Online Stable Task Assignment in Opportunistic Mobile Crowdsensing with Uncertain Trajectories

Fatih Yucel, *Member, IEEE* and Eyuphan Bulut, *Senior Member, IEEE*

***Abstract*—In opportunistic mobile crowdsensing, participants (workers) accept to carry out the requested sensing tasks only if they are already close to or within the regions of interest. Thus, the existence of an assignment opportunity between a worker-task pair strictly depends on whether or not the worker will visit the task region. However, when worker trajectories are uncertain and hence not known in advance, existing solutions fail to produce an effective task assignment. Besides, a satisfactory task assignment should respect the preferences and capacity constraints of workers and task requesters, which are generally neglected in literature. In this study, we address all of these issues together and propose novel task assignment algorithms for different settings, which we prove to be optimal in terms of preference-awareness (or stability). Extensive simulations performed on both synthetic and real data sets validate our theoretical results, and demonstrate that the proposed algorithms significantly outperform the existing solutions in terms of preference-awareness and average quality of sensing attained in the final task assignment in almost all scenarios.**

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

I. Introduction

Mobile crowdsensing (MCS) is an emerging form of crowdsourcing that aims to accomplish location-dependent sensing tasks with the help of mobile participants (workers). Notable recent applications of MCS can be found in various fields such as vehicle tracking [1], environmental protection [2], and public safety/health [3], [4]. According to the involvement level of workers, MCS applications can be classified into two types [5]: *participatory* and *opportunistic*.

In participatory MCS, workers are expected to travel to task regions by interrupting their own schedules for a period of time, while in opportunistic MCS they are asked to perform a sensing task if they are already in or close to the task region. Thus, participatory MCS campaigns usually have shorter task completion times as workers are immediately dispatched to the task regions. However, they require workers to devote their resources and time to carry out the assigned tasks, which introduces significant extra costs for workers who then need to be compensated for these costs by task requesters.

On the other hand, opportunistic MCS campaigns only consider the costs of sensing and delivery of the sensed data, which are usually much smaller compared to the time and travel costs incurred in participatory MCS campaigns. However, it can be challenging to find effective task assignments in opportunistic MCS, especially under capacity or budget constraints [6], as the assignment opportunities hinge upon the presence of workers in task regions, and they emerge and vanish as workers move.

F. Yucel and E. Bulut are both with the Department of Computer Science, Virginia Commonwealth University, Richmond, VA, 23284.
E-mail: {yucelf, ebulut}@vcu.edu.

In this study, we focus on the task assignment problem in opportunistic MCS. The three key issues that need to be addressed in this problem are (i) preference-awareness, (ii) uncertainty in worker trajectories, and (iii) capacity constraints. Below, we explore each of these issues along with the challenges they present, how they have been so far addressed in the MCS literature, and the key contributions of this paper on each of these issues.

(i) *Preference-awareness:* As rational individuals, workers and task requesters in an MCS campaign would like to maximize their utility from the campaign, hence it is natural for them to have preferences over possible assignments they could get. It has been shown [7] that disregarding these preferences to maximize the utility of the matching platform or any particular group of users could cause the majority of users to find their assignments dissatisfying, and consequently to cease participating in the campaign, putting the long-term success of the campaign in jeopardy. Moreover, dissatisfying task assignments may hinder effective functioning of the campaign, as unhappy users may refuse to fulfill the assignments made by the platform. Therefore, it is of critical importance to satisfy the needs and preferences of workers and task requesters in an MCS campaign. For these reasons, some recent studies have investigated the preference-aware task assignment problem in MCS. However, they have either assumed a participatory setting with controlled worker mobility [8]–[10], or an opportunistic setting with certain worker trajectories that are known in advance [11].

(ii) *Uncertainty in worker trajectories:* This issue arises in MCS systems, where workers prefer not to disclose their exact trajectories due to privacy concerns, or their trajectories change dynamically due to traffic/road conditions or individual factors (e.g., a taxi driver's trajectory depends on pick-up and drop-off locations of passengers). This issue is partly investigated in the MCS literature [12], [13], but without considering the preferences of workers and task requesters. There are also studies [14] that consider undeterministic MCS with uncertainty in other aspects (e.g., worker's visit probability to task region), but they focus on increasing joint task completion rates by assigning multiple workers to each task without again considering preferences.

(iii) *Capacity constraints:* To avoid disruptions to their daily schedule, workers may choose to bound the number of tasks they accept to perform for each assignment period. The

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3118134, IEEE Internet of Things Journal

classic deferred-acceptance mechanism proposed by Gale and Shapley [15] can be used to find preference-aware assignments in presence of capacity constraints, but it works only in offline settings, where the eligibility of every (worker-task) pair is known and fixed. This is not the case in our setting, where the trajectories of workers, hence which tasks they can perform, are uncertain. Although there are some studies [16], [17] that consider capacity/budget constraints for workers, they neglect to address the previous two issues.

As summarized above, these three issues have yet to be studied together, despite being crucial for the success of an opportunistic MCS campaign. To fill this gap, in this paper, we study the preference-aware task assignment problem within an opportunistic MCS model with uncertain worker trajectories and given capacity constraints.

To point out some of the challenges this problem entails, let us analyze a few different scenarios in the MCS instance illustrated in Fig. 1, which consists of two workers (w_1, w_2) and three tasks (t_1, t_2, t_3) scattered in the area. Assume that w_1 can potentially visit all three task regions, while w_2 can visit only t_3's region, but it is not known in advance if they will actually do so. Consider the three (p_1, p_2, p_3) and two (p_4, p_5) of possible trajectories and visit scenarios for w_1 and w_2, respectively, shown in Fig. 1. If the workers do not have a capacity constraint (i.e., can perform every task on their way) or have a capacity of at least three, then w_1 should always be matched with tasks t_1 and t_2 if he visits their regions, as w_1 is the only worker that can perform these tasks. However, since the region of t_3 can be visited by both workers, it is not trivial to decide an assignment for t_3 even if there is no capacity constraint for workers because the preference of t_3 based on the worker qualities needs to be considered along with how likely the workers will be visiting the region of t_3.

On the other hand, the preferences of workers become important when they are constrained by a capacity. For instance, assume that the capacity of w_1 is one, and the probability of p_3 is negligible. Then, when worker w_1 visits the region of t_1 and a matching decision needs to be made between w_1 and t_1, we need to consider the preference of w_1 on tasks t_1 and t_2 based on their rewards as well as the likelihood of p_1 and p_2. For example, even if p_1 is more probable than p_2, it may still be more profitable to skip t_1 if the reward of t_2 is significantly larger than that of t_1. In such scenarios, the matching decisions should be made according to the expected utilities of workers and task requesters to consider their preferences.

Let us consider another scenario, in which worker w_1 has a capacity of one and is more likely to take path p_3, and the reward of t_3 is much greater than that of t_1 and t_2. Then, in order for the decision of skipping the potential matching opportunities with t_1 and t_2 to be justifiable for w_1, either the probability of p_5 should be low, or the quality of w_1 should be substantially better than that of w_2 so that task t_3 would be willing to skip the opportunity to match with w_2 if w_2 ended up taking path p_5. Here, the timeliness of the visits also plays a major role. If the quality scores of the workers are very close to each other, the best strategy for the tasks would be to get matched with the first worker that visits their regions, because the risk of losing a matching opportunity at hand to wait for another worker would not be worth the extra benefit that they may possibly get by waiting. Therefore, the actual number of scenarios that needs to be examined to make an optimal task assignment gets much larger when we take the timeliness of worker visits into consideration.

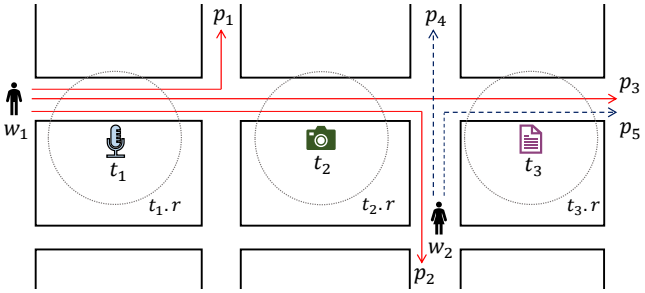


Fig. 1: An MCS instance with three tasks and two workers. Some possible worker trajectories for w_1 and w_2 are shown with solid and dashed lines, respectively, and task regions are enclosed with circles.

Moreover, in real instances, the uncertainty in worker visits may be even more severe, in which case the number of possible scenarios is likely to grow exponentially with the number of users and the campaign duration. In this study, we address these issues and provide polynomial-time algorithms to obtain task assignments that maximize the happiness of users with their assignments based on their preferences by considering all possible scenarios in an efficient manner. Our primary contributions in this study can be summarized as follows:

- We introduce the preference-aware task assignment problem in opportunistic MCS systems, where task assignments need to be made in an online manner due to uncertain worker trajectories.
- We formulate the criteria for preference-awareness in this problem after showing that the existing preference-awareness objectives used in the literature (e.g., minimizing the number of unhappy pairs) do not work when worker trajectories are uncertain.
- We study the problem in MCS systems with and without capacity constraints, and propose a polynomial-time online algorithm for each case, which we then show to be preference-aware by theoretical analysis.
- We perform extensive simulations with both real and synthetic data sets, and empirically show the superiority of the proposed algorithms over the existing solutions.

The rest of the paper is organized as follows. In Section II, we provide an overview of the related work. In Section III, we describe the system model and present the problem statement. In Section IV, we provide different algorithms to solve the problem in both online and offline settings along with their theoretical analysis. In Section V, we evaluate the performance of the proposed solutions through simulations. Finally, in Section VI, we provide our conclusions.

II. Related Work

In this section, we review the recent studies on opportunistic MCS and matching under preferences. A recent survey on mo-

bile crowdsensing and a summary of matching problems with user preferences can be found in [18] and [19], respectively.

A. Opportunistic MCS

Opportunistic MCS has drawn the attention of both academia [20] and industry [21] due to its advantages over participatory MCS in certain aspects. For example, [6] studies the maximum coverage task assignment problem in opportunistic MCS with worker trajectories that are known beforehand. It is assumed that each task needs to be performed at a certain point of interest and has a weight that indicates how important its completion is to the platform, which has a fixed budget and can hence recruit only so many workers. The objective of the platform is to select a set of workers within the budget constraints, which maximizes the weighted coverage over the set of tasks according to the given trajectories of workers. The authors develop a $(1-1/e)$-approximate algorithm, where n is the number of workers in the system. [22] studies the same problem, and proposes a greedy algorithm that, despite not having a theoretical guarantee, outperforms the algorithm proposed in [6] in terms of achieved coverage in certain settings.

The problem studied in [23] differs from those in [6] and [22] as it also considers the delivery of the sensed data in an opportunistic manner. That is, after carrying out a task, a worker should either deliver the sensed data to the server through one of the collection points (i.e., WiFi APs) on his trajectory, or transmit it to another user who will deliver it for him. Thus, here, not only the platform needs to estimate whether and when workers would visit task regions and collection points, but it is also crucial to obtain and utilize the encounter frequencies of workers to improve the delivery probability of the sensed data. The authors present different approximation algorithms for the systems with deterministic and uncertain worker trajectories, and evaluate their performance on real data sets. The data delivery aspect of the problem in [23] has also been studied in [24] and [25]. They both utilize Nash Bargaining Theory to decide on whether or not selfish data collectors and mobile (relay) users who only take part in delivery of sensed data would like to cooperate with each other according to their utility in either scenario. However, in [25], the authors consider a more complete mobile social network model and present an enhanced data collection mechanism.

In [12], the problem of maximizing spatio-temporal coverage in vehicular mobile crowdsensing with uncertain but predictable vehicle (i.e., worker) trajectories is investigated. The authors prove that the problem is NP-hard when there is a budget constraint, and propose a greedy approximation algorithm and a genetic algorithm. In [13], the authors also assume predictable worker trajectories. However, they focus on the task assignment problem in a mobile social network where task assignments and delivery of sensed data are realized in an online manner when task requesters and workers encounter with each other. They aim to minimize the task completion times, and propose different approximation algorithms to optimize both worst-case and average-case performance. For predictions of worker trajectories, [12] uses spatio-temporal trajectory matrices, while [13] assumes that user inter-meeting times follow an exponential distribution, which is widely used in the mobile social networks [26]–[28] literature.

