QoS-based Budget Constrained Stable Task Assignment in Mobile Crowdsensing

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

Abstract—One of the key problems in mobile crowdsensing (MCS) systems is the assignment of tasks to users. Most of the existing work aim to maximize a predefined system utility (e.g., quality of service or sensing), however, users (i.e., task requesters and performers/workers) may value different parameters and hence find an assignment unsatisfying if it is produced disregarding these parameters that define their preferences. While several studies utilize incentive mechanisms to motivate user participation in different ways, they do not take individual user preferences into account either. To address this issue, we leverage *Stable Matching Theory* which can help obtain a satisfying matching between two groups of entities based on their preferences. However, the existing approaches to find stable matchings do not work in MCS systems due to the many-to-one nature of task assignments and the budget constraints of task requesters. Thus, we first define two different stability conditions for user happiness in MCS systems. Then, we propose three efficient stable task assignment algorithms and discuss their stability guarantees in four different MCS scenarios. Finally, we evaluate the performance of the proposed algorithms through extensive simulations using a real dataset, and show that they outperform the state-of-the-art solutions.

Index Terms—Mobile crowdsensing, many-to-one task assignment, stable matching.

1 Introduction

Mobile Crowdsensing (MCS) is a relatively new paradigm empowered by the growing sensing capabilities of mobile devices (e.g., GPS, microphone, camera) and has been utilized as a means to accomplish the requested sensing tasks much more quickly by taking advantage of the power of crowd [1]. An MCS system consists of a platform, requesters, tasks and workers. Task requesters post their tasks together with their requirements (e.g., deadline, budget limits) and the rewards they provide. The workers register to the system and indicate their qualifications and possible restrictions (e.g., regional and temporal availability, minimum reward requirement). The platform then defines eligibility of each worker for the available tasks and either assigns them based on a predefined system utility (e.g., minimization of rewards provided) or lets the workers and task requesters interact and agree on the task allocation in a distributed fashion based on their own criteria and preferences.

A pivotal problem in MCS is the assignment of sensing tasks to workers. In fact, the utility of an MCS system is generally quantified by the quality of the assignments made by the adopted task assignment algorithm. However, the quality of assignments has been measured differently in the literature. For example, some studies [2], [3] favor the assignments that minimize the travel distance of workers, while others [4], [5] prefer those that maximize the total quality of service (QoS) received by task requesters. Despite the variety of existing task assignment algorithms [6], the ultimate goal of the assignment is mostly defined as the maximization of a system utility without considering the individual user needs and preferences. However, such solutions may result in dissatisfied users and impair their future participation, as users in practice may not want to sacrifice their individual convenience for the system utility.

To address this problem, in this paper, we leverage *Stable Matching (SM) Theory*, which integrates user preferences in a matching problem [7], to develop stable task assignments specific to MCS systems while considering the budget/QoS requirements of tasks and the relation between the QoS provided by workers and the rewards they gain. Next, we first discuss the benefits of such a stable matching based task assignment for MCS systems, and summarize our contributions.

First of all, a stable matching solution based on individual user preferences allows incorporating various metrics that are appraised uniquely by each user. For example, the priority of a worker might be the proximity to the task location, so he would form his preference list in a way that the closer tasks precede the others. On the other hand, another worker who does not mind traveling long distances can form his preference list solely based on the rewards he will be paid. Moreover, they can even reflect their own personal interests in their preference lists such as location of tasks being close to their home or work, and pleasure of performing the tasks (e.g., preference for taking pictures of a scene of interest over recording noise pollution).

The goal with the stability is to ensure that no user u (i.e., a worker or a requester) can lay claim to have deserved a better assignment. That is, all the possible assignments

- *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.*
- *M. Yuksel is with the Department of Department of Electrical and Computer Engineering, University of Central Florida, Orlando, FL 32816. E-mail: {Murat.Yuksel}@ucf.edu.*

that user u prefers[1] more than his current assignment in the stable matching are matched with someone they prefer to user u; thus, they would not like to break up with their current assignments to get matched with user u. This ensures that workers will be satisfied with their assignments and will be motivated to carry out their tasks, which will in turn improve their performance [8] and the quality of results. Moreover, this also ensures that task requesters will obtain the most benefit while respecting the workers' preferences.

In MCS systems where each task can recruit multiple workers within her budget constraints, the stability can be defined in two different ways: pairwise and coalitional (the formal stability definitions are given in Section 3). Due to the many-to-one nature of task assignments and budget constraints, the conditions of both pairwise and coalitional stability differ from the classic stability conditions specified in [7], thus existing stable matching solutions cannot be used to find pairwise or coalitional stable matchings in such systems. Moreover, depending on the relation (i.e., proportional or not) between the QoS provided by workers and the reward they gain, the hardness of the problem and the corresponding solution approach completely change.

In order to develop stable task assignments in MCS systems, we first define the stability conditions and categorize MCS scenarios based on two different criteria. Then, we present our three solutions and discuss their stability guarantees in different MCS scenarios. Our key contributions can be summarized as follows:

- We present QoS-based stability conditions under budget constraints in many-to-one MCS systems, and discuss existence and hardness results for stable matchings in different types of MCS systems.
- We propose a polynomial time algorithm to find pairwise stable matchings in MCS systems in which the preference profiles of tasks are identical (i.e., *uniform MCS system*), which is a significant improvement from the exponential time algorithm proposed in [9]. Also, our algorithm does not require the rewards that workers will be paid to be proportional with the corresponding QoS they provide (i.e., *proportional MCS system*), while [9] does.
- Despite the nonexistence results for pairwise stable matchings in general settings, we prove that there always exists a pairwise stable matching in a proportional MCS system by providing a pseudo-polynomial time algorithm that always finds pairwise stable matchings in these systems. Furthermore, this algorithm outperforms all the benchmark algorithms in terms of pairwise stability regardless of the type of the MCS system.
- We propose a heuristic algorithm that also runs in pseudo-polynomial time and produces considerably higher quality assignments in terms of overall stability and user happiness compared to the benchmark algorithms especially in proportional MCS systems.
- In addition to the theoretical analysis of the proposed algorithms, we provide extensive simulation results where we compare the performance of our algorithms with three benchmark algorithms in different types of MCS systems.

