedly running MSCTA_Greedy[6] without capability constraints and using the rounded average number of completed tasks as the capability of each worker. After generating cap_j for each worker, we turn our attention to task dependency. For gMission, task timestamps are generated using a normal distribution, and dependencies are constructed for every 10 tasks according to the time order, following Ref. [6].

We compare our algorithms with two baselines: Random and MSCTA_Greedy[6]. The Random algorithm randomly selects feasible task-worker pairs until no feasible pair remains. The MSCTA_Greedy algorithm[6] greedily assigns executable tasks to workers according to platform utility and updates dependencies iteratively.

In our proposed SA-based greedy algorithm, we set I_0 to 100, I_{min} to 0.01, N to 1 000, and γ to 0.95.

To evaluate the performance of our algorithms, we employ a set of metrics that cover both efficiency and fairness aspects, as follows.

1) The number of completed tasks: it is simply the count of tasks that are successfully assigned to workers and completed. This metric can be easily calculated as $\sum_{w_j \in W} cnt_j$, and provides insight into the effectiveness of the algorithms.

2) Execution time: this metric measures the computational efficiency of the algorithms. It is defined as the total duration from the start of the algorithms until

all tasks have been assigned. Shorter execution time indicates more efficient algorithms, which is crucial for SC platforms.

3) *Utility_Sum*: this metric quantifies the aggregate benefit derived by all workers, which can be calculated based on Eq. (4). A higher *Utility_Sum* suggests a better assignment of tasks.

4) *CFI*: *CFI* evaluates the equity of task assignments among workers. It is computed based on Eq. (5); a *CFI* closer to 1 indicates a better capability-aware fairness distribution, meaning that tasks are evenly assigned without overburdening any particular worker.

By utilizing these evaluation metrics, we aim to provide a comprehensive assessment of our proposed algorithms.

4.2 Experimental results

4.2.1 Effect of initial task utility

We first study the effect of initial task utility. Results on Didi and gMission are shown in Figs. 2 and 3.

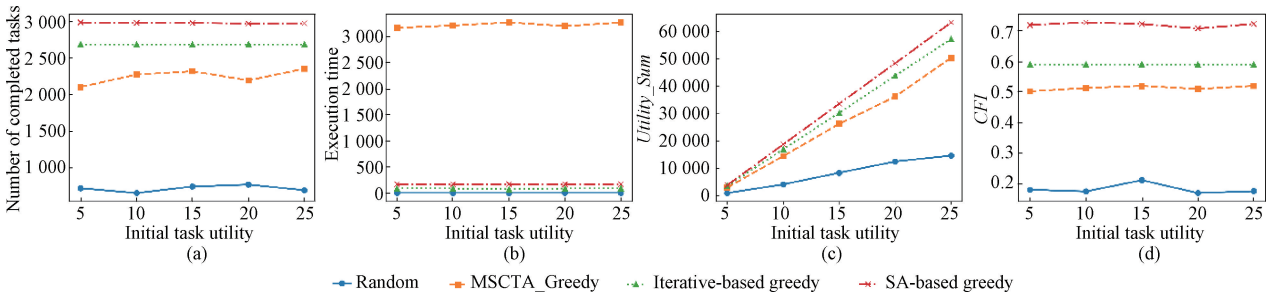


Fig. 2 Effect of initial task utility on Didi: (a) number of completed tasks; (b) execution time; (c) *Utility_Sum*; (d) *CFI*

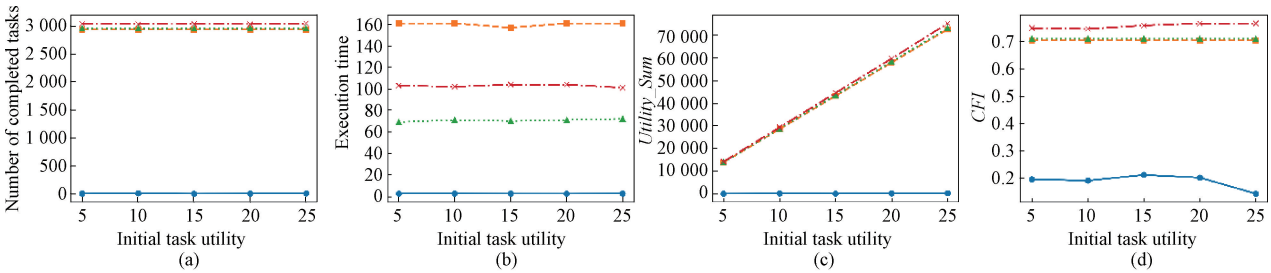


Fig. 3 Effect of initial task utility on gMission: (a) number of completed tasks; (b) execution time; (c) *Utility_Sum*; (d) *CFI*