B. Matching under preferences

In two-sided matching problems with preferences, the general objective is to find a (stable) matching that satisfies the individuals on both sides so that they will not seek to find a better matching. Variations of such matching problems are seen in a wide range of applications from channel assignments in cognitive radio networks [29] to residency and fellowship matching [30]. Given that task requesters and workers in MCS systems also have individual preferences, and ignoring them is likely to upset them, several recent studies [7], [8], [10], [31] address the preference-aware task assignment problem in MCS as well.

Particularly, [7] introduces the problem of maximizing user happiness without affecting the convenience of the central matching platform. [8] and [10] study the stable task assignment problem under budget constraints, which is proven to be NP-hard even in significantly simplified settings. [8] considers a many-to-one matching scenario with fixed rewards, and proposes polynomial and pseudo-polynomial time algorithms that solve the problem optimally in restricted settings and find approximate solutions for the strongest stability criteria. On the other hand, [9] and [10] consider a many-to-many matching scenario with capacity and budget constraints, respectively. [10] presents an approximation algorithm that dynamically adjusts the rewards to be paid to workers throughout the matching process, whereas [9] proposes an adaptation of the Gale-Shapley algorithm [15]. [31] studies the stable task assignment problem in crowdsourcing, and differs from [8]–[10], as it considers minimum quality requirements of task requesters as well as budget constraints. Unlike these studies that assume a participatory setting, [11] studies the problem of preference-aware task assignment in an opportunistic setting, but assumes that worker trajectories are fixed and known to the matching platform in advance.

A few studies look at the stable matching problem in settings with incomplete information on user preferences or dynamic user arrivals/departures. [32] and [33] both study the stable taxi dispatching problem considering passenger and taxi preferences. However, the objective adopted in [33] is to find locally optimal stable assignments, whereas that in [32] is to minimize the number of unhappy taxi-passenger pairs globally. [34] investigates the stable matching problem in the presence of uncertainty in user preferences. Lastly, [35] looks at the problem of minimizing the number of partner changes that need to be made in a stable matching to maintain stability when preference profiles of some users change in time.

None of the aforementioned studies, however, addresses the problem of finding stable/preference-aware task assignments under capacity constraints in opportunistic MCS, where worker trajectories are uncertain, and thus, if/when a worker would visit a certain task's region and be able to perform the task is not known in advance. In this paper, we focus on this problem, define it formally, and present efficient algorithms for different scenarios.

TABLE I: Key notations and descriptions.

Notation	Description
$\mathcal{T}$, $\mathcal{W}$	Set of tasks and workers, respectively
m, n	Number of tasks and workers, respectively
$\mathcal{M}$	A many-to-one matching between $\mathcal{T}$ and $\mathcal{W}$
$\mathcal{M}(u)$	Assigned worker (task set) to task (worker) u in $\mathcal{M}$
$[0, T]$	Current assignment period
$t.r$	Region of task t
$[t.b, t.d]$	Time interval in which t should be performed
$m(t)$	Reward associated with task t
$c(w)$	Capacity of worker w
$q(w)$	Quality score of worker w
$\lambda_{i,j}$	Average time between the visits of w_i to $t_j.r$
$V_{i,j}(L)$	Probability that w_i visits $t_j.r$ in a time frame of length L
$\mathcal{A}_s$	All possible visit scenarios for time frame [s,T]
A_s^l	lth visit scenario in $\mathcal{A}_s$
$p(A_s^l)$	Probability of A_s^l occurring
M_s^l	Stable matching (SM) for scenario A_s^l
$\phi(w_i, t_j)$	Priority of worker-task pair w_i, t_j
$F_{i,j}$	Set of tasks w_i finds more favorable than t_j
$G_{i,j}$	Set of workers t_j finds more favorable than w_i
$P_s(i, j)$	Probability of w_i and t_j being matched in SMs for $\mathcal{A}_s$

III. Problem Formulation

A. System Model

At the center of our system model is a service provider (SP) that receives the sensing task inquiries from different requesters and assigns them to appropriate participants. Formally, in each (hourly, daily, weekly) assignment period that is divided into discretized time-steps $(0, 1, .., T)$, the responsibility of SP is to assign a set of sensing tasks $\mathcal{T} = \{t_1, t_2, \ldots, t_m\}$ to a set of workers registered to the system $\mathcal{W} = \{w_1, w_2, \ldots, w_n\}$ in a way that will satisfy both parties (user satisfaction criteria will be described below).

Each task t has spatio-temporal constraints for successful completion. Let $t.r$ and $[t.b, t.d]$ denote the geographic region and the time frame (between the **b**eginning time and **d**eadline) in which task t should be performed, respectively. Tasks are assumed to require simple sensing activity such as taking pictures [36], recording noise levels [37], and reporting traffic volume [21] or crowdedness [20]. Thus, they take a few seconds to complete (thus neglected for simplicity), and they can be completed anytime during the specified time frame. Each task t is also associated with a monetary reward $m(t)$ that is paid to the worker who performs it by the task requester upon successful delivery of the sensed data.

The workers in our system model perform opportunistic sensing, so they do not travel to the task locations by interrupting their own schedules. Instead, they get assigned to a task only when they happen to be in the task region. Therefore, there is no travel cost associated with the tasks. Each worker w has a capacity $c(w)$, which indicates the maximum number of tasks worker w is willing to perform in a single assignment period. This can be a necessary constraint if the tasks require a certain level of involvement, causing the workers to lose some time. However, we also investigate task assignments in MCS systems that do not require involvement of workers and hence have no capacity constraints. Each worker w also has a quality score $q(w)$, which may refer to the likelihood of completing the assigned tasks [38], the expected quality of the sensed data [8], or the trustworthiness of the worker [39]. Some real-world mobile crowdsensing/sourcing systems that use a single numerical value to specify the qualification of workers include Waze [21] and Uber [40], which, respectively, utilize what is called Waze points and a five-star quality rating system.

In order to simplify our analysis, we rearrange the worker and task sets in decreasing order of the quality scores of workers and rewards of tasks, respectively. Thus, hereafter we have $q(w_i) > q(w_{i+1})$, $1 \leqslant i < n$, and $m(t_j) > m(t_{j+1})$, $1 \leqslant j < m$. If there are ties, we assume they are either broken by secondary factors such as registration time, or in an arbitrary manner so that there is only one possible order for both sets.

We assume that the trajectories of workers are uncertain but predictable, and unfold in real time during the assignment period, so the task assignments have to be made in an online manner. Let $\lambda_{i,j}$ be the average inter-visit time of worker w_i to the region $t_j.r$. Then, assuming an exponential distribution (similar to [13], [26]–[28]), the probability that worker w_i visits $t_j.r$ in a time frame of length L is computed as follows:

$$V_{i,j}(L) = 1 - e^{-L/\lambda_{i,j}} \tag{1}$$

The results of this study, however, do not depend on the underlying distribution model, and other probability functions including those produced by machine learning methods [41], which can integrate any dependency between the visits of a worker to different regions, can be used as well.

Once the task set for the current assignment period is determined, each worker w_i will be asked to provide SP with $\lambda_{i,j}$ values for the region of each task $t_j \in \mathcal{T}$. To this end, workers should be maintaining their visit records with a sufficient geographic density, as task regions may differ between assignment periods. They can submit arbitrarily large numbers for regions they have not visited, or for which they do not feel comfortable disclosing their true visit frequency for privacy-related reasons. Also, they can always inform SP of the regions they will definitely visit if certain parts of their trajectories are (or become) fixed. A legitimate concern here would be the possibility of receiving fabricated $\lambda_{i,j}$ values from some workers aiming to increase their gains from the system in a malicious manner. However, workers are required to inform SP when they enter one of the task regions to be considered for the assignment of the corresponding task, as a task may be assigned to a worker only when the worker is in the task region. Thus, SP can easily verify the accuracy of the received $\lambda_{i,j}$ values based on the visit frequencies of worker w_i, and reduce the quality scores of dishonest workers. We lastly note that in addition to the inter-visit times, the following information is assumed to be available at the beginning of the assignment period: $q(w)$ and $c(w)$ for each worker w, and $m(t)$, $t.b$, $t.d$ and $t.r$ for each task t.

B. Problem Definition

We represent the task assignments in our model with a matching $\mathcal{M}$ between the sets $\mathcal{W}$ and $\mathcal{T}$, where $\mathcal{M}(w)$ and $\mathcal{M}(t)$ denote the set of tasks assigned to worker w and the worker assigned to task t, respectively. If user (worker or task)

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3118134, IEEE Internet of Things Journal

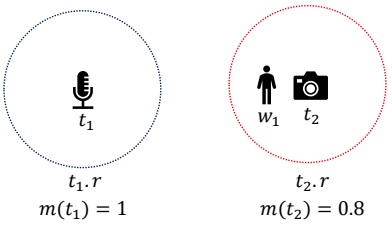


Fig. 2: An instance with a worker and two tasks. Let the probability of w_1 visiting $t_1.r$ before the deadline of t_1 be 0.6, and the probability of w_1 revisiting $t_2.r$ before the deadline of t_2 be 0. Also, let $q(w_1) = c(w_1) = 1$.

u is unassigned in $\mathcal{M}$, then $\mathcal{M}(u) = \varnothing$. For a matching to be feasible according to our system model, it should satisfy the following constraints for each $w \in \mathcal{W}$ and $t \in \mathcal{T}$:

- $\mathcal{M}(t) \in \mathcal{W} \cup \{\varnothing\}$,
- $\mathcal{M}(w) \subseteq \mathcal{T}$,
- $|\mathcal{M}(w)| \leqslant c(w)$,
- $\mathcal{M}(t) = w \Leftrightarrow t \in \mathcal{M}(w)$.

We will use the following definitions to refer to the user happiness in a matching.

Definition 1 (Unhappy pair)**.** *A worker-task pair (w, t) is said to be an unhappy pair (UP) in a matching $\mathcal{M}$ if*

- *worker w has visited region $t.r$ between the time frame $[t.b, t.d]$,*
- *task t is either unmatched or matched to a worker w' with a smaller QoS score than worker w (i.e., prefers w to w'),*
- *worker w has unused capacity (i.e., $|\mathcal{M}(w)| < c(w)$), or the reward of at least one task t' in $\mathcal{M}(w)$ is smaller than that of task t (i.e., prefers t to t').*

From the perspective of worker w and task t, the first condition in the definition indicates that there was in fact an opportunity for them to get matched, and the last two indicate that SP instead matched them with some other users that they prefer less, or left them unmatched/with an unused capacity.

Definition 2 (Stable matching)**.** *A matching $\mathcal{M}$ is stable if it contains no unhappy pairs.*

Although we can always find a stable matching (SM) in an offline setting (as it will be shown in the next section), it may not be possible to do so in an online setting where we do not know whether and when a worker will visit a region. Consider the instance in Fig. 2. Given that worker w_1 is currently in $t_2.r$, SP should decide whether to assign him to task t_2.

- If it assigns worker w_1 to task t_2, but then worker w_1 visits $t_1.r$, (w_1, t_1) will be an unhappy pair because worker w_1 prefers task t_1 to task t_2 as $m(t_1) > m(t_2)$, and task t_1 prefers being matched to worker w_1 to being unmatched.
- If it does not assign worker w_1 to task t_2, and worker w_1 does not visit $t_1.r$, then (w_1, t_2) will be an unhappy pair because worker w_1 and task t_2 prefer being matched to each other to being unmatched.

Thus, it is not possible to ensure perfect user happiness without knowing the exact worker trajectories. Besides, in an online setting, minimizing the *expected* number of unhappy pairs may not actually maximize user happiness either. Again, in the instance in Fig. 2, SP should avoid matching worker w_1 to task t_2 in order to minimize the expected number of unhappy pairs, because as shown in Table II, the expected number of unhappy pairs is larger when they get matched ($w_1 \Rightarrow t_2$). Yet the expected profit of worker w_1 in case he is not matched to task t_2 is $m(t_1) \times 0.6 = 0.6$, which is smaller than the profit he would make if he was matched to t_2 ($m(t_2) = 0.8$). So, worker w_1 would prefer to be matched to task t_2 despite the increase in the expected number of unhappy pairs he will form.

