# Time-dependent Stable Task Assignment in Participatory Mobile Crowdsensing

Fatih Yucel and Eyuphan Bulut
Department of Computer Science, Virginia Commonwealth University
401 West Main St. Richmond, VA 23284, USA
{yucelf, ebulut}@vcu.edu

2020 IEEE 45th Conference on Local Computer Networks (LCN) | 978-1-7281-7158-6/20/$31.00 ©2020 IEEE | DOI: 10.1109/LCN48667.2020.9314829

***Abstract*—Finding efficient task assignments is key to the success of mobile crowdsensing campaigns. Many studies in the literature focus on this problem and propose solutions that optimize the goals of mobile crowdsensing platform, but disregard user preferences. On the other hand, in a few recent studies that consider user preferences, workers are assigned a single task at a time, and the effect of these assignments to their prospective utilities is ignored. In this paper, we address these issues and study the task assignment problem considering both the user preferences and impact of each task assignment on the long-term utility of workers given the spatio-temporal characteristics of tasks. We propose a dynamic programming based task assignment algorithm that guarantees the satisfaction of users with their assignments. Through simulations, we compare it with a state-of-the-art algorithm and show the superiority of our algorithm in various aspects.**

***Index Terms*—Participatory mobile crowdsensing, task assignment, stable matching.**

## I. Introduction

Mobile crowdsensing (MCS) aims to leverage the ubiquity of mobile devices with embedded sensors (e.g., microphone, camera) owned by people and perform various sensing tasks without deploying dedicated sensor networks [1]. The success of an MCS system is mostly defined by the task assignment or user recruitment process. In worker-scarce MCS systems, some workers may need to be assigned to multiple tasks in order to meet the demands of task requesters in the platform. This also gives the workers an opportunity to gain more rewards. In such MCS systems where workers can be assigned to multiple tasks and each task is assigned to only a worker, there are basically two ways to handle the task assignments: *instant* (or local) task assignment and *predetermined* (or global) task assignment.

In instant task assignment, workers are assigned to a single task at a time, and they are periodically assigned to a new one only after they perform their previous assignments. However, this brings an uncertainty that will affect the workers' ability to plan the rest of their days. For example, a worker who will not be assigned to an acceptable task in the next assignment rounds will not be able to readily schedule his day for those time periods in advance as it is not clear to him whether he will get an assignment and hence be busy until the very last moment. Moreover, since instant task assignment does not consider the potential future assignments for workers, it may end up producing a task assignment that is optimal locally, but not globally. Predetermined task assignment circumvents the mentioned issues by deciding all assignments for the foreseeable future (e.g., hourly, daily) in advance. That is, it considers the impact of each task assignment to the utility of the workers in the remaining part of the considered time frame and seeks to find a globally optimal solution.

There are various studies that look at the task assignment problem in MCS systems with several different objectives such as minimizing the worker travel costs [2], maximizing the total profit of workers [3] and maximizing the quality of service (QoS) in the completed tasks [4]. However, most of the existing work either do not consider multiple task assignment to each worker or adopt instant task assignment. A few studies [5], [6] look at this joint task assignment and path planning problem and aim to minimize the total delay in completed tasks while also targeting a large number of completed tasks. However, these solutions mostly consider system level objectives and do not take into account user preferences during task assignment. Thus, they may result in dissatisfied users and affect the future user participation. To address this, several recent studies [7]–[11] utilize Stable Matching Theory [12] and provide task assignments satisfying all users. However, these solutions do not take the scheduling of tasks into account, thus may assign a worker with a set of tasks that he cannot perform in time. In this paper, we study this challenging problem and propose a task assignment algorithm that considers scheduling of tasks for workers and respects user preferences so that no user will have a desire to deviate from his assignment.

The rest of the paper is organized as follows. In Section II, we introduce the system model. In Section III, we describe our dynamic programming based solution. In Section IV, we evaluate the performance of the proposed solution through simulations. Finally, we provide our conclusions in Section V.

