Spatial Task Allocation Mechanisms in Mobile Crowdsensing (Postprint)

Authors: Xing Qian, Sun Xuemei, Zhang Lishen

Date: 2019-01-03T00:00:00+00:00

Abstract

In mobile crowdsensing, the spatial distance between users and tasks in spatial task allocation problems directly affects the cost required for task completion, yet this aspect has been insufficiently considered in existing research. Therefore, this paper designs a spatial task allocation mechanism for mobile crowdsensing with the objective of minimizing sensing costs. First, an efficient task allocation method is proposed based on genetic algorithms and greedy algorithms to minimize sensing costs. Second, to address the stochastic nature of user sensing quality, an update mechanism for user sensing quality is designed based on users' historical sensing performance and current task execution status. To validate the effectiveness of the proposed mechanism, simulation experiments are conducted comparing it with two baseline task allocation methods. Experimental results demonstrate that the proposed mechanism achieves superior performance in terms of total sensing cost and total distance traveled by users during task execution, thereby indicating promising application prospects for this spatial task allocation mechanism.

Full Text

Assignment Mechanism for Spatial Tasks in Mobile Crowd Sensing

Xing Qian, Sun Xuemei, Yuan Chunmiao
School of Computer Science & Software Engineering, Tianjin Polytechnic University, Tianjin 300387, China

Abstract

In mobile crowd sensing (MCS), the spatial distance between users and tasks directly affects the cost required to complete tasks, yet existing research has

inadequately addressed this consideration. To minimize sensing costs, this paper proposes a spatial task allocation mechanism for MCS. First, we design an efficient task allocation method based on a hybrid genetic and greedy algorithm with the objective of minimizing sensing costs. Second, to address the randomness of user sensing quality, we develop a user sensing quality update mechanism that incorporates both historical performance and current task execution results. We validate the proposed mechanism through simulation experiments comparing it against two benchmark allocation methods. Experimental results demonstrate that our mechanism achieves superior performance in terms of total sensing cost and total distance traveled by users to execute tasks, indicating strong applicability for spatial task allocation in MCS.

Keywords: mobile crowd sensing; spatial tasks; task allocation; cost minimization

0 Introduction

Smartphones have become mainstream mobile communication devices, typically equipped with powerful sensors such as GPS, cameras, and digital compasses. This has enabled mobile crowd sensing (MCS)—an emerging paradigm for data collection utilizing smartphones—to gain significant attention in academia. MCS systems generally comprise three components: task owners, cloud platforms, and sensing users. Task owners submit sensing tasks to the cloud platform, which then intelligently allocates these tasks to sensing users through sophisticated assignment algorithms. The users subsequently complete the assigned tasks and upload the collected data. Selecting appropriate sensing users represents a critical challenge in MCS, as rational task allocation schemes are fundamental to efficient task execution.

Most current MCS applications involve spatial tasks, including traffic planning, environmental monitoring, and public safety, which impose more stringent requirements on task allocation methods. While numerous studies have addressed task allocation in MCS, the field remains in its early stages, with many methods and theories requiring further refinement. For instance, some researchers have designed task allocation frameworks that account for uncertainty in data quality, while others have addressed continuous time-window-based allocation problems using dynamic programming or formulated task allocation as integer programming problems solved via greedy techniques. However, these approaches are unsuitable for spatial task allocation. Recent work has modeled spatial task allocation as a multi-constrained knapsack problem or proposed multi-task allocation frameworks incorporating queuing, reputation, and delegation mechanisms. Nevertheless, these methods optimize data utility rather than considering how spatial distance between users and tasks impacts sensing costs. Another approach focuses on minimizing total travel distance and the number of users for multi-task scenarios but neglects task quality requirements for sensing data.

Building upon existing task allocation methods, this paper addresses spatial task allocation by designing a mechanism that minimizes sensing costs. Through simulation experiments, we compare our proposed GGA-TA allocation algorithm against two benchmark methods and analyze the results comprehensively.