Decision	Scenario	# of UPs	Expected # of UPs
$w_1 \Rightarrow t_2$	w_1 visits $t_1.r$	1	0.6
	otherwise	0	
$w_1 \nRightarrow t_2$	w_1 visits $t_1.r$	0	0.4
	otherwise	1	

TABLE II: Analysis of all possible scenarios in the instance illustrated in Fig. 2. (UP is short for unhappy pair.)

The example discussed above demonstrates that, in an online setting, user happiness should be measured in an online manner and by considering the impact of each matching-related decision of SP on the overall benefit that users will get from the system. In MCS systems without capacity constraints, there is no competition among task requesters, because workers would like to and can get matched with all tasks on their trajectory. Therefore, in such systems, the stability of task assignments (or preference-awareness) can be ensured by maximizing the expected assignment quality of each task based on the visit probabilities of the workers.

However, in the presence of capacity constraints, there is a competition among both workers and task requesters, because the fact that workers are able to perform only a limited number of tasks transforms the task assignment problem into a limited resource allocation problem. In this setting, the expected utilities of users become interdependent, and get affected by each matching decision of SP. Consequently, the expected utility of a user after a certain time-step depends on the decision mechanism that will be used by SP in that time frame. Besides, the number of possible visit scenarios increases exponentially with respect to the length of the assignment period and the number of users. To address these challenges, in this paper, we map all possible (exponentially many) visit scenarios that can happen after time-step s to all possible stable matchings in these scenarios along with their likelihood of occurrence, which we can analyze in polynomial time and use to estimate the expected user utilities for the time period $[s, T]$.

Suppose that a worker w_i with a remaining capacity of $c_i^s \geqslant 1$ is in the region of a currently unassigned task t_j at time-step $s : t_j.b \leqslant s \leqslant t_j.d$, so SP has to make a matching decision for the pair. Let $\mathcal{A}_s = \{A_s^1, A_s^2, \ldots, A_s^k\}$ be the set of all possible worker visit scenarios for the time frame $[s, T]$ given the visit probabilities of the workers for all task regions. That is, $\mathcal{A}_s = R_1^s \times R_2^s \times .. \times R_n^s$, where R_i^s is the set of all possible spatio-temporal trajectories of worker w_i after time-step s. Let $p(A_s^l)$ denote the probability that $A_s^l \in \mathcal{A}_s$ will occur. Then, we have $\sum_{l=1}^{k} p(A_s^l) = 1$. Let M_s^l be the stable matching in the scenario A_s^l between the tasks that are unassigned and

the workers that have a positive remaining capacity at time-step s (since the visits in A_s^l are known, a stable matching can be found using the offline stable matching algorithm that will be described in Section IV-B). Also, let $\hat{M}_s^l$ be the stable matching in the same scenario assuming that w_i is matched to t_j. Then, assuming SP is making optimal assignments in terms of stability, the expected total reward worker w_i would get in time frame $[s, T]$ if he was not assigned to task t_j at time-step s can be computed by:

$$\mathbf{W}_i(s) = \sum_{l=1}^{k} \left(p(A_s^l) \times \sum_{t \in M_s^l(w_i)} m(t) \right), \tag{2}$$

and that if he was assigned to task t_j by:

$$\mathbf{W}'_{i,j}(s) = m(t_j) + \sum_{l=1}^{k} \left(p(A_s^l) \times \sum_{\hat{t} \in \hat{M}_s^l(w_i)} m(\hat{t}) \right). \tag{3}$$

Analogously, the expected sensing quality to be received by task t_j if it is not assigned to worker w_i at time-step s and otherwise can be, respectively, computed by:

$$\mathbf{T}_j(s) = \sum_{l=1}^{k} \left(p(A_s^l) \times q(M_s^l(t_j)) \right), \tag{4}$$

and

$$\mathbf{T}'_{j,i}(s) = q(w_i). \tag{5}$$

Then, we can define a decision-time unhappy pair as follows.

Definition 3 (Decision-time unhappy pair)**.** *A worker-task pair (w_i, t_j) is said to be a decision-time unhappy pair if the following conditions hold for any time-step s in $[t_j.b, t_j.d]$:*

- *worker w_i has a positive remaining capacity,*
- *task t_j is unassigned,*
- *worker w_i is in region $t_j.r$, and*
- *either (i) SP matches worker w_i to task t_j, but at least one of them would be better off otherwise, i.e.,*

$$\mathbf{W}_i(s) > \mathbf{W}'_{i,j}(s) \quad or \quad \mathbf{T}_j(s) > \mathbf{T}'_{j,i}(s), \tag{6}$$

- *or (ii) SP does not match worker w_i to task t_j, but they both would be better off otherwise, i.e.,*

$$\mathbf{W}'_{i,j}(s) > \mathbf{W}_i(s) \quad and \quad \mathbf{T}'_{j,i}(s) > \mathbf{T}_j(s). \tag{7}$$

In our example illustrated in Fig. 2, assuming s is the current time-step, we have two possible trajectories that can be seen after s (i.e., w_1 visits $t_1.r$ or he does not; $|\mathcal{A}_s| = 2$) with the given probabilities. This yields $\mathbf{W}'_{1,2}(s) = 0.8 > \mathbf{W}_1(s) = 0.6$ and $\mathbf{T}'_{2,1}(s) = 1 > \mathbf{T}_2(s) = 0$. Hence, worker w_1 and task t_2 will, as desired, form a decision-time unhappy pair due to (7) if SP fails to match them.

Definition 4 (Online stable matching)**.** *A matching $\mathcal{M}$ is called an online stable matching if it does not admit any decision-time unhappy pairs.*

Consequently, our objective in the MCS systems with capacity constraints is to find an online stable matching, and we call such a matching *optimal* in terms of preference-awareness. It is straightforward to see that the optimal matching strategy to this end would be to match a worker-task pair if (7) holds. However, the difficult part is to compute the values of $\mathbf{W}_i(s)$, $\mathbf{W}'_{i,j}(s)$ and $\mathbf{T}_j(s)$, because $\mathcal{A}_s$ grows exponentially with the number of users and length of the assignment period (T). In the following section, we will show how to compute these values efficiently without actually forming the set $\mathcal{A}_s$. The key notations used throughout the paper are summarized in Table I for convenience.

IV. Proposed Solution

A. Task Assignment without Worker Capacity Constraints

In MCS systems with simple sensing tasks that do not require an active involvement of workers, workers can be assumed to be able to perform all of the tasks on their trajectories (i.e., have no capacity constraint). Moreover, in the case of uniformly distributed tasks in an area or short assignment periods, since workers could visit only a limited number of task regions, it would still be safe to disregard the capacity constraints. In other words, even if workers had capacity constraints in any of these cases, they would be overshadowed by the spatio-temporal constraints and can hence be ignored during the task assignment process (at least until the point where assigning another task to a worker would violate his capacity constraint). An example of an MCS system without capacity constraints would be a traffic monitoring system such as Waze [21], where the speed of traffic, which can be estimated by the speed of change in the GPS coordinates of workers, can be sensed and transmitted to SP automatically by workers' mobile devices without requiring active involvement of workers.

In this type of MCS systems, workers would like to perform each and every task that is on their trajectory and does not conflict with their preferences in order to maximize their profits. However, task requesters would still desire to have their sensing tasks performed by workers with the highest quality scores. Thus, the problem transforms into a one-sided matching problem in terms of user preferences. That is, to find optimal task assignments we just need to maximize the sensing quality received by task requesters. Moreover, we can consider each task separately, because the assignment quality of a task t_j depends only on which workers will visit the task region $t_j.r$ and the time of their visits, and is independent of the visits of workers to the other task regions due to the absence of capacity constraints. To solve this problem, we utilize *Optimal Stopping Theory (OST)* [42], which provides a dynamic programming-based framework for the decision problems with a finite horizon (e.g., the secretary hiring problem). This is suitable for our problem, because for each task t_j there will be a number of decision points at the times $t_j.r$ is visited by any worker, and at each of these we should decide whether to wait for a higher quality worker or to assign task t_j to the worker w_i who is currently in the region $t_j.r$ based on the quality of w_i and the expected quality to be achieved if we choose to wait instead.

Let $\mathbf{E}_j(s)$ be the expected assignment quality for task t_j after the time-step s. Since task t_j can only be performed

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3118134, IEEE Internet of Things Journal

Algorithm 1: Calculation of $\mathbf{E}_j(s)$ for all practical values of time-step s

1 $\mathbf{E}_j(t_j.d) \leftarrow 0$
2 **for** $s \leftarrow t_j.d - 1$ *down to* $t_j.b$ **do**
3 $\mathbf{E}_j(s) \leftarrow q(w_1) \times V_{1,j}(1)$
4 $\rho \leftarrow 1 - V_{1,i}(1)$
5 **for** $i \leftarrow 2$ *to* n **do**
6 **if** $q(w_i) \geqslant \mathbf{E}_j(s+1)$ **then**
7 $\mathbf{E}_j(s) \leftarrow \mathbf{E}_j(s) + q(w_i) \times V_{i,j}(1) \times \rho$
8 $\rho \leftarrow \rho \times (1 - V_{i,j}(1))$
9 **else**
10 **break**
11 $\mathbf{E}_j(s) \leftarrow \mathbf{E}_j(s) + \mathbf{E}_j(s+1) \times \rho$

between $[t_j.b, t_j.d]$, we have

$$\mathbf{E}_j(s) = \mathbf{E}_j(t_j.b),\ s < t_j.b, \tag{8}$$

and

$$\mathbf{E}_j(s) = 0,\ s \geqslant t_j.d. \tag{9}$$

Since each worker w_i will visit the region $t_j.r$ in the time frame $[s, s+1)$ with the probability $V_{i,j}(1)$, we have the following recursive relation between $\mathbf{E}_j(s)$ and $\mathbf{E}_j(s+1)$:

$$\mathbf{E}_j(s) = \sum_{i=1}^{n} \Big(\max\big(q(w_i), \mathbf{E}_j(s+1)\big) \times V_{i,j}(1) \times \rho_j(i) \Big) + \mathbf{E}_j(s+1) \times \rho_j(n+1), \tag{10}$$

where $\rho_j(i)$ is the probability that no worker with an index smaller than i visits $t_j.r$ within a time frame of length 1. Since the smallest worker index is 1, we have $\rho_j(1) = 1$, and the value of $\rho_j(i)$ for $2 \leqslant i \leqslant n+1$ can be computed by:

$$\rho_j(i) = \rho_j(i-1) \times \big(1 - V_{i-1,j}(1)\big) \tag{11}$$

Note that although there are 2^n possible scenarios (i.e., each worker being within or outside of the task region) for each time frame of length 1 in terms of worker visits to a task region, we consider only n of them to calculate (10), because if w_i is in the region, whether $w_{i+1}, .., w_n$ are within or outside of the region is irrelevant as they are preferred less than w_i. Therefore, using the base cases $\mathbf{E}_j(t.d) = 0$ and $\rho_j(1) = 1$, we can recursively compute all values of $\mathbf{E}_j(s)$ for $t_j.b \leqslant s < t_j.d$ in polynomial time as described in Algorithm 1.

In this algorithm, when we calculate $\mathbf{E}_j(s)$, we utilize the fact that task t_j would like to match only with workers with a quality score that is greater than or equal to $\mathbf{E}_j(s+1)$ (line 6) at time-step s because, otherwise, it would be more advantageous for it to wait for the next time-step. Thus, we consider only these workers in lines 3-10 in decreasing order of their quality scores to compute the expected utility of task t_j based on the visit probabilities of these workers to its region in case it will be matched with one of these workers at time-step s. Since the expected utility of task t_j at time-step s will be the same as that at time-step $s+1$ if none of these workers visits the region of task t_j between time-steps s and $s+1$, we finally increase the value of $\mathbf{E}_j(s)$ by $\mathbf{E}_j(s+1) \times \rho$ in line 11, where ρ is calculated between lines 4-10 as the probability that $t_j.r$ will not be visited by any of these workers between time-steps s and $s+1$.