## II. System Model

We assume a system model with an MCS platform that aims to find a predetermined task assignment for each assignment cycle between a set of available workers $\mathcal{W} = \{w_1, w_2, \ldots, w_m\}$ and a set of tasks $\mathcal{T} = \{t_1, t_2, \ldots, t_n\}$ published by the registered task requesters. Each task $t$ has strict spatio-temporal requirements and needs to be performed at a specific location $\vec{l_t}$ in a certain time frame $[t.s, t.e]$. Besides, each task $t$ has a minimum QoS requirement $q^t_{min}$ and offers a monetary incentive/reward $r_t(w)$ to each worker $w$ that satisfies this requirement.

We assume a participatory MCS platform, thus workers are willing to interrupt their daily schedule to perform assigned tasks. For individual rationality, we let each worker $w$ define a minimum profit requirement, $r^w_{min}$, which ensures that if he is assigned to any non-empty set of tasks in an assignment cycle, he will make at least $r^w_{min}$ profit. Let $t^S_1, t^S_2, \ldots, t^S_k$ be a given set $S$ of tasks in non-decreasing order of their starting times (i.e., $t^S_i.s \leq t^S_{i+1}.s$). We denote whether worker $w$ can perform the tasks in $S$ in time by $P_w(S)$, which, given the average speed $s_w$ of worker $w$ and $d(\vec{l}_w, \vec{l}_t)$ denoting the distance between locations $\vec{l}_w$ and $\vec{l}_t$, can be formally defined as

$$P_w(S) = \begin{cases} 1, & \text{if } p_w(t^S_1) \times \prod_{i=1}^{k-1} p_w(t^S_i, t^S_{i+1}) = 1 \\ 0, & \text{otherwise,} \end{cases}$$

where $p_w(t) = 1$ if $d(\vec{l}_w, \vec{l}_t) \leq s_w \times t.s$, 0 otherwise, and $p_w(t_i, t_j) = 1$ if $d(\vec{l}_{t_i}, \vec{l}_{t_j}) \leq s_w \times (t_j.s - t_i.e)$, 0 otherwise. Let $c_w(t)$ denote the sensing cost of task $t$ for worker $w$, which may depend on various factors such as the energy consumption on the sensing device and time/effort required to perform the task. Also, let $c_w(x, y) = d(x, y) \times \theta_w$ be the cost of travel between locations $x$ and $y$ for worker $w$, where $\theta_w$ denotes the travel cost of $w$ per km. Then, we can calculate the profit that worker $w$ will make when assigned to $S$ with $P_w(S) = 1$ by

$$U_w(S) = \sum_{t \in S} g_w(t) - c_w(\vec{l}_w, \vec{l}_{t^S_1}) - \sum_{i=1}^{|S|-1} c_w(\vec{l}_{t^S_i}, \vec{l}_{t^S_{i+1}}),$$

where $g_w(t) = r_t(w) - c_w(t)$.

Each worker $w$ has a QoS score $q(w)$, so we can define the set of tasks that find worker $w$ acceptable in terms of QoS requirement and are also individually acceptable for worker $w$ in terms of spatio-temporal and profit-based requirements as

$$\mathcal{E}(w) = \{t : q^t_{min} \leq q(w), g_w(t) > 0, p_w(t) = 1\}. \quad (1)$$

A worker-task pair $(w, t)$ is called a *qualified* pair only if $t \in \mathcal{E}(w)$. Lastly, we denote whether a task set $S$ is admissible for worker $w$ by $\mathcal{E}_w(S)$, which is determined by

$$\mathcal{E}_w(S) = \begin{cases} 1, & \text{if } S \subseteq \mathcal{E}(w),\ P_w(S) = 1 \text{ and } U_w(S) \geq r^w_{min} \\ 0, & \text{otherwise.} \end{cases} \quad (2)$$

Below, we first give a formal definition of a matching and then provide the constraints for user satisfaction (stability).