As shown in Figs. 2(a) and 3(a), the Random algorithm completes the fewest tasks, the MSCTA_Greedy algorithm achieves moderate performance, and the proposed algorithms perform better. The SA-based greedy algorithm achieves the highest number of completed tasks. Figures 2(b) and 3(b) show that the execution time remains stable under different initial task utilities. The MSCTA_Greedy algorithm takes the longest time, followed by SA-based greedy and iterative-based greedy algorithms, while the Random algorithm is the fastest. Figures 2(c) and 3(c) show that *Utility_Sum* increases with the initial task utility. The SA-based greedy algorithm achieves the highest *Utility_Sum*, followed by the iterative-based greedy, MSCTA_Greedy, and Random algorithms. Figures 2(d) and 3(d) indicate that the proposed algorithms achieve a higher *CFI* than the baselines, with the SA-based greedy algorithm performing the best.

Overall, the proposed algorithms improve task completion, *Utility_Sum*, and *CFI* while maintaining acceptable execution time.

4.2.2 Effect of distance limit parameter

The effect of the distance limit parameter ρ, which determines the maximum acceptable task-worker distance, is evaluated. ρ varies from 0.37 to 0.41 on Didi and from 0.04 to 0.08 on gMission. Results are shown in Figs. 4 and 5.

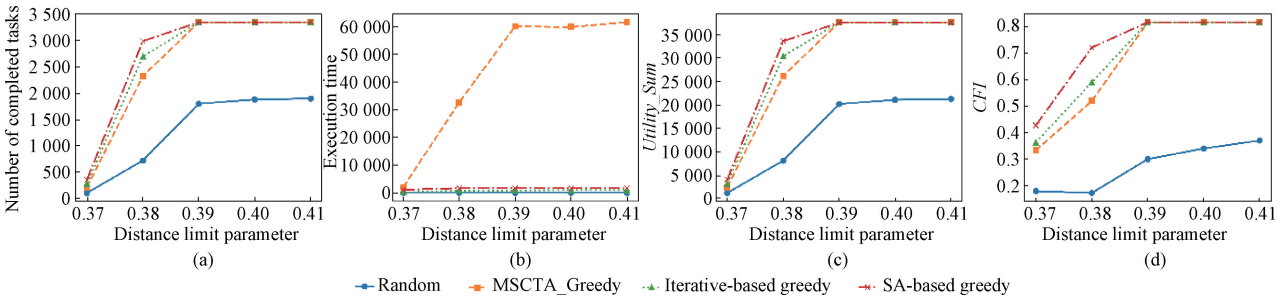


Fig. 4 Effect of distance limit parameter on Didi: (a) number of completed tasks; (b) execution time; (c) *Utility_Sum*; (d) *CFI*

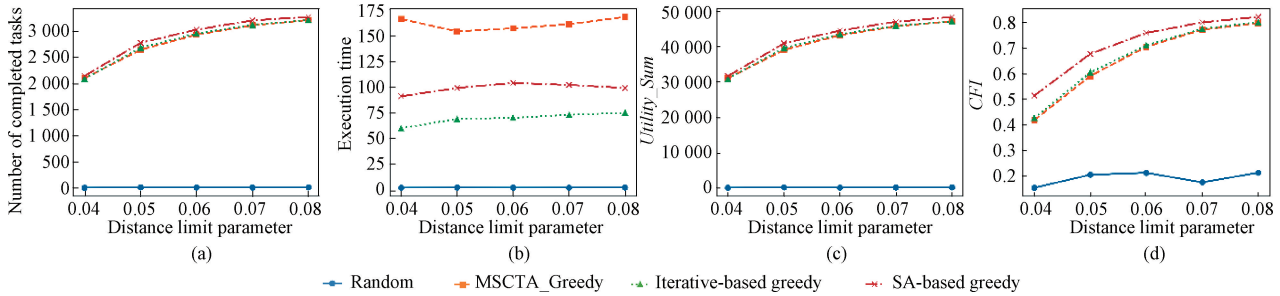


Fig. 5 Effect of distance limit parameter on gMission: (a) number of completed tasks; (b) execution time; (c) *Utility_Sum*; (d) *CFI*