Algorithm 2: OST-based Algorithm (OSTA) at time-step s

1 **if** $q(w_i) \geqslant \mathbf{E}_j(s)$ **then**
2 match w_i to t_j
3 terminate the algorithm for t_j

A summary of the optimal decision mechanism that will be run for each task t_j whenever $t_j.r$ is visited by a worker w_i is given in Algorithm 2. We assume all $\mathbf{E}_j(s)$ values for $s : t_j.b \leqslant s \leqslant t_j.d$ are precomputed and stored in a lookup table, but it is also possible to compute only $\mathbf{E}_j(s)$ for $s : \hat{s} \leqslant s \leqslant t_j.d$ at the first time ($\hat{s}$) a worker visits $t_j.r$ to avoid computing $\mathbf{E}_j(s)$ values that will never be used. The algorithm simply checks whether it is more advantageous to match with the visiting worker or to skip the opportunity (line 1), and makes a matching decision accordingly. Due to sparse nature of visits in mobile networks, we assume that there will be a single matching decision to make at each time-step. However, if there are multiple workers that visit the region of a task at a certain time-step, it suffices to run Algorithm 2 for the worker with the highest quality score. For each task t_j, the algorithm will be run until either task t_j gets matched, or it expires. Since Algorithm 2 assigns a task t_j to a worker w_i *at* time-step s only if w_i provides a higher QoS than the expected QoS that t_j will obtain *after* time-step s according to the visit probabilities of workers, and its each matching decision between time-steps $t.b$ and $t.d$ accordingly maximizes the expected value of the QoS to be received in the end by the requester of t_j, we have the following result.

Corollary 1. *Algorithm 2 always makes the optimal matching decisions for task requesters when workers do not have capacity constraints (i.e., maximizes the expected assignment quality $q(\mathcal{M}(t))$ for each task t).*

Running time. Algorithm 2 obviously has a time complexity of $O(1)$, however $\mathbf{E}_j(s)$ needs to be precomputed for all feasible j and s values by running Algorithm 1. For each task t_j, we precompute $\mathbf{E}_j(s)$ values for all $s : t_j.b \leqslant s \leqslant t_j.d$, and computing each $\mathbf{E}_j(s)$ value takes $O(n)$ time. Thus, the total time complexity becomes $O(n\tau)$, where $\tau = \sum_{t \in \mathcal{T}} (t.d - t.b)$.

B. Generic Task Assignment with Capacity Constraints

In this section, we first describe an optimal algorithm to find stable matchings in offline settings where the trajectory of each worker is known in advance. Then, exploiting the ideas behind the offline algorithm, we provide our algorithm for the online settings. We begin with the following definition.

Definition 5 (Pair priority)**.** *The priority $\phi(w_i, t_j)$ of a worker-task pair (w_i, t_j) refers to the relative importance of the pair in terms of stability, and can be defined as*

$$\phi(w_i, t_j) = \max(m, n) \times \min(i, j) + \max(i, j) \tag{12}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3118134, IEEE Internet of Things Journal

where a smaller value indicates a higher priority.

Note that the pair priority values enable us to specify user preferences in pairs. That is, the priority value $\phi(w_i, t_j)$ of a worker-task pair (w_i, t_j) tells us that

- w_i prefers t_j to t_k, $\forall t_k : \phi(w_i, t_j) < \phi(w_i, t_k)$, and
- t_j prefers w_i to w_k, $\forall w_k : \phi(w_i, t_j) < \phi(w_k, t_j)$.

For convenience, we let $F_{i,j} = \{t_k : \phi(w_i, t_k) < \phi(w_i, t_j)\}$ denote the set of tasks w_i finds more favorable than t_j, and $G_{i,j} = \{w_k : \phi(w_k, t_j) < \phi(w_i, t_j)\}$ denote the set of workers t_j finds more favorable than w_i.

1) Offline algorithm: In Algorithm 3, we present a pseudo-code description of the offline algorithm. In line 1, it finds the set $\mathcal{A}$ of all eligible worker-task pairs that can be matched to each other (i.e., the task region visited by the worker). Then, in each step, it finds (line 3) and matches (lines 4-5) the worker-task pair with the highest priority in $\mathcal{A}$, which is followed by removing all pairs that become infeasible due to the most recent pair assignment (lines 6-8). This continues until the set $\mathcal{A}$ becomes empty. In the following theorem, we prove the optimality of this algorithm.

Theorem 1. *Algorithm 3 always produces a stable matching in offline settings.*

Proof. We prove this by contradiction. Assume that there is an unhappy pair (w_i, t_j) in the final matching $\mathcal{M}$ produced by the algorithm. According to Definition 1, worker w_i has visited $t_j.r$ within the time frame of task t_j, so (w_i, t_j) is in $\mathcal{A}$ in the beginning.

- If $|\mathcal{M}(w_i)| < c(w_i)$ and $\mathcal{M}(t_j) = \varnothing$, then the pair (w_i, t_j) should still be in $\mathcal{A}$, which indicates that $\mathcal{A}$ is non-empty, contradicting the termination condition of the algorithm.
- If $|\mathcal{M}(w_i)| < c(w_i)$ and $\mathcal{M}(t_j) = w_k$, then for (w_i, t_j) to be an unhappy pair, we should have $q(w_i) > q(w_k)$, hence
$$i < k \text{ and } \phi(w_i, t_j) < \phi(w_k, t_j). \tag{13}$$
Since at the time the pair (w_k, t_j) was selected by the algorithm, the pair (w_i, t_j) was still in $\mathcal{A}$ (as $|\mathcal{M}(w_i)| < c(w_i)$) and has a higher priority than (w_k, t_j), the algorithm should have selected (w_i, t_j), which is a contradiction.
- If $|\mathcal{M}(w_i)| = c(w_i)$ and $\mathcal{M}(t_j) = \varnothing$, then for (w_i, t_j) to be an unhappy pair, we should have $m(t_k) < m(t_j)$ for at least one task $t_k \in \mathcal{M}(w_i)$. Thus,
$$j < k \text{ and } \phi(w_i, t_j) < \phi(w_i, t_k). \tag{14}$$
As in the previous scenario, this indicates that the pair (w_i, t_j) should have been selected prior to (w_i, t_k), which is a contradiction.
- If $|\mathcal{M}(w_i)| = c(w_i)$ and $\mathcal{M}(t_j) = w_k$, then for (w_i, t_j) to be an unhappy pair, we should have
$$q(w_i) > q(w_k) \text{ and } m(t_j) > m(t_l) \tag{15}$$
for at least one task $t_l \in \mathcal{M}(w_i)$. This means $i < k$ and $j < l$, and hence
$$\phi(w_i, t_j) < \phi(w_k, t_j) \text{ and } \phi(w_i, t_j) < \phi(w_i, t_l). \tag{16}$$
Then, the pair (w_i, t_j) should have been selected prior to both (w_i, t_l) and (w_k, t_j), which is also a contradiction and completes the proof. □

Algorithm 3: Offline SM Algorithm

```
1 A ← {(w,t) : w visits t.r in [t.b, t.d]}
2 while A ≠ ∅ do
3     (w_i, t_j) ← arg min_{(w,t)∈A} φ(w,t)
4     M(w_i) ← M(w_i) ∪ t_j
5     M(t_j) = w_i
6     if |M(w_i)| = c(w_i) then
7         A ← A\{(w,t) : w = w_i}
8     A ← A\{(w,t) : t = t_j}
9 return M
```

2) Online algorithm: Theorem 1 shows that in the presence of capacity constraints, a worker-task pair dominates the pairs with lower priority scores if it is in the set $\mathcal{A}$, which implies that the highest priority pair will be matched with the same probability of being in the set $\mathcal{A}$. Likewise, the next highest priority pair will be matched with the probability of being in the set $\mathcal{A}$ in case it is not eliminated by the higher priority pair, and so on. We utilize this observation to find the matching probability of a worker-task pair in the stable matchings for all possible scenarios ($\mathcal{A}_s$) that can occur after a certain time-step (s), and use it to make decisions in online setting.

Lemma 1. *Given a worker-task pair (w_i, t_j) and time-step s at which w_i has a remaining capacity of c_i^s and t_j is unmatched, the probability $P_s(i,j)$ of w_i and t_j being matched in a stable matching in any of the possible scenarios that can occur between time-steps $s < t_j.d$ and T can be computed by*

$$P_s(i,j) = \sum_{k=1}^{c_i^s} Q_{i,j}^s[k] \times \overbrace{\prod_{k=1}^{i-1} \big(1 - P_s(k,j)\big)}^{\eta_{i,j}^s} \times V_{i,j}(t_j.d - s), \tag{17}$$

where $\eta_{i,j}^s$ denotes the probability that t_j will not be matched with any worker in $G_{i,j}$ times the probability that w_i visits $t_j.r$ until $t_j.d$, and $Q_{i,j}^s[k]$ denotes the probability that w_i will be matched to exactly $c_i^s - k$ of the tasks in $F_{i,j}$, and can be computed by

$$Q_{i,j}^s[k] = \begin{cases} 1, & \text{if } j = 1,\ k = c_i^s \\ 0, & \text{if } j = 1,\ k \neq c_i^s \\ Q_{i,j-1}^s[k] + Q_{i,j-1}^s[k+1] \times \eta_{i,j-1}^s, & \text{if } j > 1,\ k = 0 \\ Q_{i,j-1}^s[k] \times (1 - \eta_{i,j-1}^s), & \text{if } j > 1,\ k = c_i^s \\ Q_{i,j-1}^s[k+1] \times \eta_{i,j-1}^s + Q_{i,j-1}^s[k] \times (1 - \eta_{i,j-1}^s). & \text{otherwise} \end{cases} \tag{18}$$

Proof. In order for worker w_i and task t_j to be matched in a stable matching $\mathcal{M}$ in a given scenario (e.g., $A_s^l \in \mathcal{A}_s$), the following three conditions should be satisfied:

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3118134, IEEE Internet of Things Journal

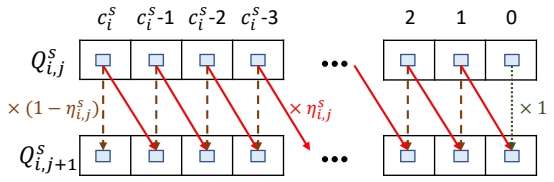


Fig. 3: Illustration of the update procedure for the remaining capacity probabilities defined in (18). Dashed and solid edges indicate a contribution with a factor of $1-\eta_{i,j}^s$ and $\eta_{i,j}^s$, respectively, while the dotted edge indicates a direct addition.

- at most c_i^s-1 of the higher priority pairs in the set $\{(w_i,t_k): t_k \in F_{i,j}\} = \{(w_i,t_k): k<j\}$ should be matched in $\mathcal{M}$ (i.e., w_i should be matched with at most c_i^s-1 of the tasks that he prefers more than t_j), because, otherwise, the pair (w_i,t_j) will be eliminated. In (17), the probability of this is given by
$$\sum_{k=1}^{c_i^s} Q_{i,j}^s[k]. \tag{19}$$
In other words, (19) is the probability that worker w_i will have a positive remaining capacity and be able to match with task t_j. The calculation of the $Q_{i,j}^s[k]$ values is realized in a recursive manner as described in (18). Since $F_{i,1}=\varnothing$, we initially have $Q_{i,1}^s[c_i^s]=1$ and $Q_{i,1}^s[k<c_i^s]=0$. We can then compute $Q_{i,j+1}^s$ from $Q_{i,j}^s$ using the recursive procedure illustrated in Fig. 3.
- none of the higher priority pairs in $\{(w_k,t_j): w_k \in G_{i,j}\} = \{(w_k,t_j): k<i\}$ should be matched in $\mathcal{M}$, because, otherwise, task t_j will already be matched with a more favorable worker, hence the pair (w_i,t_j) will be eliminated. In (17), this is given in a recursive fashion as follows:
$$\prod_{k=1}^{i-1} 1-P_s(k,j). \tag{20}$$
- worker w_i should visit the region of task t_j between $[s,t_j.d]$. That is, the pair (w_i,t_j) should be in the set $\mathcal{A}$ of Algorithm 3. This occurs with the probability of $V_{i,j}(t_j.d-s)$, which is the last term in (17). □

Algorithm 4 summarizes the procedure to calculate $P_s(i,j)$ values for all $1\leqslant i\leqslant n, 1\leqslant j\leqslant m$ values. In this algorithm, we maintain a variable u_j for each task t_j, which refers to the probability of task t_j not being matched to any of the workers that considered so far in the algorithm, hence initialized to be 0 in line 1 if t_j is already matched before time-step s, and 1 otherwise. Note that once $P_s(i,j)$ is calculated, the values of $P_s(k,j)$ for $k>i$ and $P_s(i,l)$ for $l>j$ are independent of each other. Thus, we can first compute $P_s(1,j)$ starting from $j=1$ to $j=m$, then $P_s(2,j)$ for all j values in the same order, and so on. This ensures that the matching probabilities of all interdependent worker-task pairs will be calculated following the priority order.