The rest of the paper is organized as follows. We present an overview of related work in Section 2. In Section 3, we provide the system model together with stability definitions and discuss the existence and hardness of stable matchings in different MCS categories. In Section 4, we elaborate on the proposed solutions and discuss their stability guarantees. In Section 5, we present an extensive evaluation of the proposed algorithms in various settings via simulations. Finally, we end up with conclusion in Section 6.

2 Related Work

2.1 Mobile Crowdsensing

Mobile crowdsensing has attracted a lot of attention recently and numerous studies exploring different aspects have been performed. One of the key problems investigated is the task assignment (or worker recruitment) problem since the overall performance of an MCS system and the satisfaction of its users are highly dependent on the efficiency of the assignments. Different objectives have been considered such as maximizing the number of completed tasks [10], minimizing the completion times of tasks [11], incentives provided to the users [12], energy consumption [4], and traveling distance of workers [3]. The heterogeneity of tasks [13], [14], security [15], privacy [16], and trustfulness [17] of workers have also been addressed in some studies.

Besides, similar to this paper, a number of studies [5], [18], [19], [20], [21], [22] have adopted quality-based utility functions, and proposed incentive mechanisms or task assignment methods that maximize the total quality of service/sensing. Particularly, [18] and [19] study the problem of finding and recruiting the worker set that will provide the largest weighted coverage quality over a set of points of interest that the service provider aims to cover with a fixed budget. On the other hand, [20], [21], [22] propose incentive mechanisms, where the utilities of workers for the tasks of requesters are determined based on the quality of the data they can provide for those tasks, which is estimated by various factors such as their reputation and the quality of sensor devices that they use to carry out the tasks. Moreover, [21] also considers the temporal changes in the quality of workers for the crowdsourcing campaign, and proposes a framework to estimate these changes for each run of the task allocation scheme based on historical performance of workers.

Despite such extensive work, the problem is mostly studied from the system's perspective without considering individual preferences of users (i.e., workers and requesters). However, users may not be willing to sacrifice their individual convenience for the system utility (e.g., QoS), thus resulting task assignments may not be appealing to users and undermine future participation. Note that in the studies that examine the mechanisms incentivizing users to participate in crowdsensing, such as auctions and reputation systems, this problem is not solved either, as they still aim to maintain system goals by promoting participation. Some very recent studies [9], [23] address this issue, and consider user preferences and stability in the assignment process. [9] studies the many-to-one stable task assignment problem

1. When we use the active voice for a task (e.g., prefers, pays a reward), we mean the person/entity that corresponds to it, i.e., the task requester/owner. Also, we use male and female subject pronouns for workers and tasks respectively as a reference to the original stable marriage problem [7].

under budget constraints and proposes an ILP-based exponential time algorithm to find pairwise stable matchings in MCS systems that are both uniform and proportional. [23] studies the same problem, but assumes each task has a capacity (instead of a budget) that specifies the maximum number of workers she can hire. Hence, these solutions are either computationally expensive or only applicable to a certain type of MCS systems.

2.2 Stable Matching

Stable Matching (or Stable Marriages) (SM) problem has been introduced by Gale and Shapley [7], and can be defined as the problem of matching two groups of objects such that no pair or set of users has a mutual or collective desire (based on their preferences) to deviate from their assigned partners in order to match with each other. SM has been utilized to model various problems such as spectrum sharing [24], driver-rider matching in ride-sharing systems [25], assignment of medical school students to hospitals for residency training program [26], and matching electric vehicles for power transfer [27]. There are also variations of stable matching problem where the stability is considered together with another utility metric. For example, [28] studies the maximum weighted stable matching problem (i.e., the stability is the primary objective) and proposes a polynomial time exact algorithm to solve it. On the other hand, [10] studies the problem of finding a matching with minimum instability among all maximum cardinality matchings (i.e., the stability is the secondary objective). As this problem is NP-hard, the authors propose two different polynomial time heuristic algorithms.

The most relevant studies to this paper in the stable matching literature are [29] and [30], which study the many-to-one stable matching of students-colleges and doctors-hospitals, respectively. In [29], all colleges define a utility and a wage value for students, and aim to hire the best set of students (i.e., with the highest total utility) within their budget constraints. Each student also forms a preference list over colleges. The authors prove that there may not exist a stable matching in this setting and even checking the existence is NP-hard. However, they provide a polynomial time algorithm that finds pairwise stable matchings in the so called typed weighted model where students are categorized into groups (e.g., Master's and PhD students) and colleges are restricted to define a set of possible wages for each group (i.e., they cannot define a particular wage for each student). [30] studies the same problem and proposes two different fully polynomial-time approximation algorithms with some performance guarantee in terms of coalitional stability for general and proportional (i.e., the wage of doctors are proportional to their utility for hospitals) settings. However, the study does not provide an experimental analysis of the algorithms or discuss their actual/expected performance in these settings. Moreover, the proposed solutions can only be applied to a limited set of scenarios. In this paper, we provide solutions for different types of MCS systems, compare them with the existing many-to-one stable matching algorithms [9], [30], and show that our solutions outperform them most of the time in terms of different stability conditions.