---

1 Problem Model

Consider a cloud platform where multiple task publishers have posted N sensing tasks at different locations, represented by the set $T = \{1, 2, \dots, N\}$. Let $l_i = (l_{i,1}, l_{i,2})$ denote the spatial location of task i, where $l_{i,1}$ and $l_{i,2}$ represent the x and y coordinates in 2D space. Let $U = \{1, 2, \dots, M\}$ represent the set of users.

In MCS, sensing tasks impose not only location constraints but also quality requirements for the collected data. Task owners typically require that the quality of sensing data exceeds a certain threshold. We define data quality as the total amount of qualified data—data that simultaneously satisfies clarity, sensing time, and location requirements. Let q_i denote the quality requirement for task i. Assume the platform has M registered users waiting for task assignments.

Since MCS users are ordinary citizens rather than professionals, their sensing capabilities vary significantly. We define a user' s sensing quality as the total amount of qualified data collected when executing a single task. Let λ_k denote the sensing quality of user k, f_k denote the fixed incentive paid to user k, and p_k denote user k's cost per unit distance traveled. Due to device energy constraints, each user can execute only a limited number of tasks, denoted by n_k as the maximum execution capacity of user k.

After task publication, the platform obtains task and user information to perform intelligent allocation. To minimize sensing costs, each task is assigned to exactly one user, subject to two critical constraints: the assigned user' s sensing quality must meet or exceed the task' s quality requirement to ensure successful execution, and the number of tasks assigned to each user cannot exceed their maximum execution capacity.

The sensing cost primarily comprises fixed incentives paid to selected users and dynamic incentives (the product of per-unit-distance cost and total travel distance). From the platform' s perspective, our objective is to design a spatial task allocation mechanism that minimizes sensing costs while satisfying task quality requirements, user sensing capabilities, maximum execution capacities, and spatial distances between users and tasks. The objective function is formulated as:

$$\min \sum_{k=1}^{M} \sum_{i=0}^{N} \sum_{j=0}^{N} p_k \cdot d_{ij} \cdot X_{ijk} + \sum_{k=1}^{M} \sum_{i=1}^{N} f_k \cdot y_{ik} \tag{1}$$

where X_{ijk} is a binary variable indicating whether user k travels from task i to task j (1 if yes, 0 otherwise). For convenience, we assume all users depart from the same location and return to this origin after completing assigned tasks. A virtual task point 0 serves as the departure point, with $d_{0i} = d_{i0}$ for all i. The first term represents dynamic incentives paid to selected users, where d_{ij} denotes the Euclidean distance between tasks i and j. The second term represents fixed incentives paid to users.

The objective function (1) is subject to the following constraints:

$$\lambda_k \geq q_i \cdot y_{ik}, \quad \forall i \in T, \forall k \in U \tag{2}$$

$$\sum_{i=1}^{N} y_{ik} \leq n_k, \quad \forall k \in U \tag{3}$$

$$\sum_{k=1}^{M} y_{ik} = 1, \quad \forall i \in T \tag{4}$$

where y_{ik} is a binary variable indicating whether user k executes task i (1 if yes, 0 otherwise). Constraint (2) ensures that the sensing quality of any user assigned to a task meets that task' s quality requirement. Constraint (3) limits the number of tasks assigned to each user to their maximum execution capacity. Constraint (4) ensures each task is executed exactly once.

---

2 Spatial Task Allocation Mechanism

To solve the spatial task allocation problem defined above, we design a comprehensive mechanism consisting of two main components: (a) a spatial task allocation method based on a hybrid genetic and greedy algorithm (GGA-TA) that minimizes sensing costs, and (b) a user sensing quality update mechanism.

2.1 Spatial Task Allocation Method GGA-TA

Spatial task allocation in MCS is a classic NP-hard problem. Genetic algorithms, as search heuristics inspired by natural evolution, are well-suited for complex optimization problems. However, due to the large scale of MCS task allocation, conventional genetic algorithms struggle to produce satisfactory solutions. Therefore, we propose GGA-TA, a hybrid approach combining genetic and greedy algorithms.