Figures 4(a) and 5(a) show that the number of completed tasks increases with ρ and then gradually saturates, since more workers become available until worker capabilities are nearly fully used. Figures 4(b) and 5(b) show that the MSCTA_Greedy algorithm has the largest execution time, while the Random algorithm is the fastest. The SA-based greedy algorithm requires more time than the iterative-based greedy algorithm because of the additional annealing process. Figures 4(c) and 5(c) show that *Utility_Sum* follows the same trend as the number of completed tasks. The SA-based greedy algorithm achieves the highest *Utility_Sum*. Figures 4(d) and 5(d) show that *CFI* increases with ρ and then stabilizes. The SA-based greedy algorithm maintains the best fairness performance.

In summary, appropriately relaxing ρ can effectively enhance performance and benefits while solving the CAFCTA problem. However, beyond a certain threshold, the improvement effects are no longer significant. Our proposed SA-based greedy algorithm performs the best, followed by the iterative-based greedy algorithm. The MSCTA_Greedy algorithm, due to its higher computational complexity, leads to greater time costs and does not perform as well as our proposed algorithms.

4.2.3 Effect of cooling rate

We evaluate the effect of the cooling rate γ in the SA-based greedy algorithm by setting the initial temperature to 100. Results are shown in Fig. 6. γ varies from 0.91 to 0.99 in the two datasets.

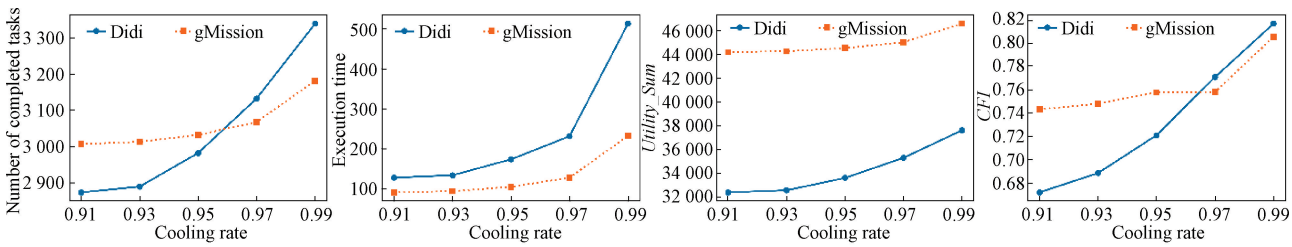


Fig. 6 Effect of cooling rate in SA-based greedy algorithm: (a) number of completed tasks; (b) execution time; (c) *Utility_Sum*; (d) *CFI*

From Fig. 6(a), it can be observed that the number of completed tasks shows an upward trend on both datasets, particularly with a more significant increase on Didi. This is because Didi has a more complex task distribution, which benefits more from the enhanced exploration that comes with a higher cooling rate. In contrast, gMission has a simpler structure, leading to less significant improvements. In Fig. 6(b), it can also be observed that as γ increases, the execution time of the SA-based greedy algorithm rises accordingly. This is because as γ increases, the SA-based greedy algorithm requires a larger number of iterations in the SA loop, thereby increasing the computation time. In Fig. 6(c), *Utility_Sum* also increases as γ grows. Similarly, *CFI* exhibits a slow upward trend in Fig. 6(d). This indicates that, within the proposed SA-based greedy algorithm, an

increase in the number of SA loop iterations can effectively enhance its performance at the cost of increased execution time.

Therefore, a smaller γ is preferable for time-sensitive scenarios, whereas a higher γ is preferable when higher solution quality is desired.

4.2.4 Effect of initial temperature

We further study the effect of the initial temperature I_0 by fixing $\gamma = 0.95$. Results are shown in Fig. 7. As shown in Fig. 7(a), the number of completed tasks shows an upward trend in both datasets. Initially, the growth in the number of completed tasks is significant, but gradually slows down. As I_0 increases, the execution time rises in both datasets (Fig. 7(b)). This is because a higher initial temperature leads to an increased number of iterations in the SA loop, thereby increasing the computational time. From Fig. 7(c), the trend of *Utility_Sum* mirrors that of the number of completed tasks. This suggests that increasing I_0 gradually enhances *Utility_Sum*. As shown in Fig. 7(d), *CFI* improves as I_0 increases. This indicates that a higher initial temperature enables the SA-based greedy algorithm to assign tasks more evenly, thus improving overall fairness. Thus, I_0 should be selected by balancing the solution quality and execution time.