The concept of stability in the fairness problems, which are commonly studied in the wireless communication literature [31], [32], is different than the one considered in this paper. This is because in the fairness problems there is a set of resources (e.g., bandwidth) that need to be distributed among a group of users in a fair manner [33]. This is a one-sided allocation problem with no preference relation between the resources and users, and the stability refers to the equilibrium state of the allocation in terms of fairness (e.g., Lyapunov stability [34], flow-level stability [35]). However, in our problem, each node (i.e., worker/task) in the both side of the matching problem is a distinct individual with personal preferences (i.e., two-sided matching), and the stability concerns with the happiness of all nodes with their assignments based on their bilateral preferences.

3 System Model

3.1 Assumptions

We assume a system model with a set of workers $\mathcal{W} = \{w_1, w_2, \ldots, w_n\}$ and a set of sensing tasks $\mathcal{T} = \{t_1, t_2, \ldots, t_m\}$. Let $c_t(w)$ denote the cost of performing task t for worker w, which may be calculated by taking into account various factors such as the time and cost required to travel to the task location and perform the task, energy consumption on the worker's device due to sensing, and privacy risks to the worker. Also, let $r_t(w)$ denote the reward that worker w is offered to carry out task t. Since a rational worker will aim to maximize his profit and will not accept to perform the tasks that cost higher than the corresponding rewards he will be paid, we can define the preference list of worker w as

$$P_w = t_{i_1}, t_{i_2}, \ldots, t_{i_k} \tag{1}$$

where $P_w \subseteq \mathcal{T}$, $\forall t \in P_w$, $r_t(w) > c_t(w)$, and $\forall t' = t_{i_j}, t'' = t_{i_{j+1}}$, $r_{t'}(w) - c_{t'}(w) > r_{t''}(w) - c_{t''}(w)$. We denote the jth task (t_{i_j}) in P_w by $P_w(j)$ and utilize $t' \succ_w t''$ notation to express that t' precedes t'' in P_w. For each worker w, $\succ_w$ is a strict relation, so even if two tasks provide the same gain to worker w, we assume that he prefers one over another (i.e., no ties are allowed). Note that in our system model, we only need the preference profiles of workers, so a worker can form his preference list himself by estimating his profit from each task using the announced rewards for the tasks on the platform and then submit only the list to the platform. Alternatively, he can submit an estimated cost value for each task to the platform, which can then form his preference list based on this information.

On the other hand, a rational task requester will try to maximize the total quality of service (QoS) she gets from the workers she hires within her budget constraint. Let $q_t(w)$ denote the QoS that worker w can provide for task t and b_t denote the budget of task t. Then, we can define the preference list of task t as

$$P_t = S_1, S_2, \ldots, S_k \tag{2}$$

where $\forall S \in P_t$, $S \subseteq \mathcal{W}$ and $\sum_{w \in S} r_t(w) \leq b_t$, and $\forall S_i, S_{i+1} \in P_t$, $\sum_{w \in S_i} q_t(w) \geq \sum_{w \in S_{i+1}} q_t(w)$. Note that we allow ties in preference lists of tasks as it is very likely for tasks to have multiple sets with an equal total quality value in their preference list. Thus, a task is said to prefer a

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/TMC.2020.2997280, IEEE Transactions on Mobile Computing

TABLE 1: Notations.

Notation	Description
$\mathcal{W}, \mathcal{T}$	Set of workers and tasks, respectively
n, m	Number of workers and tasks, respectively
$\mathcal{M}$	Many-to-one task assignment
$c_t(w)$	Cost of performing task t for worker w
$q_t(w)$	QoS that worker w can provide for task t
$q_t(S)$	Total QoS that $S \subseteq \mathcal{W}$ can provide for task t
$Q_t(S)$	List of QoS that workers in S can provide for task t
$r_t(w)$	Reward offered to worker w by task t
$r_t(S)$	Total reward offered to $S \subseteq \mathcal{W}$ by task t
$R_t(S)$	List of rewards offered to workers in S by task t
b_t	Budget of task t
β	$\max_{t \in \mathcal{T}} b_t$
$b_t^{\mathcal{M}}$	Remaining budget of task t in $\mathcal{M}$
P_u	Preference list of user (worker/task) u
$P_w(x)$	xth task in P_w

Fig. 1: A uniform and proportional MCS instance with 3 workers $(1, 2, 3)$ and 2 tasks (x, y). $[crq_t(w) = c_t(w), r_t(w), q_t(w)]$

set S' of workers to another set S'' of workers only if S' has a greater total quality value than S''. For ease of reading, we let $r_t(S) = \sum_{w \in S} r_t(w)$, $q_t(S) = \sum_{w \in S} q_t(w)$ and use $S' \succ_t S''$ notation to indicate $q_t(S') > q_t(S'')$.

Given a worker-task pair (w, t), we assume that $\nexists S \in P_t : w \in S$ if worker w finds task t unacceptable (i.e., $r_t(w) <= c_t(w)$). Also, in MCS systems where the rewards are determined by the server instead of the task requesters, we might have $r_t(w) > b_t$ for a worker-task pair (w, t). In this case, if $t \in P_w$, we remove task t from P_w as the budget of task t is insufficient to recruit worker w. Lastly, we assume that for each task t, b_t and $r_t(w)$ values for all $w \in \mathcal{W}$ are either defined as integers or scaled into integers with the smallest scaling factor (different tasks might have different scaling factors).

3.2 Matching Stability Definitions

To make a formal development and evaluation of our matching algorithms, we make the following definitions and observations:

Definition 1 (Feasible matching). *A mapping*

$$\mathcal{M} : (\mathcal{W} \mapsto \mathcal{T} \cup \{\emptyset\}) \cup (\mathcal{T} \mapsto 2^{\mathcal{W}})$$

TABLE 2: Preference lists of the users in Fig. 1.