1) Chromosome Encoding and Fitness Function

GGA-TA employs natural number encoding, where $0, 1, 2, \ldots, M$ represents M users, with task indices corresponding to user positions. As shown in [Figure 1:

see original paper], assuming 8 tasks and 3 users, a chromosome might indicate that user 1 executes tasks 1, 4, and 6; user 2 executes tasks 3, 5, and 8; and user 3 executes tasks 2 and 7.

This encoding ensures each task is executed exactly once but does not guarantee that assigned users meet quality requirements or that task counts remain within user capacity limits. To handle constraints (2) and (3), GGA-TA introduces a penalty coefficient w when evaluating chromosomes. The evaluation process is detailed in Algorithm 1. To enhance efficiency, GGA-TA incorporates a greedy algorithm for route planning within individual user assignments—users always execute the nearest available task first, as shown in steps c) and d) of Algorithm 1.

Algorithm 1: Chromosome Evaluation

Input: Chromosome C, quality requirements $Q = (q_1, q_2, \dots, q_N)$, user qualities $\Gamma = (\lambda_1, \lambda_2, \dots, \lambda_M)$, fixed incentives $F = (f_1, f_2, \dots, f_M)$, per-distance costs $P = (p_1, p_2, \dots, p_M)$, max capacities $N = (n_1, n_2, \dots, n_M)$.

Output: Evaluation value E of chromosome C.

a) Determine the set of selected users S from chromosome C, initialize $k = 1$ and $E = 0$.

b) If $k > |S|$, proceed to step 5; otherwise, determine the task set R_k that user k must execute based on chromosome C. If $|R_k| > n_k$, set $E = E + w \cdot (|R_k| - n_k)$.

c) If R_k is empty, set $k = k+1$ and return to step b); otherwise, select the task $j \in R_k$ closest to user k's current location, set $E = E + p_k \cdot \text{distance}(E_k, E_j)$, where E_k represents user k' s current location and E_j represents task j' s location. Set $y_{jk} = 1$ (binary variable indicating if user k executed task j) and update user k' s current location to task j' s location.

d) Remove task j from set R_k and return to step c).

e) Output the evaluation value E of chromosome C.

Note that while Section 1 assumes all users depart from the same location and return to the origin for exposition convenience, GGA-TA remains applicable when users start from different locations and are not required to return to their starting points.

The fitness value of a chromosome is calculated using:

$$\text{fitness} = \frac{E_{\max} - E}{E_{\max} - E_{\min}} \tag{5}$$

where E is the chromosome' s evaluation value, $E_{\min}$ is the minimum evaluation value in the current population, and $E_{\max}$ is the maximum evaluation value.

Since lower evaluation values are better, higher fitness values indicate superior chromosomes.

2) Population Initialization

Initial populations significantly impact genetic algorithm performance. For large-scale spatial task allocation, random initialization hinders rapid convergence to optimal solutions. GGA-TA generates the initial population based on task quality requirements, as described in Algorithm 2.

Algorithm 2: Initial Population Generation

Input: Task quality requirements $Q = (q_1, q_2, \ldots, q_N)$, user qualities $\Gamma = (\lambda_1, \lambda_2, \ldots, \lambda_M)$, population size Size.

Output: Initial population G.

a) Initialize $k = 1$.

b) If $k >$ Size, output population G and terminate; otherwise, initialize task index $i = 1$ and chromosome $C_k = \emptyset$.

c) If $i > N$, set $k = k+1$ and return to step b); otherwise, determine task i' s quality requirement q_i, find all users with sensing quality $\lambda_k \geq q_i$ to form subset S_i, randomly select one user from S_i and assign to c_i, set $i = i+1$, and return to step c).

3) Selection Operator

The selection operator preserves high-fitness individuals for the next generation while eliminating low-fitness ones. GGA-TA sorts all individuals by fitness in descending order and retains the top half of the population, discarding the bottom half.

4) Crossover Operator

GGA-TA employs partial matched crossover: two chromosomes are randomly selected with two crossover points, and the chromosome segments between these points are exchanged.