According to Lemma 1, we can express $P_s(i,j)$ as the ratio of the number of stable matchings that worker w_i and task t_j are matched to each other to the total number of stable matchings in all possible scenarios after time-step s. Thus, given the $P_s(i,j)$ values for all i,j pairs, we can compute $\mathbf{W}_i(s)$ and $\mathbf{T}_j(s)$ as follows:

$$\mathbf{W}_i(s) = \sum_{k=1}^{m} P_s(i,k)\times m(t_k), \tag{21}$$

$$\mathbf{T}_j(s) = \sum_{k=1}^{n} P_s(k,j)\times q(w_k). \tag{22}$$

On the other hand, the value of $\mathbf{W}'_{i,j}(s)$ depends on the probability of worker w_i being matched with each task t_k in the stable matchings that can be seen after time-step s assuming worker w_i and task t_j will be matched at s. This probability is denoted by $\hat{P}_s(i,k)$, and can simply be calculated by assuming task t_j is already matched and replacing c_i^s with c_i^s-1 in (17) and (18) (and running Algorithm 4 accordingly). Then, we can compute $\mathbf{W}'_{i,j}(s)$ as:

$$\mathbf{W}'_{i,j}(s) = m(t_j) + \sum_{k\in\{1..m\}\setminus\{j\}} \big(\hat{P}_s(i,k)\times m(t_k)\big). \tag{23}$$

Lastly, we have $\mathbf{T}'_{j,i}(s)=q(w_i)$. Using these values, we can make an optimal matching decision for the worker-task pair (w_i,t_j) in terms of online stable matchings at any time-step s worker w_i is in the region of task t_j.

A summary of the decision process is described in Algorithm 5. In line 1, we compute the matching probabilities for all worker-task pairs by calling Algorithm 4, and then, based on these probabilities, we compute the expected utilities of worker w_i and task t_j at time-step s for the scenarios they do and do not get matched with each other in lines 2-3. If getting matched with each other is more preferable (line 4) for both worker w_i and task t_j by a constant factor α (which will be discussed below), then a positive matching decision is

Algorithm 4: Calculation of $P_s(i,j)$ for all i,j

```
1 for j ← 1 to m do u_j ← 1 − |M(t_j)|
2 for i ← 1 to n do
3     compute Q^s_{i,1} according to (18)
4     for j ← 1 to m do
5         vp ← V_{i,j}(t_j.d − s)
6         η^s_{i,j} ← vp × u_j
7         P_s(i,j) ← η^s_{i,j} × Σ_{k=1}^{c_i^s} Q^s_{i,j}[k]
8         compute Q^s_{i,j+1} from Q^s_{i,j} according to (18)
9         u_j ← u_j − P_s(i,j)
```

Algorithm 5: PRobabilistic Stable Task Assignment (PRSTA)$_\alpha(w_i,t_j)$ at time-step s

```
1 Compute P_s(k,l) and P̂_s(k,l), ∀k,l, via Algorithm 4
2 Compute W_i(s), T_j(s), W'_{i,j}(s) according to (21), (22), (23), respectively
3 T'_{j,i}(s) ← q(w_i)
4 if W'_{i,j}(s) > αW_i(s) and T'_{j,i}(s) > αT_j(s) then
5     match w_i to t_j
```

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3118134, IEEE Internet of Things Journal

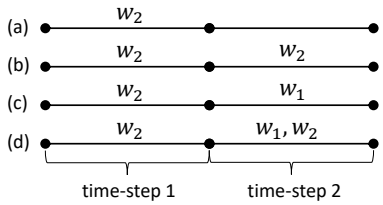


Fig. 4: Four of the possible visit orders of two workers (w_1, w_2) to the region of task t in an MCS instance with an assignment period of length two. For example, in scenario (c), the task region is visited by w_2 in time-step 1, and by w_1 in time-step 2.

made in line 5. As earlier, we assume at most one matching decision is being made at each time-step, but multiple worker-task pairs can be processed in the order of pair priority if needed. Given Definition 3 & 4, since the proposed algorithm makes a positive matching decision for a worker-task pair if (7) holds when $\alpha = 1$, we have the following result.

Corollary 2. *PRSTA*$_{1.0}$ *algorithm always produces online stable matchings.*

Note that the proposed method to compute expected user utilities does not consider the order of visits of workers to the task regions. Let us consider the instance given in Fig. 4 to explain this issue and why it is necessary to incorporate a constant α factor in the matching decisions of Algorithm 5 as a heuristic to address it. In this example, we assume that the visit probability of worker w_1 to the region of task t is quite high, and his quality score is significantly larger than that of worker w_2 so that even if w_2 visits the task region in time-step 1, it is advantageous for t to wait for w_1. Thus, in all four scenarios, the decision at time-step 1 when w_2 visits the task region should be not to assign him to the task.

However, when we compute the expected utilities of users considering all possible stable matchings and their probability of occurrence, w_2 should be assigned to t in scenarios (a) and (b) given that w_1 does not visit the task region in either time-step in these scenarios. Therefore, Algorithm 5 considers the matching (w_2, t) for scenarios (a) and (b) during computation of expected user utilities, and the matching (w_1, t) for the other scenarios. Yet when we see a similar visit pattern for time-step 1 in the online setting, since we do not in advance know the visit pattern for time-step 2, we need to either assign w_2 to t or not. Here, the utility of t will inevitably be overestimated, because if it gets assigned to w_2, its actual utility will also be $q(w_2)$ in scenarios (c) and (d), which is smaller than its expected utility $q(w_1)$ based on the stable matching (w_1, t) of these scenarios. On the other hand, if it does not get assigned to w_2, then it will be left unmatched in scenario (a), and its actual utility (0) will be worse than its expected utility $q(w_2)$ based on the stable matching (w_2, t) of this scenario. To alleviate the impact of these overestimations on the performance of the algorithm, we require (line 4 of Algorithm 5) that the expected utility of a user after the current time-step s is at least $1/\alpha$ times better than the utility he can get at time-step s to skip the existing matching opportunity. Note that using such a constant factor does not favor any groups of users in the system, and hence does not invalidate its preference-awareness, in general, as long as a single, universal α factor is used for all decisions to ensure fairness towards different users. We empirically examine the algorithm's performance with various α values in the next section.

Running time. The time complexity of Algorithm 4 is $O(mnc_{max})$, where $c_{max} = \max_{w \in \mathcal{W}} c(w)$. Since the most expensive operation in Algorithm 5 is to run Algorithm 4 to find the matching probabilities, the worst-case running time of Algorithm 5 is also $O(mnc_{max})$. This can also be expressed as $O(nm^2)$, as the largest feasible $c_{max} = m$.

V. Simulation Results

A. Settings

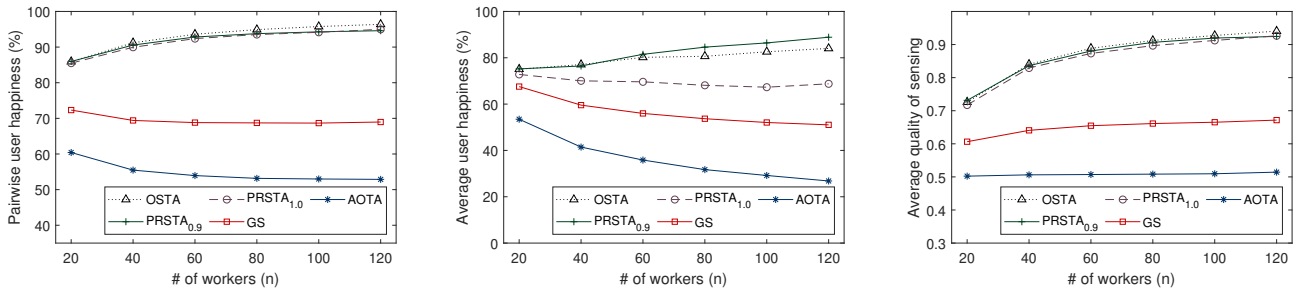
We perform simulations utilizing both a real data set and a synthetic data set. The synthetic data set is generated using 60 workers and 100 tasks in a 4 hours long assignment period. We randomly set the quality scores of the workers and the task rewards from the range $(0, 1)$, and assign a capacity to each worker between 1 and 10 (we also look at the case without worker capacity constraints). For each worker-task pair (w_i, t_j), we randomly set the value of $\lambda_{i,j}$ between 8 to 24 hours. This generates instances where each worker visits, on average, 23% of all task regions in an assignment period. We examine the performance of the algorithms in instances with different worker/task counts, capacity constraints, and inter-visit times as well.

For the real data set, we utilize the San Francisco taxi data set [43], which contains the traces of 536 yellow cabs during May of 2018. For each instance, we randomly select a day as the assignment period, and then pick 60 taxis and use their traces on that day as worker trajectories. We divide the SF city into 121×100 regions of approximately $10^2 \times 10^2$ square meters, and create a task on randomly selected 100 regions that have at least 1000 traces in the whole data set. The other parameters are assigned similarly with the synthetic data set, and for each worker-task pair (w_i, t_j), the value of $\lambda_{i,j}$ is extracted from the traces.

To avoid introducing arbitrary random values for parameters that do not affect the performance of the algorithms in a notable way, we let the time frame of each task be the same as the duration of the assignment period in both data sets. Also, the assignment period is divided into a minute long time-steps in both data sets. Following the procedures described above, we generate 100 different instances of both synthetic and real data sets, and present the averaged results.

We compare the performance of the proposed algorithms with the Average makespan sensitive Online Task Assignment (AOTA) algorithm [13] and the Gale-Shapley (*GS*) algorithm [15] by adapting them to our setting. The former aims to minimize the average task completion time, and is adapted to our setting as follows. Whenever a matching decision between a worker-task pair (w, t) needs to be made, we construct a weighted bipartite graph between workers and tasks, where the weight of the edge between (w, t) is 0, and the other edge weights (w', t') are the average inter-visit time

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3118134, IEEE Internet of Things Journal



(a) (b) (c)

Fig. 5: Performance of algorithms with varying worker counts in synthetic data set without capacity constraints ($m = 100$).

of workers (w') to task regions ($t'.r$). We then find a minimum weight matching M on this graph, and assign w to t if they are matched to each other in M. Note that in the setting without worker capacity constraints, whenever a worker is in the region of a yet uncompleted task, the AOTA algorithm greedily matches the worker to the task, as this is guaranteed to minimize the average task completion time.

The GS algorithm is normally used to find stable matchings when the preference lists of individuals are static and known in advance. In our setting, however, it is neither possible nor desirable to match a worker-task pair if the worker does not visit the task region even if they happen to prefer each other the most. Since worker visits are uncertain in our setting, user preferences change dynamically based on worker trajectories, thus the GS algorithm cannot be used directly. We adapt it to our setting as follows. When a matching decision needs to be made at a time-step s for a worker-task pair, we form the preference lists of all workers with a positive remaining capacity and all unmatched tasks based on how likely they will have a chance to match and how beneficial they are to each other. Specifically, the preference list of each worker w_i is formed as $t_{\sigma_1}, t_{\sigma_2}, .., t_{\sigma_k}$ in order of non-increasing preference such that $m(t_{\sigma_j}) \times V_{i,\sigma_j}(t_{\sigma_j}.d - s) \geqslant m(t_{\sigma_{j+1}}) \times V_{i,\sigma_{j+1}}(t_{\sigma_{j+1}}.d - s)$ for all $j : 1 \leqslant j < k$. The preference lists of tasks are formed similarly using the quality scores of the workers. Then, the GS algorithm is run to find a stable matching for these preference lists. If the currently examined worker-task pair is matched in this stable matching, we also match them in the real matching problem, otherwise we leave them unmatched for that time-step. For the PRSTA$_\alpha$ algorithm, we present the results for $\alpha = 1.0$ and $\alpha = 0.9$ in general as PRSTA$_{1.0}$ guarantees to produce online stable matchings, and PRSTA$_{0.9}$ is empirically shown to produce high quality final assignments with respect to the other performance metrics. However, we also examine the performance of the PRSTA$_\alpha$ algorithm with different values of α.