User	Preference list
x	$\{2,3\}, \{1\}, \{2\}, \{3\}, \emptyset$
y	$\{2\}, \{3\}, \emptyset$
1	x
2	x, y
3	y, x

is a feasible many-to-one matching if it satisfies the following:

- $\forall (w, t) \in \mathcal{W} \times \mathcal{T}$, $\mathcal{M}(w) = t$ *iff* $w \in \mathcal{M}(t)$,
- $\forall w \in \mathcal{W}$, $t \in P_w$ *if* $\mathcal{M}(w) = t$,
- $r_t(\mathcal{M}(t)) \leq b_t$.

Here, given a matching $\mathcal{M}$ and $w \in \mathcal{W}, t \in \mathcal{T}$, the partner[2] of worker w is denoted by $\mathcal{M}(w)$ and the partner set of task t is denoted by $\mathcal{M}(t)$. If $\mathcal{M}(u) = \emptyset$ for user $u \in \mathcal{W} \cup \mathcal{T}$, it means user u is unmatched in $\mathcal{M}$. Note that the last set in the preference list of each task t is $\emptyset$, so we have $S \succ_t \emptyset, \forall S \in (P_t \setminus \emptyset)$. Also, even though the preference lists of workers do not include $\emptyset$, since we assume that the workers in our system are rational, we have $t \succ_w \emptyset$, for all $w \in \mathcal{W}$ and $t \in P_w$. We denote the remaining budget of task t in $\mathcal{M}$ by $b_t^{\mathcal{M}} = b_t - r_t(\mathcal{M}(t))$.

Definition 2 (Unhappy pair). *Given a matching $\mathcal{M}$, a worker w and a task t form an unhappy (blocking) pair $\langle w, t \rangle$ if $t \succ_w \mathcal{M}(w)$ and there is a subset $S \subseteq \mathcal{M}(t)$ such that $\{w\} \succ_t S$ and $r_t(w) \leq b_t^{\mathcal{M}} + r_t(S)$.*

Definition 3 (Pairwise stable matching). *A matching $\mathcal{M}$ is said to be pairwise stable if it does not admit any unhappy pairs.*

Definition 4 (Unhappy coalition). *Given a matching $\mathcal{M}$, a subset of workers $S \subseteq \mathcal{W}$ and a task t form an unhappy (blocking) coalition $\langle S, t \rangle$ if $\forall w \in S$, $t \succ_w \mathcal{M}(w)$ and there is a subset $S' \subseteq \mathcal{M}(t)$ such that $S \succ_t S'$ and $r_t(S) \leq b_t^{\mathcal{M}} + r_t(S')$.*

The reason such a coalition $\langle S, t \rangle$ is said to block the stability of the matching is that the users in the coalition can communicate with each other and decide to jointly update their partners to have a better assignment.

Definition 5 (Coalitionally stable matching). *A matching $\mathcal{M}$ is said to be coalitionally stable if it does not admit any unhappy coalitions.*

Note that the coalitional stability is a stronger requirement compared with the pairwise stability. In fact, since every unhappy pair $\langle w, t \rangle$ corresponds to an unhappy coalition $\langle \{w\}, t \rangle$, coalitionally stable matchings are also pairwise stable, but not the other way around. For example, on the MCS instance shown in Fig 1 (with preference lists given in Table 2), the matching in which task x is matched with the worker 1 and task y is matched with the worker 2 is a pairwise stable matching, yet it has an unhappy coalition ($\langle \{2, 3\}, x \rangle$), and hence is not coalitionally stable. This is because worker set $\{2, 3\}$ provides a higher QoS (i.e., 7) to x than what her current assignment, worker 1, provides (i.e., 5) and both worker 2 and worker 3 prefer task x to their currently assigned tasks as their net income (i.e., reward - cost) with task x are larger.

2. The partner of a worker refers to the task that the worker is assigned to perform, while the partner set of a task refers to the set of all workers assigned with the task.

Algorithm 1: Check Pair $(w, t, \mathcal{M}, \textit{CheckIf})$

Input: (w, t): Worker, task pair to check
$\mathcal{M}$: The current many-to-one matching
CheckIf = CoalitionallyUnhappyPair or UnhappyPair

1 **if** $\mathcal{M}(w) = t$ *or* $\mathcal{M}(w) \succ_w t$ **then**
2 **return** *false*
3 **end**
4 $S \leftarrow \mathcal{M}(t)$
5 **if** *CheckIf = CoalitionallyUnhappyPair* **then**
6 **foreach** $w' \in \mathcal{W}$ **do**
7 **if** $t \in P_w$ *and* $t \succ_{w'} \mathcal{M}(w')$ **then**
8 $S \leftarrow S \cup \{w'\}$
9 **end**
10 **end**
11 $S \leftarrow S \setminus \{w\}$
12 **end**
13 $S_{max} \leftarrow$ *solve01Knapsack*$(b_t - r_t(w), R_t(S), Q_t(S))$
14 **if** $q_t(S_{max}) + q_t(w) > q_t(\mathcal{M}(t))$ **then**
15 **return** *true*
16 **else**
17 **return** *false*
18 **end**

Definition 6 (Coalitionally unhappy pair)**.** *Given a matching $\mathcal{M}$, a worker w and a task t form a coalitionally unhappy pair if there is an unhappy coalition $\langle S, t\rangle$ such that $w \in S$.*

When the objective is to achieve pairwise stability, the number of unhappy pairs can be used as the degree of instability of the resulting matching. On the other hand, it is not feasible to use the number of unhappy coalitions to measure the coalitional stability for two reasons. First, since the number of unhappy coalitions in a matching can be as large as $m(2^n - 1)$, even just enumerating them would take exponential time. Second, given a worker-task pair (w, t), knowing all the unhappy coalitions $\langle S, t\rangle : w \in S$ does not provide any useful information to either party, whereas knowing that there is at least one makes them aware that there is a matching in which they are matched to each other and are both better off. For these reasons, we propose to use the number of coalitionally unhappy pairs to measure the coalitional instability of a matching. Note that we can check whether a certain worker-task pair (w, t) is a coalitionally unhappy pair as described in Algorithm 1. This algorithm uses a sub-procedure named *solve01Knapsack*(c, W, V) in line 13, which denotes the dynamic-programming based algorithm [36] to find the optimal solution for an instance of 0-1 knapsack problem with a knapsack capacity of c, and k items ($k = |W| = |V|$) whose weights and values are given in order in W and V, respectively. It returns the item set that has the largest total value among the sets that have a total weight less than c. Since solving the 0-1 knapsack problem is the most costly operation in Algorithm 1 and has a time complexity of $\mathcal{O}(nb_t)$, we can find and count the unhappy and coalitionally unhappy pairs in a matching in pseudo-polynomial time $\mathcal{O}(mn^2\beta)$, where $\beta = \max_{t\in\mathcal{T}} b_t$. Lastly, it should be noted that every unhappy pair is also a coalitionally unhappy pair.