5) Mutation Operator

To avoid local optima, GGA-TA uses single-point mutation. A mutation point (task j) is randomly selected with quality requirement q_j. To accelerate convergence, the mutation assigns a user with sensing quality $\lambda_k \geq q_j$ to that point.

6) Termination Condition

GGA-TA terminates when reaching the maximum evolution generation. The complete algorithm is presented in Algorithm 3.

Algorithm 3: GGA-TA Algorithm

Input: N, M, Q, Γ, F, P, N, Size, $G_{\max}$, p_c, p_m.

Output: Task allocation scheme Θ.

a) Initialize generation counter $t = 1$ and call Algorithm 2 to generate initial population G_1.

b) If $t > G_{\max}$, output the chromosome with highest fitness in G_t as Θ and terminate; otherwise, call Algorithm 1 to evaluate all chromosomes in G_t and calculate their fitness using Equation (5).

c) Sort chromosomes in G_t by fitness in descending order, assign the top $\lceil \text{Size}/2 \rceil$ chromosomes to G_{t+1}, and set $i = 1$.

d) If $i > \lceil \text{Size}/2 \rceil$, proceed to step f); otherwise, randomly select two chromosomes C_1 and C_2 from the top $\lceil \text{Size}/2 \rceil$ chromosomes, generate a random number $r \in [0, 1]$. If $r \geq p_c$, set $C_1' = C_1$, $C_2' = C_2$ and proceed to step e); otherwise, randomly generate two crossover points, exchange the chromosome segments between these points to obtain new chromosomes C_1' and C_2'.

e) Add C_1' and C_2' to G_{t+1}, set $i = i + 2$, and return to step d).

f) If $t > G_{\max}$, output the chromosome with minimum evaluation value in G_{t+1} as Θ and terminate; otherwise, generate random number $r \in [0, 1]$. If $r \geq p_m$, set $t = t + 1$ and return to step b); otherwise, randomly select a chromosome C from G_{t+1}, generate a mutation point corresponding to task j, select a user with sensing quality $\lambda_k \geq q_j$ from set U and assign to the mutation point, then return to step f).

GGA-TA' s selection operator retains the top half of the population by fitness (Algorithm 3, step c)), uses this retained half as parents to generate the other half (steps d)-e)), then applies single-point mutation to form the next generation (step f)). Upon termination, the algorithm outputs the highest-fitness chromosome from the final population as the task allocation scheme.

describes common parameters used throughout the mechanism.

Table 1: Common Parameters

Symbol	Description
N, M	Number of tasks and users
$Q = (q_1, q_2, \ldots, q_N)$	Quality requirements for tasks
$\Gamma = (\lambda_1, \lambda_2, \ldots, \lambda_M)$	Sensing quality of users
$F = (f_1, f_2, \ldots, f_M)$	Fixed incentives for users
$P = (p_1, p_2, \ldots, p_M)$	Dynamic (per-distance) costs for users
E_i	Evaluation value of chromosome i
$G_{\max}$	Maximum number of iterations
Θ	Final task allocation scheme

2.2 User Sensing Quality Update Mechanism

The non-professional nature of MCS users makes their per-task sensing quality unpredictable. However, we can estimate user quality by analyzing historical performance. To obtain more accurate estimates, we implement a quality update mechanism that revises user sensing quality after each task execution based on uploaded data, thereby informing subsequent allocations. The update formula is:

$$\lambda_k^{(r+1)} = \alpha \cdot \lambda_k^{(r)} + (1-\alpha) \cdot \frac{A_{kr}}{t_k} \tag{6}$$

where $\lambda_k^{(r)}$ represents user k' s sensing quality after completing r tasks, A_{kr} denotes the total qualified data uploaded by user k when executing task r, and α is a constant that adjusts the influence of current task results on the quality update. To avoid distortions from 偶然 performance, the mechanism weights historical quality more heavily, with α typically set in the range $[0.5, 1]$.