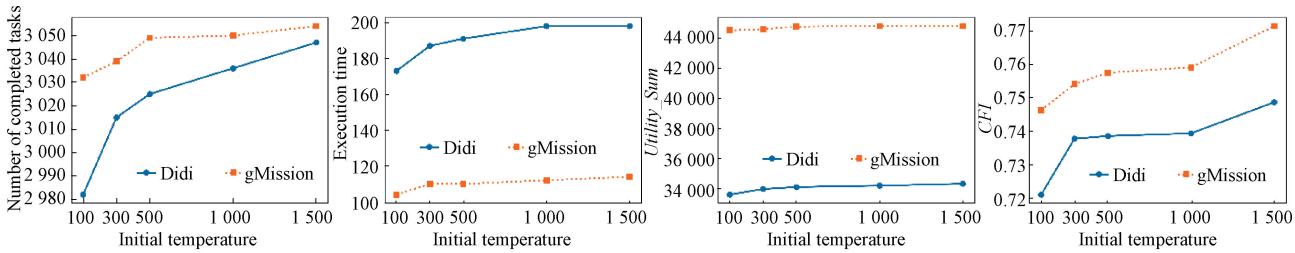


Fig. 7 Effect of initial temperature in SA-based greedy algorithm: (a) number of completed tasks; (b) execution time; (c) *Utility_Sum*; (d) *CFI*

5 Conclusions

This paper studies the CAFCTA problem in SC, aiming to maximize the number of completed tasks, total utility of workers, and capability fairness index under worker capability, task dependency and time constraints. We prove that the offline CAFCTA problem is NP-hard and propose an iterative-based greedy algorithm and an SA-based greedy algorithm. Experiments on two real-world datasets validate the effectiveness and efficiency of the proposed algorithms. Future work will study assignment stability and responsiveness in dynamic environments.

References

[1] Guo B, Liu Y, Wang L Y, et al. Task allocation in spatial crowdsourcing: current state and future directions [J]. *IEEE Internet of Things Journal*, 2018, 5(3): 1749-1764.

[2] Chen Z, Fu R, Zhao Z Y, et al. gMission: a general spatial crowdsourcing platform [J]. *Proceedings of the VLDB Endowment*, 2014, 7(13): 1629-1632.

[3] Mavridis P, Gross-Amblard D, Miklós Z. Using hierarchical skills for optimized task assignment in knowledge-intensive crowdsourcing [C]// Proceedings of the 25th International Conference on World Wide Web. New York: ACM, 2016: 843-853.

[4] Difallah D E, Demartini G, Cudré-Mauroux P. Scheduling human intelligence tasks in multi-tenant crowd-powered systems [C]//Proceedings of the 25th International Conference on World Wide Web. New York: ACM, 2016: 855-865.

[5] Celis L E, Reddy S P, Singh I P, et al. Assignment techniques for crowdsourcing sensitive tasks [C]//Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing. New York: ACM, 2016: 836-847.

[6] Liu Z, Li K L, Zhou X, et al. Multi-stage complex task assignment in spatial crowdsourcing [J]. *Information Sciences: an International Journal*, 2022, 586(C): 119-139.

[7] Yao J J, Yang L, Xu X H. Online dependent task assignment in preference aware spatial crowdsourcing [J]. *IEEE Transactions on Services Computing*, 2023, 16(4): 2827-2840.

[8] Tan W A, Liang Z J, Liu J, et al. Dependency-based task assignment in spatial crowdsourcing [C]//Computer Supported Cooperative Work and Social Computing. Singapore: Springer, 2023: 48-61.

[9] Zhang J L, Jiang T X, Gao X F, et al. An online fairness-aware task planning approach for spatial crowdsourcing [J]. *IEEE Transactions on Mobile Computing*, 2024, 23(1): 150-163.

[10] Wu B W, Han K, Zhang E P. On the task assignment with group fairness for spatial crowdsourcing [J]. *Information Processing & Management*, 2023, 60(2): 103175.