3.3 Classification of MCS Systems

We classify MCS systems according to the variability in the QoS provided by the workers for different tasks (uniform/non-uniform), and the relationship between the QoS provided by the workers and the rewards they are offered (proportional/non-proportional). Note that these classifications are exclusive; thus, it is possible to have four different MCS systems, namely, (i) proportional and non-uniform, (ii) non-proportional and non-uniform, (iii) proportional and uniform, and (iv) non-proportional and uniform.

Definition 7 (Uniform MCS system)**.** *An MCS system is called uniform if the QoS provided by each worker is the same for all tasks.*

That is, for all $(w, t, t') \in \mathcal{W} \times \mathcal{T}^2$, $q_t(w) = q_{t'}(w)$. This indicates that all tasks have the same preference ordering for all $S, S' \subseteq \mathcal{W}$ since we have $q_t(S) = q_{t'}(S)$ and $q_t(S') = q_{t'}(S')$, $\forall t, t' \in \mathcal{T}$. However, they may not have the same preference list because of the difference in their budgets (i.e., a task will not include a certain worker set in her preference list if the total reward to be paid to that worker set exceeds her budget) and being unacceptable to different workers. Despite their simplicity, uniform MCS systems are actually quite common. For example, all MCS systems in which the QoS of workers are determined solely based on trustworthiness or seniority scores of workers (e.g., Waze [37] in which users are ranked according to what is called Waze points that they collect by performing different tasks such as editing the map), or that only contain very basic tasks (e.g., taking a picture of a scene, measuring noise pollution) that do not demand any expertise and can be performed as effectively by all workers can be viewed as uniform MCS systems. An MCS system that is not uniform is called *a non-uniform MCS system*.

Definition 8 (Proportional MCS system)**.** *An MCS system is called proportional if, for each task, the rewards that are offered to the workers are proportional to the QoS they provide.*

That is, $\frac{r_t(w)}{q_t(w)} = \theta_t$ for all $(w, t) \in \mathcal{W} \times \mathcal{T}$, where θ_t is a constant defined by task t. Thus, different tasks might have a different reward per QoS ratio. Note that in proportional MCS systems, the objective of tasks can be expressed as maximizing the total reward paid to workers within the budget constraints as it also maximizes the total QoS they get. Hence, we will use $r_t(w)$ in place of $q_t(w)$ in the relevant sections. Also, if an MCS system is not proportional, we simply call it *a non-proportional MCS system*.

3.4 Existence of Stable Matchings

In the following theorems, we give the existence results for pairwise and coalitionally stable matchings in different types of MCS systems.

Theorem 1. *There exists a non-proportional MCS instance, in which none of the feasible matchings is pairwise stable.*

Proof. We prove by giving a counterexample. Let $q_x(3) = 6$ in the instance given in Fig. 1. This adjustment results in $\frac{r_x(2)}{q_x(2)} \neq \frac{r_x(3)}{q_x(3)}$, hence makes the instance non-proportional. It also changes the preference list of task x, which becomes

$P_x = \{2,3\}, \{3\}, \{1\}, \{2\}, \emptyset$. The preference list of the other users remain the same as given in Table 2. Let us analyze all possible task assignments in this modified instance.

- Assume $\mathcal{M}(y) \neq \{3\}$. Then, for $(3, y)$ to not be an unhappy pair, we must have $\mathcal{M}(y) = \{2\}$. In this case, if $\mathcal{M}(x) \neq \{3\}$, then $(3, x)$ is an unhappy pair. If $\mathcal{M}(x) = \{3\}$, then $(2, x)$ is an unhappy pair.
- Assume $\mathcal{M}(y) = \{3\}$. If $\mathcal{M}(x) \neq \{1\}$, then $(1, x)$ is an unhappy pair, and if $\mathcal{M}(x) = \{1\}$, then $(2, y)$ is an unhappy pair.

Therefore, we conclude that no pairwise stable matching exists in the given non-proportional MCS instance. □

Since every unhappy pair is also an unhappy coalition, the following corollary is an immediate result of Theorem 1.

Corollary 1.1. *There exists a non-proportional MCS instance, in which none of the feasible matchings is coalitionally stable.*

Theorem 2. *There exists a uniform and/or proportional MCS instance, in which none of the feasible matchings is coalitionally stable.*

Proof. We prove by giving a counterexample. Note that the MCS instance given in Fig. 1 is both uniform and proportional. Then, it suffices to show that all feasible matchings that can be defined on this instance have at least one unhappy coalition.

- Assume $\mathcal{M}(x) \neq \{1\}$. Then, the worker set $\{1\}$ and task x do not form an unhappy coalition only if $\mathcal{M}(x) = \{2,3\}$. However, if $\mathcal{M}(x) = \{2,3\}$, then the worker set $\{3\}$ and task y form an unhappy coalition.
- Assume $\mathcal{M}(x) = \{1\}$. If $\mathcal{M}(y) \neq \{2\}$, then $(\{2\}, y)$ is an unhappy coalition. If $\mathcal{M}(y) = \{2\}$, then $(\{2,3\}, x)$ is an unhappy coalition.