---

3 Simulation Experiments

To evaluate our spatial task allocation mechanism for MCS, we designed the GGA-TA algorithm considering spatial distances, task quality requirements, user sensing qualities, and maximum execution capacities. Liu et al. proposed the PT-Most algorithm for multi-task scenarios that minimizes sensing costs but neglects user quality and task requirements. To demonstrate GGA-TA' s effectiveness, we compare against two variants of PT-Most: CostFirst (prioritizing users with low fixed incentives) and RatioFirst (prioritizing users with high quality-to-incentive ratios), with the constraint that users failing to meet task quality requirements cannot be assigned.

Experiments are conducted in MATLAB, simulating N sensing tasks randomly distributed in a 600×600 km region. Each task' s minimum quality requirement q_i follows a uniform distribution on $[0, 30]$. We simulate M sensing users starting from location $(0, 0)$, with sensing quality $\lambda_k \sim U[0, 30]$, fixed incentive $f_k \sim U[1, 30]$, dynamic cost $p_k \sim U[0.1, 0.3]$, and maximum execution capacity $n_k \sim U[4, 8]$. Additional parameters are listed in .

To ensure near-optimal convergence, we set population size to 1500, penalty coefficient $w = 100000$, crossover probability $p_c = 0.9$, and mutation probability $p_m = 0.05$. Each experiment runs GGA-TA 10 times, reporting both minimum and average results.

Table 2: Parameter Settings

Parameter	Value
Population size	1500
Maximum generations	100
Penalty coefficient w	100000
Crossover probability p_c	0.9
Mutation probability p_m	0.05

3.1 Impact of User Population Size on Algorithm Performance

We examine two scenarios: $N = 10$ tasks (small-scale) and $N = 20$ tasks (large-scale), analyzing how varying user population M affects performance.

a) $N = 10$ **tasks:** Results are shown in [Figure 2: see original paper] and [Figure 3: see original paper]. With 10 tasks, as M increases, GGA-TA' s total sensing cost and total travel distance initially decrease then stabilize. CostFirst and RatioFirst exhibit substantially higher costs and distances because GGA-TA intelligently integrates multiple factors—task quality requirements, user sensing qualities, and spatial distances—to find optimal solutions, whereas the benchmarks rely solely on greedy strategies that easily fall into local optima.

b) $N = 20$ **tasks:** Results are shown in [Figure 4: see original paper] and [Figure 5: see original paper]. With 20 tasks, GGA-TA maintains consistently low and stable costs and distances across all user population sizes, significantly outperforming both benchmarks.

3.2 Impact of Task Quantity on Algorithm Performance

To analyze task quantity effects, we fix $M = 20$ users and vary N. Results are presented in [Figure 6: see original paper], [Figure 7: see original paper], and [Figure 8: see original paper].

As shown in [Figure 6: see original paper] and [Figure 7: see original paper], all three algorithms exhibit increasing total costs and travel distances as task quantity grows. However, RatioFirst fails to allocate all tasks when $N \geq 36$, and CostFirst fails when $N \geq 51$. While GGA-TA occasionally produces infeasible solutions in single runs for $N > 48$, successful allocation is achieved within 10 runs, consistently delivering lower costs and distances than the benchmarks.

[Figure 8: see original paper] illustrates execution time versus task quantity. As N increases, GGA-TA' s computational load and execution time grow substantially. Being a heuristic algorithm requiring iterative optimization, GGA-TA' s execution time exceeds that of the greedy-based benchmarks.

These experiments demonstrate that GGA-TA successfully allocates all tasks while minimizing sensing costs, effectively integrating task quality requirements, user sensing qualities, maximum execution capacities, and spatial distances to outperform CostFirst and RatioFirst in both total cost and travel distance metrics.

4 Conclusion

This paper addresses spatial task allocation in MCS by designing a mechanism that minimizes sensing costs while satisfying task quality requirements. Experiments validate the mechanism' s effectiveness for small-to-medium scale problems. However, GGA-TA' s execution time increases with task quantity, limiting its scalability for large-scale scenarios. Future work will focus on developing algorithms suitable for large-scale spatial task allocation problems.