**Definition 1** (Matching)**.** *A mapping between the sets $\mathcal{W}$ and $\mathcal{T}$ is a feasible (F) and rational (R) matching (or task assignment) if*

- *[F] $\forall w \in \mathcal{W} : \mathcal{M}(w) \subseteq \mathcal{T}$ and $P_w(\mathcal{M}(w)) = 1$,*
- *[F] $\forall t \in \mathcal{T} : \mathcal{M}(t) \in \mathcal{W}$ or $\mathcal{M}(t) = \emptyset$,*
- *[F] $\forall w, t \in \mathcal{W} \times \mathcal{T} : \mathcal{M}(t) = w$ iff $t \in \mathcal{M}(w)$,*
- *[R] $\forall w \in \mathcal{W} : U_w(\mathcal{M}(w)) \geq r^w_{min}$ if $|\mathcal{M}(w)| > 0$,*
- *[R] $\forall t \in \mathcal{T} : q(\mathcal{M}(t)) \geq q^t_{min}$ if $\mathcal{M}(t) \neq \emptyset$.*

**Algorithm 1:** TCSTA ($\mathcal{W}$, $\mathcal{T}$)

```
1  T' ← T
2  M(u) ← ∅, ∀u ∈ W ∪ T
3  w'_1, w'_2, ..., w'_m ← sort W such that q(w'_i) ≥ q(w'_{i+1})
4  for i ← 1 to m do
5      E ← T' ∩ E(w'_i)
6      t'_1, t'_2, ..., t'_k ← sort E such that t'_{i+1}.e ≥ t'_i.e
7      u, S ← solveDP(w'_i, E)
8      util_max ← 0, index ← 0
9      for j ← 1 to k do
10         util ← u[j] + g_{w'_i}(t'_j) − c_{w'_i}(l_{w'_i}, l_{t'_j})
11         if util > util_max then
12             util_max ← util
13             index ← j
14     if util_max ≥ r^{w'_i}_min then
15         F ← S[index] ∪ {t'_index}
16         M(w'_i) ← F
17         foreach t ∈ F do
18             M(t) ← w'_i, T' ← T' \ t
19 return M
```

Note that $\mathcal{M}(u)$ denotes the partner or partner set (for workers) of $u$ in $\mathcal{M}$.

**Definition 2** (Unhappy coalition)**.** *Given a matching $\mathcal{M}$, a worker $w$ and a subset $S$ of tasks form an unhappy coalition (denoted by $\langle w, S \rangle$) if $\forall t \in S$, $q(w) > q(\mathcal{M}(t))$ (where $q(\emptyset) = 0$), and $\exists V \subseteq \mathcal{M}(w)$ such that $\mathcal{E}_w(S \cup V) = 1$ and $U_w(S \cup V) > U_w(\mathcal{M}(w))$. That is, tasks in $S$ prefer $w$ to their partners in $\mathcal{M}$, and $w$ prefers $S$ to a subset of his partner set in $\mathcal{M}$.*

If $\mathcal{M}$ contains an unhappy coalition $\langle w, S \rangle$ such that $t \in S$, the worker-task pair $(w, t)$ is said to be an *unhappy* pair (and users in at least one unhappy pair are said to be unhappy users).

**Definition 3** (Stable matching)**.** *A matching $\mathcal{M}$ is (coalitionally) stable if it contains no unhappy coalitions.*

## III. Proposed Solution

The outline of the proposed solution is presented in Algorithm 1. It maintains a list $T'$ of available tasks that are still unmatched, which is initialized in line 1 to contain all the tasks in the platform. In line 2, it creates a matching $\mathcal{M}$ where the partner of each worker and task is set to be an empty set. In the main for loop in lines 4-18, it iterates the workers in non-increasing order of their QoS values, which is found in line 3, and finds the optimal task set for the $i$th worker ($w'_i$) among the tasks that are still available and in the eligible task set $\mathcal{E}(w'_i)$ of worker $w'_i$ (line 5). Before describing the process of finding the optimal task set for a worker, we first provide the following theorem.