□

We will show in the next section that there always exists a pairwise stable matching in uniform and proportional MCS systems.

3.5 Hardness of Finding Stable Matchings

Consider an MCS instance with n workers $(w_1, w_2, \ldots, w_n)$ and a single task (t). Note that, by definition, finding a coalitionally stable matching in this instance is exactly the same problem with finding an optimal solution for a 0-1 knapsack instance with a knapsack that has a weight capacity of b_t and n items such that the weight and value of ith item are, respectively, the reward $(r_t(w_i))$ and the QoS $(q_t(w_i))$ of ith worker (w_i) for task t. Since the 0-1 knapsack problem is NP-hard, we can conclude that the problem of finding a coalitionally stable matching (even for an MCS system that has only one task) is NP-hard, as well.

In fact, as proved in [29], given a many-to-one matching instance with budget constraints, both checking the existence of a coalitionally stable matching and finding one if exists are NP-hard. Also, since even checking whether a particular worker-task pair form an unhappy pair in a given matching is NP-hard (as it requires to solve the corresponding 0-1 knapsack problem shown in Algorithm 1), it is highly likely that the same hardness results apply to the pairwise stable matchings, as well.

3.6 Problem Formulation

Given the definitions as well as the nonexistence and hardness results provided above, we can formally define our objective function as

$$\text{minimize} \sum_i \sum_j u_{ij} \tag{3}$$

such that

$$\sum_j x_{ij} \leq 1 \qquad \forall i$$

$$\sum_i x_{ij} \times r_{t_j}(w_i) \leq b_{t_j} \qquad \forall j$$

$$x_{ij} \leq e_{ij} \qquad \forall i, j$$

where

$$u_{ij} = \begin{cases} 1, & \text{if } \langle w_i, t_j \rangle \text{ is a (coalitionally) unhappy pair} \\ 0, & \text{otherwise} \end{cases}$$

$$x_{ij} = \begin{cases} 1, & \text{if } w_i \text{ is assigned to } t_j \\ 0, & \text{otherwise} \end{cases}$$

$$e_{ij} = \begin{cases} 1, & \text{if } t_j \in P_{w_i} \text{ (\textit{eligibility})} \\ 0, & \text{otherwise.} \end{cases}$$

That is, we would like to produce feasible matchings with as few (coalitionally) unhappy pairs as possible. When the goal is to minimize the number of *coalitionally* unhappy pairs (i.e., $u_{ij} = 1$ for coalitionally unhappy pairs), the optimization objective in (3) attains the strongest stability conditions, but becomes intractable in all types of MCS systems. On the other hand, minimizing the number of *unhappy pairs* is a more practical objective and generally adequate to virtually satisfy the users for two reasons. First, it is much harder for a pair of users to find out that they are a coalitionally unhappy pair than that they are simply an unhappy pair, since the former requires them to know the preferences and the current partners of all workers in the platform. Second, unlike unhappy pairs, modifying the matching to make a coalitionally unhappy pair happy necessitates that all the workers in the corresponding unhappy coalition (see Definition 6) cooperate and break up with their current partners simultaneously, which might be hard to attain.

It is also desirable to minimize the degree of user unhappiness in general rather than the number of unhappy users. In this case, an alternative objective would be to minimize the highest dissatisfaction ratio in the matching from the perspective of tasks, since they, unlike workers who are simply either happy or not with their assignments, have a degree of unhappiness based on the total QoS service they receive (i.e., *non-binary utility*) in the many-to-one matching scenario described earlier. Formally, let

$$\mathcal{S}_t = \{S : \langle S, t \rangle \text{ is an unhappy coalition}\},$$

and $\forall S \in \mathcal{S}_t$, let $S^R \subseteq \mathcal{M}(t)$ be the set with the lowest total quality such that its removal from the partner set of task t suffices to accept S (i.e., $r_t(S) \leq b_t^{\mathcal{M}} + r_t(S^R)$). Then, the dissatisfaction ratio of task t can be computed by:

$$\delta_t = \begin{cases} 1, & \text{if } \mathcal{S}_t = \emptyset \\ \infty, & \text{if } \mathcal{S}_t \neq \emptyset \text{ and } \mathcal{M}(t) = \emptyset \\ \max\limits_{S \in \mathcal{S}_t} \frac{q_t(S) + q_t(\mathcal{M}(t) \setminus S^R)}{q_t(\mathcal{M}(t))}, & \text{otherwise.} \end{cases}$$

Algorithm 2: Uniform Task Assignment ($\mathcal{W}, \mathcal{T}, P_{\mathcal{T}}$)

Input: $\mathcal{W}$: The set of workers
$\mathcal{T}$: The set of tasks
$P_{\mathcal{T}}$: The common preference profile of tasks

1 **for** $i \leftarrow 1$ *to* n **do**
2 $w \leftarrow i$th worker in $P_{\mathcal{T}}$
3 **for** $j \leftarrow 1$ *to* $|P_w|$ **do**
4 $t \leftarrow P_w(j)$; ▷ w proposes to t
5 **if** $b_t^{\mathcal{M}} \geq r_t(w)$ **then**
6 $\mathcal{M}(w) \leftarrow t$
7 $\mathcal{M}(t) \leftarrow \mathcal{M}(t) \cup \{w\}$
8 $b_t^{\mathcal{M}} \leftarrow b_t^{\mathcal{M}} - r_t(w)$
9 **break**
10 **end**
11 **end**
12 **end**
13 **return** $\mathcal{M}$

Thus, for a task t the optimal (minimum) value of δ_t is 1. Finally, the objective function can formally be defined as:

$$\text{minimize} \quad \max_{t \in \mathcal{T}} \delta_t \tag{4}$$

We will address this version of the problem using the following definition.