In order to evaluate and compare the performance of the algorithms, we utilize the following metrics, which capture different aspects of user satisfaction and efficiency.

- *Pairwise user happiness (PUH):* This is calculated as

$$100 \times \frac{b-a}{b}, \tag{24}$$

where a is the number of unhappy pairs, and b is the number of worker-task pairs (w, t) that had at least one matching opportunity during the assignment period, i.e., w visits $t.r$ between $[t.b, t.d]$, and at the time of the visit, w has a non-zero remaining capacity and t is unmatched.

- *Average user happiness (%):* Given a matching $\mathcal{M}$, let $\mathcal{S}_u$ be the set of tasks (workers) with whom worker (task) u forms an unhappy pair. Then, we can define the happiness ratio of user u as follows:

$$\theta_u = \begin{cases} 1, & \text{if } \mathcal{S}_u = \varnothing \\ 0, & \text{if } \mathcal{S}_u \neq \varnothing,\ \mathcal{M}(u) = \varnothing \\ \min\limits_{v \in \mathcal{S}_u} \left\{ \frac{\hat{f}(u)}{\hat{f}(v)} \right\}, & \text{otherwise} \end{cases} \tag{25}$$

where

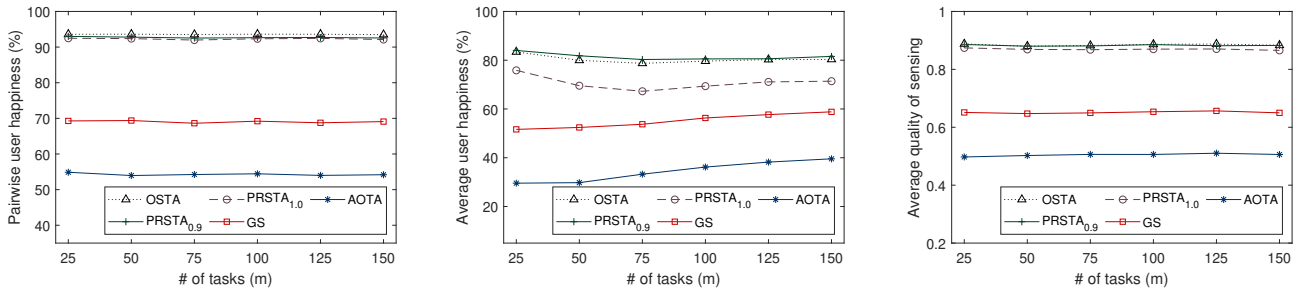
$$\hat{f}(u) = \begin{cases} q(\mathcal{M}(u)), & \text{if } u \in \mathcal{T} \\ \min\limits_{t \in \mathcal{M}(u)} \{m(t)\}, & \text{if } u \in \mathcal{W} \end{cases} \tag{26}$$

and $f(v) = m(v)$ if v is a task, and $f(v) = q(v)$ if it is a worker. Here, $\theta_u = 1$ if user u does not form any unhappy pairs, and $\theta_u = 0$ if he forms unhappy pairs and is unmatched (i.e., since his current utility is 0, he is infinitely unhappy). Otherwise, its happiness is computed as the ratio of his current utility to the maximum utility he could achieve if he was matched to one of the users in the unhappy pairs he forms. Accordingly, the average user happiness is computed by $100 \times \sum_{u \in \mathcal{W} \cup \mathcal{T}} \theta_u / U$, where $U = m + n$.

- *Average quality of sensing:* This is the average quality of sensing/service provided to the task requesters, and is computed by $\sum_{t \in \mathcal{T}} q(\mathcal{M}(t))/m$, where $q(\mathcal{M}(t)) = 0$ if $\mathcal{M}(t) = \varnothing$.
- *Online user happiness:* To show the optimality of the PRSTA$_{1.0}$ algorithm empirically, we look at the happiness of the users with the matching decisions in capacity-constrained settings. This is computed similarly to pairwise user happiness, but a and b in (24) are set, respectively, as the number of decision-time unhappy pairs and the number of times the algorithm is run to make a matching decision, which can be different for each algorithm.

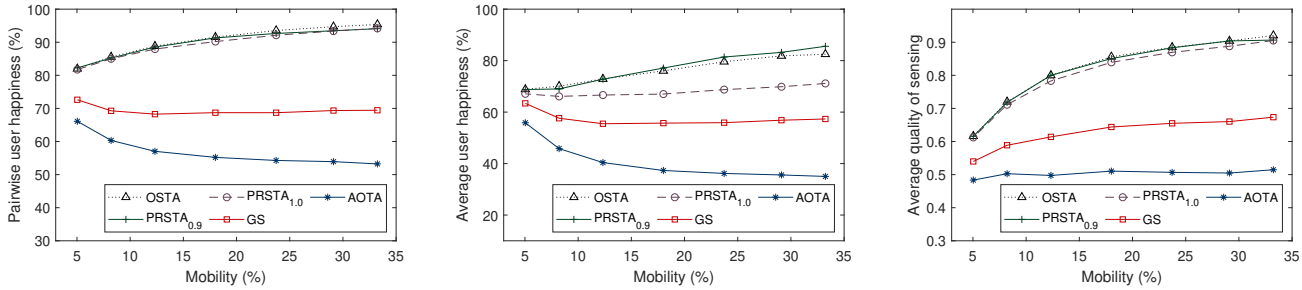
We also look at the running times of the algorithms to analyze how quickly they make the matching decisions.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3118134, IEEE Internet of Things Journal



(a) (b) (c)

Fig. 6: Performance of algorithms with varying task counts in synthetic data set without capacity constraints $(n = 60)$.



(a) (b) (c)

Fig. 7: Performance of algorithms with varying mobility rates in synthetic data set without capacity constraints $(m, n = 100, 60)$.

B. Results

1) Results without Worker Capacity Constraints: We first look at the results in the synthetic data set without capacity constraints. Fig. 5 shows the impact of the number of workers on the performance of the algorithms. We see that the proposed algorithms substantially outperform the others, and the OSTA algorithm has the best performance for the most part, as expected. In fact, it is only slightly outperformed by the $\text{PRSTA}_{0.9}$ algorithm in terms of average user happiness. This indicates that despite producing matchings with marginally worse pairwise user happiness, the $\text{PRSTA}_{0.9}$ algorithm can produce more balanced matchings, in which the degree of unhappiness of the users that form at least one unhappy pair is lower. This is simply because of reducing the risk levels by setting $\alpha = 0.9$, and seeking to match users with possibly not perfect, but good enough candidates.

In Fig. 5c, we see that the proposed algorithms achieve better average quality of sensing scores with increasing number of workers, because as the number of workers increases, there will also be more high-quality workers. However, the GS and AOTA algorithms do not benefit much from this significantly as the former uses an inaccurate approximation for the expected user utilities, and the latter mostly ignores the matching opportunities that may come in the future to minimize the average task completion time.

In Fig. 6, we examine the performance of the algorithms with various task counts in the systems without capacity constraints. Since the workers do not have a capacity constraint, increasing the number of tasks does not escalate the competition between tasks (unlike what we will see in the presence of capacity constraints), thus we do not see big differences in the performance of the algorithms with the exception that the proposed algorithms perform slightly worse, and the others slightly better in terms of average user happiness.

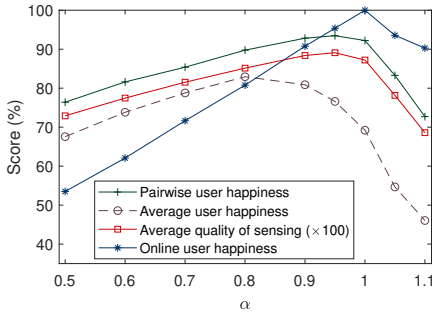
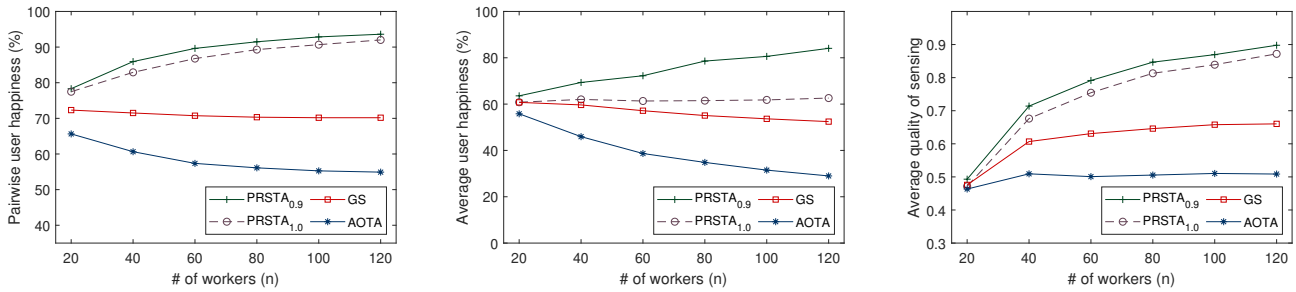


Fig. 8: Performance of the PRSTA_{α} algorithm in synthetic data set without capacity constraints $(m = 100, n = 60)$.

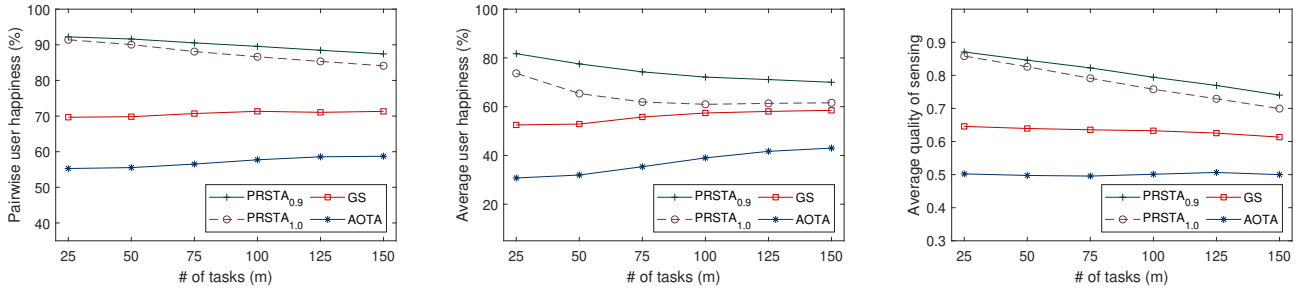
Next, in Fig. 7a-c, we look at the performance of the algorithms against varying degree of mobility, which is defined as the average percentage of the task regions visited by each worker. We generate instances with different mobility levels (i.e., percentage of all task regions visited by each worker, on average) by adjusting the range of the $\lambda_{i,j}$ values for worker-task pairs (e.g., increasing the average value of $\lambda_{i,j}$ results in lower mobility). As expected, with higher mobility, the high-quality workers visit more task regions, hence we see a profound increase in the average quality of sensing scores of the proposed algorithms. A remarkable point is that the GS and AOTA algorithms produce matchings with worse pairwise/average user happiness scores with increasing mobility, because the amount of better matching opportunities to be seen in the future, which are mostly neglected by these algorithms, becomes larger with increasing mobility. Lastly, in

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3118134, IEEE Internet of Things Journal



(a) (b) (c)

Fig. 9: Performance of algorithms with varying worker counts in synthetic data set with capacity constraints $(m = 100)$.