[11] Zhao Y, Zheng K, Guo J N, et al. Fairness-aware task assignment in spatial crowdsourcing: game-theoretic approaches [C]//2021 IEEE 37th

International Conference on Data Engineering (ICDE). Piscataway: IEEE, 2021: 265-276.
[12] Chen Z, Cheng P, Chen L, et al. Fair task assignment in spatial crowdsourcing [J]. *Proceedings of the VLDB Endowment*, 2020, 13(12): 2479-2492.
[13] Tong Y X, Zhou Z M, Zeng Y X, et al. Spatial crowdsourcing: a survey [J]. *The VLDB Journal*, 2020, 29(1): 217-250.
[14] Li M C, Zheng Y Y, Jin X, et al. Task assignment for simple tasks with small budget in mobile crowdsourcing [C]//2018 14th International Conference on Mobile Ad-Hoc and Sensor Networks (MSN). Piscataway: IEEE, 2019: 68-73.
[15] Blake B A. Assignment of independent tasks to minimize completion time [J]. *Software: Practice and Experience*, 1992, 22(9): 723-734.
[16] Dang H, Nguyen T, To H. Maximum complex task assignment: towards tasks correlation in spatial crowdsourcing [C]//Proceedings of International Conference on Information Integration and Web-based Applications & Services. New York: ACM, 2013: 77-81.
[17] Bhatti S S, Gao X F, Chen G H. General framework, opportunities and challenges for crowdsourcing techniques: a comprehensive survey [J]. *Journal of Systems and Software*, 2020, 167: 110611.
[18] Jain R, Chiu D, Hawe W. A quantitative measure of fairness and discrimination for resource allocation in shared computer systems [PP/OL]. arXiv (1998-09-24) [2024-10-28]. https://arxiv.org/abs/cs/9809099.
[19] Liu Q, Abdessalem T, Wu H Y, et al. Cost minimization and social fairness for spatial crowdsourcing tasks [C]//Database Systems for Advanced Applications. Cham: Springer, 2016: 3-17.
[20] Song T S, Xu K, Li J N, et al. Multi-skill aware task assignment in real-time spatial crowdsourcing [J]. *GeoInformatica*, 2020, 24(1): 153-173.
[21] Cheng P, Lian X, Chen L, et al. Task assignment on multi-skill oriented spatial crowdsourcing [J]. *IEEE Transactions on Knowledge and Data Engineering*, 2016, 28(8): 2201-2215.
[22] Ni W Z, Cheng P, Chen L, et al. Task allocation in dependency-aware spatial crowdsourcing [C]//2020 IEEE 36th International Conference on Data Engineering (ICDE). Piscataway: IEEE, 2020: 985-996.
[23] Tong Y X, She J Y, Ding B L, et al. Online mobile micro-task allocation in spatial crowdsourcing [C]//2016 IEEE 32nd International Conference on Data Engineering (ICDE). Piscataway: IEEE, 2016: 49-60.
[24] Tong Y X, Wang L B, Zhou Z M, et al. Dynamic pricing in spatial crowdsourcing: a matching-based approach [C]//Proceedings of the 2018 International Conference on Management of Data. New York: ACM, 2018: 773-788.
[25] Tong Y X, Wang L B, Zhou Z M, et al. Flexible online task assignment in real-time spatial data [J]. *Proceedings of the VLDB Endowment*, 2017, 10(11): 1334-1345.

空间众包中能力感知公平复杂任务分配

方 秀, 朱天成, 易宇琼, 黄馨苇, 孙国豪*
东华大学 信息与智能科学学院, 上海 201620

摘 要: 空间众包需要高效的任务分配算法, 尤其是对于需要按顺序完成的复杂任务。尽管已有大量关于复杂任务分配的研究, 但大多数工作忽视了公平性和工人能力的重要性, 导致资源浪费。公平性是指确保每位工人在任务分配过程中得到公正对待。工人能力则指他们能够完成的任务数量上限。本文定义了空间众包中能力感知公平复杂任务分配(capability-aware fair complex task assignment, CAFCTA)问题, 目标是在工人能力、任务依赖关系和时间约束下, 最大化已完成任务数量、工人总效用及能力感知公平性。为有效解决 CAFCTA 问题, 我们提出了两种算法: 一种基于贪心策略的快速分配算法, 用于将任务快速匹配给最合适的工人; 另一种是结合模拟退火的增强贪心算法, 引入随机搜索机制以避免陷入局部最优。实验结果表明了所提算法的有效性和高效性。

关键词: 空间众包; 任务分配; 公平性; NP 难; 贪心策略; 模拟退火