Definition 9 (Coalitionally α-stable matching)**.** *A matching $\mathcal{M}$ is said to be coalitionally α-stable if $\forall t \in \mathcal{T}$*

$$\delta_t \leq \alpha. \tag{5}$$

4 Stable Task Assignment

In this section, we provide the details of the proposed task assignment algorithms.

4.1 Uniform Task Assignment (UTA) Algorithm

The stable task assignment problem in uniform MCS systems has recently been investigated in [38], and an ILP-based algorithm with a $\mathcal{O}(nm2^n)$ time complexity was proposed to find pairwise stable matchings in MCS systems that are both uniform and proportional. Here, we propose *UTA algorithm* that finds pairwise stable matchings in all uniform MCS systems (i.e., proportional/non-proportional) in only $\mathcal{O}(n \log n + nm)$ time.

A pseudo-code description of UTA algorithm is given in Algorithm 2. First, apart from the set of workers and tasks, UTA algorithm takes the common preference profile of tasks ($P_{\mathcal{T}}$) as input, which is simply a sorted version of the worker set $\mathcal{W}$, in which workers with higher QoS values precede the others. Formally, it can be defined as

$$P_{\mathcal{T}} = w_{i_1}, w_{i_2}, \ldots, w_{i_n}$$

where $\forall w' = w_{i_j}, w'' = w_{i_{j+1}}$, $q_t(w') \geq q_t(w'')$, $\forall t \in \mathcal{T}$. Since we assume that the system is uniform, hence the QoS value of a worker is same for all tasks, it is in fact possible to create such a list. Then, the algorithm begins to seek the best available assignment for the workers (w) in order of their appearance in $P_{\mathcal{T}}$ (i.e., in decreasing order of their QoS). To this end, it iterates through the tasks in their preference lists (P_w) in order to find the first task in their preference lists (i.e., the most preferred) that has sufficient amount of remaining budget to hire them. If it finds such a task t for worker w, it updates the matching and the remaining budget of task t, otherwise it leaves worker w unassigned and continues the assignment process with the next worker in $P_{\mathcal{T}}$. We now show that the resulting matching will always be pairwise stable.

Theorem 3. *In uniform MCS systems, UTA algorithm always produces a pairwise stable matching.*

Proof. We will prove it by contradiction. Assume that the matching $\mathcal{M}$ returned by UTA algorithm contains at least one unhappy pair, say $\langle w, t \rangle$. Since worker w and task t form an unhappy pair, we know that they are not matched to each other and worker w prefers task t to his current partner in $\mathcal{M}$. This means when the algorithm was iterating the preference list of worker w in line 3, it has attempted to match him with task t, but could not do it due to the limited budget of task t, so that it either matched worker w with a task that come after t in P_w or left him unmatched. Let A be the partner set of task t and ρ be her remaining budget when the algorithm tried, but failed to assign worker w to her. Then, (i) $\rho < r_t(w)$. Since the algorithm matches the workers in order of their appearance in the common preference list of the tasks, we have $\{\overline{w}\} \succ_t \{w\}$, $\forall \overline{w} \in A$. Also, note that once the algorithm matches a worker and a task, it never unmatches them again. Thus, we have $A \subseteq \mathcal{M}(t)$. However, for (w, t) to be an unhappy pair, there should exist a subset $S' \subseteq \mathcal{M}(t)$ such that (ii) $r_t(w) \leq r_t(S') + b_t^{\mathcal{M}}$ and (iii) $\{w\} \succ_t S'$. From (iii), we have $\{w\} \succ_t \{\overline{w}\}$, $\forall \overline{w} \in S'$, which means all workers in S' got matched with task t after the algorithm failed to match worker w and task t, hence we have

$$\begin{aligned} &\rho \geq r_t(S') + b_t^{\mathcal{M}} \\ &\rho \geq r_t(w) && \text{(by (ii))} \\ &\rho > \rho && \text{(by (i))} \end{aligned}$$

which is a contradiction. □

The following corollary is a direct result of Theorem 3.

Corollary 3.1. *In uniform MCS systems, there always exists a pairwise stable matching.*

Note that even if an MCS system is not uniform (i.e., $\exists (w, t, t') \in \mathcal{W} \times \mathcal{T}^2$, $q_t(w) \neq q_{t'}(w)$), UTA algorithm can still produce pairwise stable matchings if it is possible to create a common preference list ($P_{\mathcal{T}}$) for tasks. In other words, if $\nexists (w, w', t, t') \in \mathcal{W}^2 \times \mathcal{T}^2$, $q_t(w) > q_t(w')$, $q_{t'}(w) < q_{t'}(w')$, UTA algorithm can still be used to find pairwise stable matchings.

Example: The instance given in Fig. 1 is a uniform MCS system, so UTA algorithm can be used to find a pairwise stable matching in this instance as follows. First, we create the common preference list of tasks as $P_{\mathcal{T}} = 1, 2, 3$ since $q_{x,y}(1) > q_{x,y}(2) > q_{x,y}(3)$. The first worker in $P_{\mathcal{T}}$ is worker 1, so the algorithm starts the matching process with him. Since the first and only task in his preference list, task x, has enough budget to hire him (the preference lists of users are given in Table 2), it assigns worker 1 to task x, and updates the remaining budget of task x as $7-5=2$. The next worker in $P_{\mathcal{T}}$ is worker 2, who also prefers task x to task y, but task

x does not have enough budget to hire him ($2 < 4$), so the algorithm tries to match him with task y. Task y has enough budget to hire worker 2, so they get matched and the remaining budget of task y becomes $5 - 4 = 1$. Worker 3 is the last worker in $P_{\mathcal{T}}$. However, neither task x nor task y has sufficient remaining budget to hire him, so he will be left unmatched. Thus, the final matching will be ($x \leftrightarrow 1$, $y \leftrightarrow 2$), and it can easily be checked that it does not contain any unhappy pairs and hence it is pairwise stable.