(a) (b) (c)

Fig. 10: Performance of algorithms with varying task counts in synthetic data set with capacity constraints $(n = 60)$.

Fig. 8, we look at the performance of the PRSTA$_\alpha$ algorithm with various values of α parameter. We see that the algorithm produces optimal task assignments in terms of online user happiness when $\alpha = 1$ (Corollary 2), and achieves the best performance in terms of all other metrics when α is between 0.8 and 1. With a smaller α value, the algorithm starts to match the worker-task pairs greedily, while it misses too many existing matching opportunities to wait for substantially better ones when we use an α value greater than 1.

2) Results with Worker Capacity Constraints: In Fig. 9, 10, and 11, we present the performance comparison of the algorithms in the MCS systems with capacity constraints (note that there is no result for the OSTA algorithm, as it can only be run in the systems without capacity constraints). Fig. 9 shows the performance of the algorithms with various worker counts. Although the relative performance of the algorithms is similar to the case without capacity constraints (Fig. 5), the quality of the produced matchings is generally slightly worse in terms of all performance metrics. This is because the high-quality workers will not be able to perform as many tasks as possible in this scenario, and the propriety of each matching decision becomes more important as there will be only a limited number of opportunities to make up for the previous decisions.

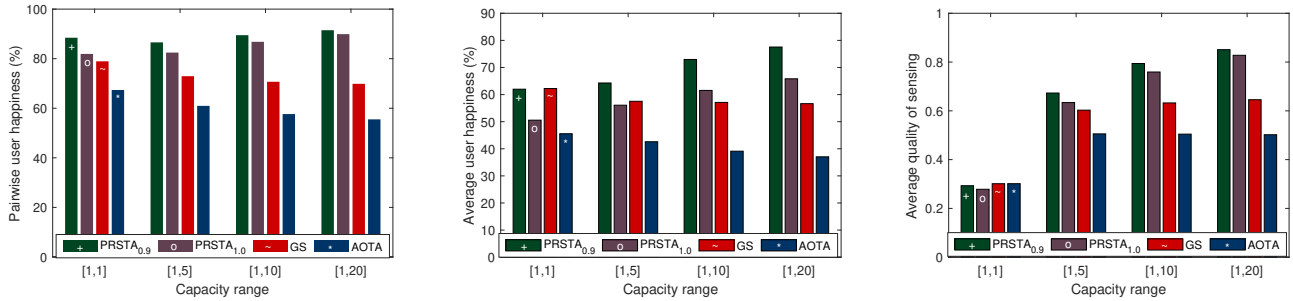
We inspect how the algorithms perform with varying number of tasks in presence of capacity constraints in Fig. 10. Different from the case without capacity constraints (Fig. 6), the user happiness and average quality of sensing achieved by the proposed algorithms get worse with increasing task counts, because, in this case, there is a competition between tasks as the high-quality workers can be matched to only a small number of tasks.

Another noteworthy point is that increasing the number of tasks has a different impact on the performance of the proposed algorithms and the others in terms of average user happiness. That is, the proposed algorithms perform slightly worse, while the GS and AOTA algorithms perform slightly better. This is due to the fact that the tasks will be, on average, matched to the workers with low quality scores or will be even unmatched when the number of tasks is large. This makes the cost of missing a present matching opportunity in terms of user happiness bigger, and the proposed algorithms consequently suffer as they frequently disregard the present matching opportunities to wait for better ones.

In Fig. 11, we analyze the effect of extending the worker capacity ranges on the performance of the algorithms. We observe that the proposed algorithms always outperform the others in terms of pairwise user happiness, and the performance difference becomes more significant with increasing worker capacities. However, when each worker can be matched with only a single worker, the GS algorithm has a similar performance with the PRSTA$_{0.9}$ algorithm in terms of average user happiness. Besides, the GS and AOTA algorithms achieve comparable average quality of sensing scores with the proposed algorithms when each worker has a capacity of one. This is because they have a lower risk of leaving the workers completely unmatched by skipping the existing matching opportunities, and this compensates for the loss of quality of sensing caused by the poor matching decisions they otherwise tend to make (as seen with larger worker capacities).

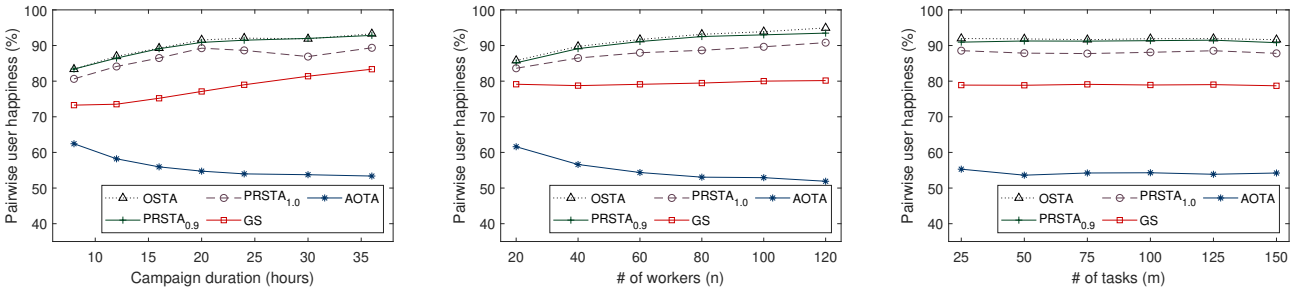
In Table III, we present a performance comparison of the PRSTA$_{0.9}$ algorithm against the optimal solutions obtained in the offline setting in terms of various metrics. The overall performance of Algorithm 3 shows that considering user preferences need not result in a significantly worse performance

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3118134, IEEE Internet of Things Journal



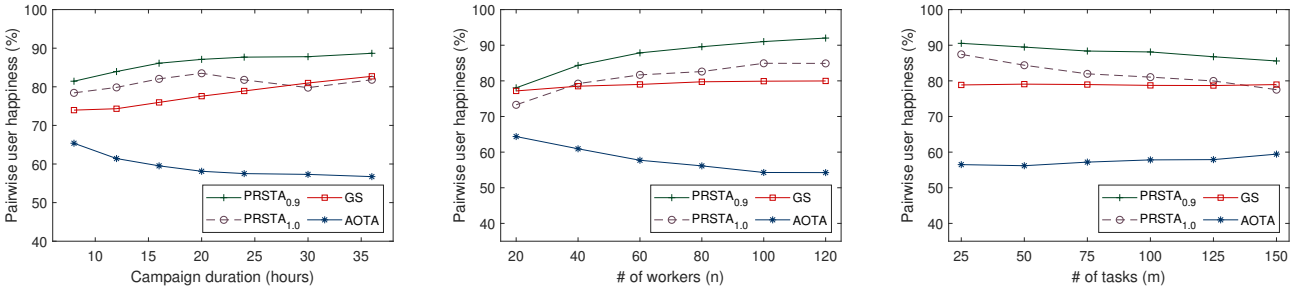
(a) (b) (c)

Fig. 11: Performance of algorithms with varying ranges of worker capacities in synthetic data set $(m = 100, n = 60)$.



(a) (b) (c)

Fig. 12: Performance of algorithms in the SF taxi data set without capacity constraints $(m = 100, n = 60)$.



(a) (b) (c)

Fig. 13: Performance of algorithms in the SF taxi data set with capacity constraints $(m = 100, n = 60)$.

Solution	PUH	Avg. QoS	Avg. Reward	Coverage
$\text{PRSTA}_{0.9}$	89.7	78.7	81.5	98.7
Algorithm 3	100	82.0	82.8	100
Offline-QoS	89.9	82.9	82.8	100
Offline-Reward	60.4	50.4	82.8	100
Offline-Coverage	59.8	49.8	82.8	100

TABLE III: Performance of the $\text{PRSTA}_{0.9}$ algorithm and the offline solutions maximizing user happiness (Algorithm 3), avg. QoS (Offline-QoS), avg. reward paid to workers (Offline-Reward), and task coverage (Offline-Coverage). The average QoS and reward scores are multiplied by 100 for ease of reading. (m=100, n=60.)

in terms of system-level utility metrics. Besides, since any online algorithm is expected to have a poorer performance than their offline counterparts in non-trivial settings, the $\text{PRSTA}_{0.9}$ algorithm (the only online solution in Table III) can be said to have a near-optimal or at least a decent performance in terms of all metrics considered.

3) Results with Real Data Set: In Fig. 12 & 13, we present the performance of the algorithms in terms of pairwise user happiness in the real data set without and with capacity constraints, respectively. Both figures show that the performance of the proposed algorithms and GS algorithm mostly improve with the extended campaign duration, yet that of the AOTA algorithm gets consistently worse as it makes almost all of the assignments right in the beginning of the campaign, as it does not care for high-quality assignments that may come later, but only seeks to minimize the average task completion time. In Fig. 12a & 13a, we see a slight fluctuation in the performance of the $\text{PRSTA}_{1.0}$ algorithm when the campaign duration is between 20-30 hours. This is mostly because of the changes in the movement patterns of the taxis between two consecutive days.

In Fig. 12, we observe that the relative performance of the algorithms and the impact of worker/task counts on the performance of all algorithms are quite similar to what we have seen in the synthetic data set (Fig. 5 & 6). In fact, the only major difference is that the GS algorithm achieves notably higher pairwise user happiness scores in the real data

set. Moreover, its performance is also significantly better with capacity constraints (Fig. 13) so that it slightly outperforms the $\text{PRSTA}_{1.0}$ algorithm when the ratio of the number of tasks to the number of workers is larger than 2. Yet it should be noted that it is always outperformed by the $\text{PRSTA}_{0.9}$ algorithm.

4) Running Time Results: Finally, we look at the running times of the algorithms[1] with varying worker/task counts and capacity ranges on an Intel core i7 processor with a memory of 16 GB and a speed of 2.5 GHz. We only present the running times for the synthetic data set as the comparison of running times of algorithms in the real data set does not exhibit any remarkable difference (except for the naturally larger total running times due to the longer campaign duration). Note that the time complexity of the GS algorithm is $O(mn)$, and that of the AOTA algorithm is $O(N^3)$, where $N = \max(|\mathcal{T}|, \sum_{w \in \mathcal{W}} c(w))$, as it requires to find a minimum weight matching on the bipartite graph between $\mathcal{W}$ and $\mathcal{T}$ with capacity constraints, for which we use the Hungarian algorithm [44]. Also recall that the worst-case running time of the PRSTA_α algorithm is $O(nmc_{max})$ (or $O(nm^2)$).

In Fig. 14, we see that in all cases the total running time of the GS algorithm is lower than the PRSTA_α algorithms, while its average running time per decision is higher. This is because the GS algorithm makes the matching decisions more greedily compared to the PRSTA_α algorithms, thus it matches most of the workers and tasks in the beginning of the campaign, which means that it will not be run again for these users, reducing the number of times it will be run in total. In general, the AOTA algorithm is also run much fewer times, but, due to its significantly worse time complexity, it mostly has the highest average running time, and ends up having the largest total running time with increasing worker capacities. On the other hand, the PRSTA_α algorithms have smaller average running times per decision, because they are run much more frequently after the first part of the campaign where there are generally fewer worker-task pairs that can still get matched.

Moreover, the $\text{PRSTA}_{1.0}$ algorithm has a significantly larger total running time than the $\text{PRSTA}_{0.9}$ algorithm as the former has a stronger requirement to match a pair, and consequently will be run considerably more times compared to the latter. This is also why we see an almost quadratic increase in the total running time of the $\text{PRSTA}_{1.0}$ algorithm with increasing worker and task counts. That is, when the number of workers/tasks increases, there will be more visits, and the $\text{PRSTA}_{1.0}$ algorithm will need to be run even more frequently. Lastly, since workers will be less selective when they have higher capacities, and accordingly tasks will end up getting matched earlier, the total/average running times of the PRSTA_α and GS algorithms do not get significantly larger with increasing capacities as seen in Fig. 14c, even though the worst-case running time of the PRSTA_α algorithm (i.e., $O(nmc_{max})$) hints at a linear grow with increasing worker capacities.