**Algorithm 2:** solveDP ($w$, $E$)

```
1  let u[1...k] and S[1...k] be two new arrays
2  u[k] ← 0, S[k] ← ∅
3  for i ← k − 1 down to 1 do
4      util_max ← 0, index ← 0
5      for j ← i + 1 to k do
6          util ← p_w(t'_i, t'_j) × (u[j] + g_w(t_j) − c_w(l⃗_{t_i}, l⃗_{t_j}))
7          if util > util_max then
8              util_max ← util
9              index ← j
10     if util_max > 0 then
11         u[i] ← util_max, S[i] ← S[index] ∪ {t'_index}
12     else
13         u[i] ← 0, S[i] ← ∅
14 return u, S
```

**Theorem 1.** *The problem of finding the highest utility (i.e., net profit) assignment for worker $w$ among a given set $S \subseteq \mathcal{E}(w)$ of tasks has an optimal-substructure property, and an optimal solution can be found using the following recursive formula:*

$$\max_{t \in S} \left\{ U[t.e] + g_w(t) - c_w(\vec{l}_w, \vec{l}_t) \right\}, \text{ where} \tag{3}$$

$$U[t.e] = \max_{t' \in S} \left\{ p_w(t, t') \times (U[t'.e] + g_w(t') - c_w(\vec{l}_t, \vec{l}_{t'})) \right\}.$$

*Proof.* The proof is omitted due to space limitations. □

Algorithm 2 solves the problem formulated in Theorem 1 using dynamic programming. It takes a worker $w$ and a set $E$ of tasks (which are assumed to be in non-decreasing order of their ending times) as input, and uses two arrays, $u$ and $S$, of length $k = |E|$ to store the solutions to the subproblems (of finding the best feasible task set in $E$ for $w$). Here, $u[i]$ ($1 \leq i \leq k$) stores the maximum utility that worker $w$ can achieve after performing task $t'_i$ (the $i$th task in E), whereas $S[i]$ stores the corresponding optimal task set.

Algorithm 1 calls Algorithm 2 in line 7 to calculate the arrays $u$ and $S$. Then, in lines 8-13, it finds which task is the best to perform for worker $w'_i$ starting from his initial location considering the utilities he can achieve after performing each one (which are stored in the array $u$). Finally, in line 14 it checks whether the utility/profit found is economically admissible for worker $w'_i$. If so, it matches the corresponding task set and worker $w'_i$ with each other, and removes the matched tasks from the available task set in lines 16-19. Below, we provide the theoretical analysis of the algorithm.

**Theorem 2.** *Algorithm 1 produces feasible and rational matchings.*

*Proof.* The proof is omitted due to space limitations. □

**Theorem 3.** *Algorithm 1 produces stable task assignments.*

*Proof.* The proof is omitted due to space limitations. □

TABLE I: Simulation parameters

| Parameter | Value | Parameter | Value | Parameter | Value |
|---|---|---|---|---|---|
| $m$ | 20 | $n$ | 200 | $q(w)$ | [5,10] |
| $\theta_w$ | [1,5] | $c_w(t)$ | [0,5] | $q^t_{min}$ | [0,10] |
| $\beta_t$ | [5,10] | | | | |

**Theorem 4.** *The running time of Algorithm 1 is $\mathcal{O}(m \log m + mn^2)$.*

*Proof.* The proof is omitted due to space limitations. □

## IV. Simulation Results

In this section, we present the empirical evaluation of the proposed algorithm. In order to create realistic MCS instance, we utilize the NYC taxi data set [13] and follow the instance generation scheme defined in [11] for this data set (where the properties of workers and tasks are set according to the taxi and passenger data, respectively). Differently from [11], we set $[t.s, t.e]$, $\forall t$ to the time frame between the pick-up and drop-off times of the corresponding passenger, and $s_w$, $\forall w$ to the average speed of the corresponding taxi in the most recent trip. The other variables are assigned randomly from the ranges given in Table I. For each worker-task pair $(w, t)$, we let $r_t(w) = q(w) \times \beta_t$, where $\beta_t$ denotes the reward constant of task $t$.