References

[1] Yin Xizhe, Shen Weiming, Samarabandu J, et al. Human activity detection based on multiple smart phone sensors and machine learning algorithms [C]//Proc of the 19th IEEE International Conference on Computer Supported Cooperative Work in Design. Piscataway, NJ: IEEE, 2015: 582-587.

[2] Ganti R K, Ye Fan, Lei Hui. Mobile crowd sensing: current state and future challenges [J]. IEEE Communications Magazine, 2011, 49(11): 32-39.

[3] 吴垚, 曾菊儒, 彭辉, 等. 群智感知激励机制研究综述 [J]. 软件学报, 2016, 27(8): 2025-2047. (Wu Yao, Zeng Juru, Peng Hui, et al. Survey on Incentive Mechanisms for Crowd Sensing [J]. Journal of Software, 2016, 27(8): 2025-2047)

[4] Karaliopoulos M, Telelis O, Koutsopoulos I. User recruitment for mobile crowdsensing over opportunistic networks [C]//Proc of IEEE Conference on Computer Communications. Piscataway, NJ: IEEE, 2015: 2254-2262.

[5] Coric V, Gruteser M. Crowdsensing maps of on-street parking spaces [C]//Proc of IEEE International Conference on Distributed Computing in Sensor Systems. Piscataway, NJ: IEEE, 2013: 115-122.

[6] Maisonneuve N, Stevens M, Niessen M, et al. Noisetube: measuring and mapping noise pollution with mobile phones [C]//Proc of the 4th International Symposium Information Technologies Environmental Engineering. Piscataway, NJ: IEEE, 2009: 215-228.

[7] Guo Bin, Chen Huihui, Yu Zhiwen, et al. FlierMeet: a mobile crowdsensing system for cross-space public information reposting, tagging, and sharing [J]. IEEE Trans on Mobile Computing, 2015, 14(10): 2020-2033.

[8] Han Kai, Zhang Chi, Luo Jun. Taming the uncertainty: budget limited robust crowdsensing through online learning [J]. IEEE/ACM Trans on Networking, 2016, 24(3): 1462-1475.

[9] Xu Jia, Xiang Jinxin, Yang Dejun. Incentive mechanisms for time window

dependent tasks in mobile crowdsensing [J]. IEEE Trans on Wireless Communications, 2015, 11(14): 6353-6364.

[10] 施战, 辛煜, 孙玉娥, 等. 基于用户可靠性的众包系统任务分配机制 [J]. 计算机应用, 2017, 37(9): 2449-2453. (Shi Zhan, Xin Yu, Sun Yu-e, et al. Task allocation mechanism for crowdsourcing system based on reliability of users [J]. Journal of Computer Applications, 2017, 37(9): 2449-2453)

[11] Miao Chunyan, Yu Han, Shen Zhiqi, et al. Balancing quality and budget considerations in mobile crowdsourcing [J]. Decision Support Systems, 2016(90): 56-64.

[12] Estrada R, Mizouni R, Otrok H, et al. A crowd-sensing framework for allocation of time-constrained and location-based tasks [J]. IEEE Trans on Services Computing, 2018, DOI: 10.1109/TSC.2017.2725835.

[13] 刘琰, 郭斌, 吴文乐, 等. 移动群智感知多任务参与者优选方法研究 [J]. 计算机学报, 2016, 40(8): 1872-1877. (Liu Yan, Guo Bin, Wu Wenle, et al. Multitask-Oriented Participant Selection in Mobile Crowd Sensing [J]. Chinese Journal of Computers, 2016, 40(8): 1872-1877)

[14] Wang Jing, Tang Jian, Xue Guoliang, et al. Towards energy-efficient task scheduling on smartphones in mobile crowd sensing systems [J]. Computer Networks, 2017 (115): 100-109.

[15] Liu C H, Zhang B, Su X, et al. Energy-aware participant selection for smartphone-enabled mobile crowd sensing [J]. IEEE Systems Journal, 2017, 11(3): 1435-1446.

Note: Figure translations are in progress. See original paper for figures.