[1]Since the OSTA makes its matching decisions in constant time, we do not present its running time. One-time cost of obtaining $\mathbf{E}_j(s)$ values for the OSTA algorithm is also very small (e.g., 32 ms, which is about 10% of the total running time of the GS algorithm, when $m = 150$ and $n = 60$)

VI. Conclusion

In this paper, we focus on the task assignment problem in opportunistic MCS. First, we present a complete system model considering the uncertainty in worker trajectories and capacity constraints of workers, and formally define the preference-aware/stable task assignment problem. We then demonstrate how to efficiently examine all practical scenarios for assignment opportunities, which arise when the workers visit the task regions, to compute the expected utilities of the task requesters and workers with and without capacity constraints. Finally, we propose polynomial-time task assignment algorithms that are proven to be preference-aware, and show via extensive simulations that they significantly outperform the existing solutions in terms of worker/task requester happiness and quality of sensing. In future work, we aim to investigate the preference-aware task assignment problem with uncertain worker trajectories in settings with non-uniform worker qualities and complex, time-constrained sensing tasks that require the cooperation of multiple workers.

Acknowledgement

This material is based upon work supported by the U.S. National Science Foundation (NSF) under Grant CNS1647217.

References

[1] H. Chen, B. Guo, Z. Yu, and Q. Han, "Crowdtracking: real-time vehicle tracking through mobile crowdsensing," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 7570–7583, 2019.

[2] T. Liu, Y. Zhu, Y. Yang, and F. Ye, "ALC^2: When active learning meets compressive crowdsensing for urban air pollution monitoring," *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 9427–9438, 2019.

[3] P. Fraga-Lamas, T. M. Fernández-Caramés, M. Suárez-Albela, L. Castedo, and M. González-López, "A review on internet of things for defense and public safety," *Sensors*, vol. 16, no. 10, p. 1644, 2016.

[4] J. M. Cecilia, J.-C. Cano, E. Hernández-Orallo, C. T. Calafate, and P. Manzoni, "Mobile crowdsensing approaches to address the COVID-19 pandemic in Spain," *IET Smart Cities*, vol. 2, no. 2, pp. 58–63, 2020.

[5] B. Guo, Z. Yu, X. Zhou, and D. Zhang, "From participatory sensing to mobile crowd sensing," in *Proc. of IEEE Percom Workshops*, 2014, pp. 593–598.

[6] M. Zhang, P. Yang, C. Tian, S. Tang, X. Gao, B. Wang, and F. Xiao, "Quality-aware sensing coverage in budget-constrained mobile crowdsensing networks," *IEEE Trans. Veh. Technol.*, vol. 65, no. 9, pp. 7698–7707, 2016.

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

[8] F. Yucel, M. Yuksel, and E. Bulut, "QoS-based Budget Constrained Stable Task Assignment in Mobile Crowdsensing," *IEEE Trans. on Mobile Computing*, pp. 1–1, 2020.

[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, 2019.

[10] C. Dai, X. Wang, K. Liu, D. Qi, W. Lin, and P. Zhou, "Stable task assignment for mobile crowdsensing with budget constraint," *IEEE Trans. on Mobile Computing*, pp. 1–1, 2020.

[11] F. Yucel, M. Yuksel, and E. Bulut, "Coverage-aware stable task assignment in opportunistic mobile crowdsensing," *IEEE Trans. on Vehicular Technology*, pp. 1–1, 2021.

[12] Z. He, J. Cao, and X. Liu, "High quality participant recruitment in vehicle-based crowdsourcing using predictable mobility," in *Proc. of IEEE INFOCOM*, 2015, pp. 2542–2550.

[13] M. Xiao, J. Wu, L. Huang, R. Cheng, and Y. Wang, "Online Task Assignment for Crowdsensing in Predictable Mobile Social Networks," *IEEE Trans. on Mobile Computing*, vol. 16, no. 8, pp. 2306–2320, 2017.

[14] Y. Yang, W. Liu, E. Wang, and J. Wu, "A prediction-based user selection framework for heterogeneous mobile crowdsensing," *IEEE Trans. on Mobile Computing*, vol. 18, no. 11, pp. 2460–2473, 2018.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3118134, IEEE Internet of Things Journal

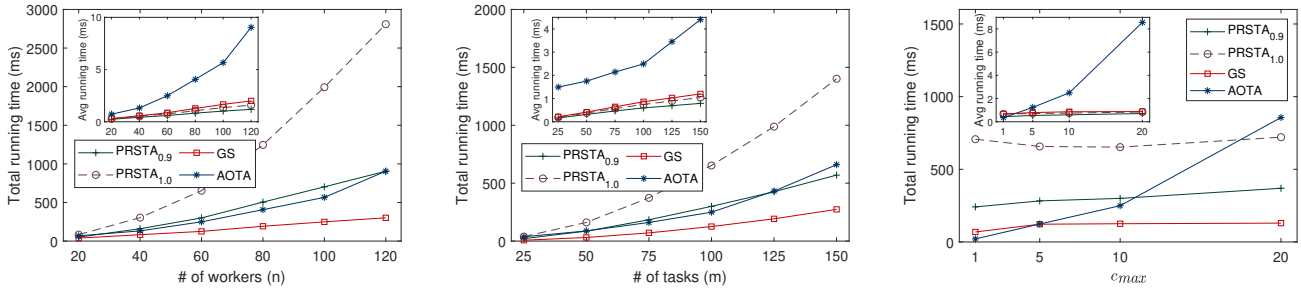


Fig. 14: Running times with varying worker and task counts (a,b); and worker capacity ranges [1,c_{max}] with $m = 100, n = 60$ (c). Average (total) running time is the average (total) time spent per (for all) matching decision(s).

[15] D. Gale and L. Shapley, "College admissions and stability of marriage. american mathematicas monthly, 69, 9-15," 1962.

[16] K. Yan, G. Luo, X. Zheng, L. Tian, and A. M. V. V. Sai, "A comprehensive location-privacy-awareness task selection mechanism in mobile crowd-sensing," *IEEE Access*, vol. 7, pp. 77 541–77 554, 2019.

[17] S. Song, Z. Liu, Z. Li, T. Xing, and D. Fang, "Coverage-oriented task assignment for mobile crowdsensing," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7407–7418, 2020.

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

[19] D. Manlove, *Algorithmics of matching under preferences*. World Scientific, 2013, vol. 2.

[20] P. Rajput, M. Chaturvedi, and V. Patel, "Opportunistic sensing based detection of crowdedness in public transport buses," *Pervasive and Mobile Computing*, vol. 68, p. 101246, 2020.

[21] "Waze," 2021. [Online]. Available: https://www.waze.com/

[22] J. Chen and J. Yang, "Maximizing coverage quality with budget constrained in mobile crowd-sensing network for environmental monitoring applications," *Sensors*, vol. 19, no. 10, p. 2399, 2019.

[23] M. Karaliopoulos, O. Telelis, and I. Koutsopoulos, "User recruitment for mobile crowdsensing over opportunistic networks," in *Proc. of IEEE INFOCOM*, 2015, pp. 2254–2262.

[24] Y. Zhan, Y. Xia, Y. Liu, F. Li, and Y. Wang, "Incentive-aware time-sensitive data collection in mobile opportunistic crowdsensing," *IEEE Trans. Veh. Technol.*, vol. 66, no. 9, pp. 7849–7861, 2017.

[25] Y. Zhan, Y. Xia, J. Zhang, and Y. Wang, "Incentive mechanism design in mobile opportunistic data collection with time sensitivity," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 246–256, 2017.

[26] J. Wu, M. Xiao, and L. Huang, "Homing spread: Community home-based multi-copy routing in mobile social networks," in *Proc. of IEEE INFOCOM*, 2013, pp. 2319–2327.

[27] A. Dhungana and E. Bulut, "Energy sharing based content delivery in mobile social networks," in *Proc. of IEEE WoWMoM*, 2019, pp. 1–9.

[28] Y. Li, D. Jin, P. Hui, and S. Chen, "Contact-aware data replication in roadside unit aided vehicular delay tolerant networks," *IEEE Trans. on Mobile Computing*, vol. 15, no. 2, pp. 306–321, 2015.

[29] P. Zhu, J. Li, D. Wang, and X. You, "Machine-learning-based opportunistic spectrum access in cognitive radio networks," *IEEE Wireless Communications*, vol. 27, no. 1, pp. 38–44, 2020.

[30] "SF Match," 2021. [Online]. Available: https://sfmatch.org/

[31] X. Yin, Y. Chen, C. Xu, S. Yu, and B. Li, "Matchmaker: Stable Task Assignment with Bounded Constraints for Crowdsourcing Platforms," *IEEE Internet of Things Journal*, 2020.

[32] B. Zhao, P. Xu, Y. Shi, Y. Tong, Z. Zhou, and Y. Zeng, "Preference-aware task assignment in on-demand taxi dispatching: An online stable matching approach," in *Proc. of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 2245–2252.

[33] H. Zheng and J. Wu, "Online to offline business: urban taxi dispatching with passenger-driver matching stability," in *Proc. of 37th IEEE Inter. Conf. on Distributed Computing Systems (ICDCS)*, 2017, pp. 816–825.

[34] H. Aziz, P. Biró, S. Gaspers, R. de Haan, N. Mattei, and B. Rastegari, "Stable matching with uncertain linear preferences," *Algorithmica*, vol. 82, no. 5, pp. 1410–1433, 2020.

[35] R. Bredereck, J. Chen, D. Knop, J. Luo, and R. Niedermeier, "Adapting stable matchings to evolving preferences," in *Proc. of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 02, 2020, pp. 1830–1837.

[36] B. Guo, H. Chen, Q. Han, Z. Yu, D. Zhang, and Y. Wang, "Worker-contributed data utility measurement for visual crowdsensing systems," *IEEE Trans. on Mobile Computing*, vol. 16, no. 8, pp. 2379–2391, 2016.

[37] R. K. Rana, C. T. Chou, S. S. Kanhere, N. Bulusu, and W. Hu, "Ear-phone: an end-to-end participatory urban noise mapping system," in *Proc. of the 9th ACM/IEEE international conference on information processing in sensor networks*, 2010, pp. 105–116.

[38] Y. Tong, J. She, B. Ding, L. Wang, and L. Chen, "Online mobile micro-task allocation in spatial crowdsourcing," in *Proc. of 32nd IEEE International Conf. on Data Engineering (ICDE)*, 2016, pp. 49–60.

[39] J. Ni, K. Zhang, Q. Xia, X. Lin, and X. S. Shen, "Enabling strong privacy preservation and accurate task allocation for mobile crowdsensing," *IEEE Trans. on Mobile Computing*, vol. 19, no. 6, pp. 1317–1331, 2020.

[40] "Uber," 2021. [Online]. Available: https://www.uber.com/

[41] D. Kong and F. Wu, "HST-LSTM: a hierarchical spatial-temporal long-short term memory network for location prediction." in *IJCAI*, vol. 18, no. 7, 2018, pp. 2341–2347.

[42] "Optimal stopping and applications." [Online]. Available: http://www.math.ucla.edu/tom/Stopping/Contents.html

[43] M. Piorkowski, N. Sarafijanovic-Djukic, and M. Grossglauser, "Crawdad dataset epfl/mobility (v. 2009-02-24)." [Online]. Available: https://crawdad.org/epfl/mobility/20090224

[44] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

Fatih Yucel (Member, IEEE) received B.S. degree in Gazi University in Turkey in 2017. He is now pursuing his PhD degree in the Computer Science Department of Virginia Commonwealth University under the supervision of Dr. Eyuphan Bulut. He is working on development of stable task assignment algorithms for mobile crowdsensing applications.

Eyuphan Bulut (Senior Member, IEEE) received the Ph.D. degree in computer science from Rensselaer Polytechnic Institute (RPI), Troy, NY, in 2011. He then worked as a senior engineer in Mobile Internet Technology Group (MITG) group of Cisco Systems in Richardson, TX for 4.5 years. He is now an Associate Professor with the Department of Computer Science, Virginia Commonwealth University (VCU), Richmond, VA. His research interests include mobile and wireless computing, network security and privacy, mobile social networks and crowd-sensing. Dr. Bulut has been serving as an Associate Editor in IEEE Access. He is also a member of ACM.