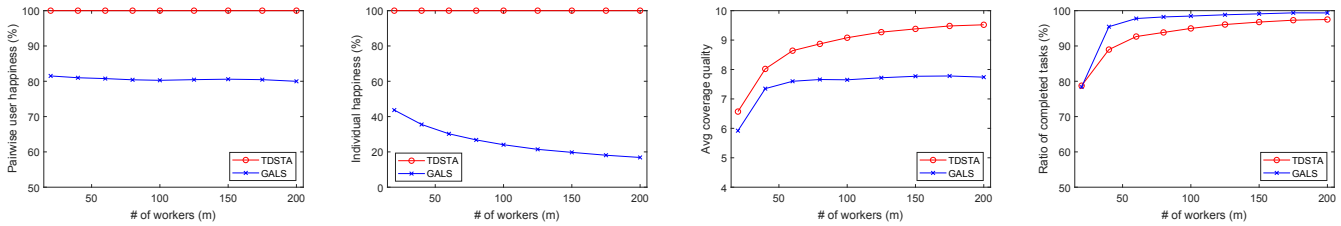
We compare our algorithm with Global Assignment and Local Scheduling (or GALS) algorithm [5], which is a state-of-the-art algorithm for spatio-temporal task matching and scheduling. The objective of this algorithm is to maximize the number of completed (matched) tasks. In the evaluation of the algorithms, we consider the following performance metrics:

- *Pairwise user happiness*: $100 \times \left(1 - \frac{\text{\# of unhappy pairs}}{\text{\# of qualified pairs}}\right)$.
- *Individual happiness*: $100 \times \left(1 - \frac{\text{\# of unhappy users}}{m+n}\right)$.
- *Average coverage quality*: $100 \times \frac{\sum_{i=1}^{n} q(\mathcal{M}(t_i))}{n}$.
- *Ratio of completed tasks*: $100 \times \frac{|\{t \in \mathcal{T} : \mathcal{M}(t) \neq \emptyset\}|}{n}$.

We lastly note that all results provided in this section are the average of the results obtained in 100 different MCS instances.

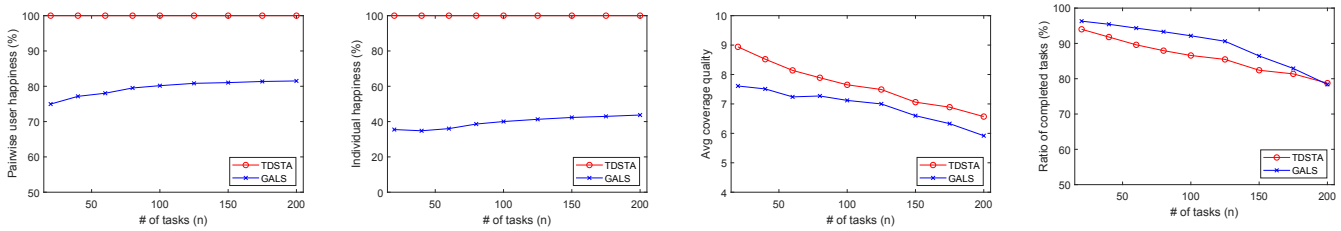
### A. Results

In Fig. 1, we first look at the impact of number of workers on the performance of the algorithms. Fig. 1a and 1b show that our algorithm produces perfect task assignments in terms of user happiness (Theorem 3) and greatly outperforms the GALS algorithm especially with respect to the individual happiness. Specifically, we see that the GALS algorithm produces matchings where nearly 20% of all qualified pairs and up to 80% of all users are unhappy with their assignments. Since our algorithm takes the preferences of tasks into consideration and the tasks prefer to be matched to the workers with higher QoS values, our algorithm also achieves a notably better average coverage quality scores than the GALS algorithm (Fig. 1c), which disregards the user preferences to maximize the number of completed tasks. As seen in Fig. 1d, it indeed matches more tasks than our algorithm does, but the difference in the number of tasks they match is at most 6%.




(a) (b) (c) (d)

Fig. 1: Performance comparison of the algorithms against varying number of workers ($n = 200$).




(a) (b) (c) (d)

Fig. 2: Performance comparison of the algorithms against varying number of tasks ($m = 20$).

Fig. 2 shows the performance of the algorithms with varying number of tasks. In Fig. 2a & 2b, we see that our algorithm maintains its superior performance over the GALS algorithm in terms of user happiness, and that the task assignments produced by the GALS algorithm always upset the majority of the individuals and about 20% of the qualified pairs. A remarkable point in this figure is that both happiness scores for the GALS algorithm slightly improve with the increasing number of tasks due to the decreasing competition between workers. On the other hand, Fig. 2c & 2d show that an increase in the number of tasks results in a lower average coverage quality and a smaller ratio of completed tasks for both algorithms, because workers can perform only a limited number of tasks due to spatio-temporal constraints.

## V. Conclusion

In this paper, we study the problem of finding stable task assignments in MCS systems that involves tasks with strict spatio-temporal constraints. We first describe the feasibility and stability (or user happiness) conditions for task assignments in such systems, and give a formal problem definition. Then, we present an efficient, dynamic-programming based algorithm that produces stable task assignments satisfying all workers and task requesters. Finally, we present an evaluation of the proposed algorithm on a real data set, where we compare its performance with that of a benchmark algorithm. The results demonstrate the superior performance of the proposed algorithm in terms of achieved coverage quality as well as user happiness with a slightly smaller percentage of completed tasks.

## Acknowledgement

This material is based upon work supported by the U.S. National Science Foundation (NSF) under Grant CNS-1647217.

## References

[1] A. Capponi, C. Fiandrino, B. Kantarci, L. Foschini, D. Kliazovich, and P. Bouvry, "A survey on mobile crowdsensing systems: Challenges, solutions, and opportunities," *IEEE communications surveys & tutorials*, vol. 21, no. 3, pp. 2419–2465, 2019.

[2] W. Gong, B. Zhang, and C. Li, "Location-based online task assignment and path planning for mobile crowdsensing," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 2, pp. 1772–1783, 2018.

[3] Y. Kang, X. Miao, K. Liu, L. Chen, and Y. Liu, "Quality-aware online task assignment in mobile crowdsourcing," in *12th IEEE Inter. Conf. on Mobile Ad Hoc and Sensor Systems (MASS)*, 2015, pp. 127–135.

[4] H. Xiong, D. Zhang, G. Chen, L. Wang, and V. Gauthier, "CrowdTasker: Maximizing coverage quality in piggyback crowdsensing under budget constraint," in *International Conference on Pervasive Computing and Communications, (PerCom), 23-27 March*, 2015, pp. 55–62.

[5] D. Deng, C. Shahabi, and L. Zhu, "Task matching and scheduling for multiple workers in spatial crowdsourcing," in *23rd SIGSPATIAL Inter. Conf. on Advances in Geographic Information Systems*, 2015, pp. 1–10.

[6] X. Tao and W. Song, "Location-dependent task allocation for mobile crowdsensing with clustering effect," *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 1029–1045, 2018.

[7] F. Yucel and E. Bulut, "User satisfaction aware maximum utility task assignment in mobile crowdsensing," *Computer Networks*, vol. 172, p. 107156, 2020.

[8] Y. Chen and X. Yin, "Stable job assignment for crowdsourcing," in *GLOBECOM 2017-2017 IEEE Global Communications Conference*. IEEE, 2017, pp. 1–6.

[9] M. Abououf, S. Singh, H. Otrok, R. Mizouni, and A. Ouali, "Gale-shapley matching game selection—a framework for user satisfaction," *IEEE Access*, vol. 7, pp. 3694–3703, 2018.

[10] C. Dai, X. Wang, K. Liu, D. Qi, W. Lin, and P. Zhou, "Stable task assignment for mobile crowdsensing with budget constraint," *IEEE Transactions on Mobile Computing*, 2020.

[11] F. Yucel, M. Yuksel, and E. Bulut, "QoS-based budget constrained stable task assignment in mobile crowdsensing," *IEEE Transactions on Mobile Computing*, 2020.

[12] D. Gale and L. S. Shapley, "College admissions and the stability of marriage," *The American Mathematical Monthly*, vol. 69, no. 1, pp. 9–15, 1962.

[13] "Taxi and limousine commission (tlc) trip record data." NYC Taxi Limousine Commission, 2019. [Online]. Available: https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page