Running time. Forming the common preference profile of tasks ($P_{\mathcal{T}}$) requires to sort the workers according to their QoS values and thus takes $\mathcal{O}(n \log n)$. Then, since the first for loop will iterate n times and the second will iterate at most m times (i.e., $|P_w| \leq m, \forall w \in \mathcal{W}$), it is straightforward to see that the worst-case running time of UTA algorithm is $\mathcal{O}(n \log n + nm)$.

4.2 Pairwise Stable Task Assignment (PSTA) Algorithm

PSTA algorithm is a pseudo-polynomial time algorithm that, unlike UTA algorithm, can be run in any type of MCS system, and aims to produce matchings with as little pairwise instability as possible (which is why it is named as *Pairwise*-STA). In fact, in Theorem 4, we will show that it always produces pairwise stable matchings in proportional MCS systems. Besides, in non-proportional MCS systems where a pairwise stable matching may not exist, it manages to produce matchings with almost optimal pairwise stability as it will be shown in Section 5.

The details of PSTA algorithm are given in Algorithm 3. It follows the classic deferred acceptance mechanism [7] but updates the set of workers assigned to a task optimally from the set of current workers and the newly proposing worker. It keeps a stack of unmatched workers that still have tasks to propose to, pops them one by one (line 3) and lets them (worker w) propose to the next task (task t) in their preference list (line 5). If task t has enough remaining budget, the algorithm directly matches them (lines 8-10). Otherwise, it finds the most favorable worker set (S_{max}) for task t among the workers in her current partner set and worker w within her budget constraint (lines 12-13), assigns that worker set as the new partner set of task t (lines 14-17) and pushes the remaining workers back onto the stack after setting them free (lines 18-20). If the partner set of task t has not changed, worker w will be the only worker to be pushed onto the stack, in which case we say that task t rejected the proposal of worker w. This continues until there is no unmatched worker that still has a task that he can propose to in his preference list. In the following theorem, we prove that the resulting matching is guaranteed to be pairwise stable if the MCS system is proportional.

Algorithm 3: PairwiseStableTaskAssignment($\mathcal{W}$,$\mathcal{T}$)

Input: $\mathcal{W}$: The set of workers
$\mathcal{T}$: The set of tasks

1 *Stack.push($\mathcal{W}$)*
2 **while** *Stack is not empty* **do**
3 $w \leftarrow$ *Stack.pop()*
4 **if** P_w *is not empty* **then**
5 $t \leftarrow P_w(1)$; ▷ w proposes to t
6 $P_w \leftarrow P_w \setminus \{t\}$
7 **if** $b_t^{\mathcal{M}} \geq r_t(w)$ **then**
8 $\mathcal{M}(w) \leftarrow t$
9 $\mathcal{M}(t) \leftarrow \mathcal{M}(t) \cup \{w\}$
10 $b_t^{\mathcal{M}} \leftarrow b_t^{\mathcal{M}} - r_t(w)$
11 **else**
12 $S \leftarrow \mathcal{M}(t) \cup w$
13 $S_{max} \leftarrow$ *solve01Knapsack*$(b_t, R_t(S), Q_t(S))$
14 $\mathcal{M}(t) \leftarrow S_{max}$, $b_t^{\mathcal{M}} \leftarrow b_t - r_t(S_{max})$
15 **if** $w \in S_{max}$ **then**
16 $\mathcal{M}(w) \leftarrow t$
17 **end**
18 **foreach** $w' \in S \setminus S_{max}$ **do**
19 $\mathcal{M}(w') \leftarrow \emptyset$, *Stack.push(w')*
20 **end**
21 **end**
22 **end**
23 **end**
24 **return** $\mathcal{M}$

Theorem 4. *In proportional MCS systems, PSTA algorithm always produces a pairwise stable matching.*

Proof. We will prove it by contradiction. Assume that in a proportional MCS system (without loss of generality, let $q_t(w) = r_t(w), \forall w, t$), PSTA algorithm produces a matching $\mathcal{M}$ that contains at least one unhappy pair, say $\langle w, t \rangle$, which either means that task t rejected worker w's proposal and they never got matched, or that task t accepted worker w's proposal, but then discarded him (possibly along with a set of workers) from her partner set to accept the proposal of another worker with a higher QoS. Let A and ρ denote task t's partner set and remaining budget at the time task t and worker w broke up by one of these two cases (i.e., rejected or discarded), respectively. Then, we have

$$\nexists S \subseteq A : r_t(S) < r_t(w), r_t(w) \leq r_t(S) + \rho \tag{6}$$

because if there were such a subset S, the 0-1 knapsack solution would include worker w instead of S, and w would not get rejected/discarded.

We define *a node* as a tuple (x, y), where x and y are the label and length of the node, respectively. Let $A_0 = \{(w', r_t(w')) : w' \in A\}$, so each node in A_0 corresponds to a worker in A. Also, let $l(S)$ be the total length of the nodes in S. Then, from (6), we have

$$\nexists S \subseteq A_0 : l(S) < r_t(w), r_t(w) - l(S) \leq \rho \tag{7}$$

Note that task t will discard a group G of workers from her partner set only when she is proposed by a worker w' who has a higher total reward than G and will not violate her budget constraint when replaced by G. After the break up of worker w and task t, whenever such a change (say ith change) occurs in the partner set of task t, we create A_i from A_{i-1} as follows:

1) $A_i \leftarrow A_{i-1}$.
2) Let K be the set of nodes in A_i that have the same label with any worker in G.
3) Change the labels of all nodes in K as w'.
4) Create a new node $(w', r_t(w') - r_t(G))$ (note that $r_t(G) = l(K)$) and add it to A_i.

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/TMC.2020.2997280, IEEE Transactions on Mobile Computing

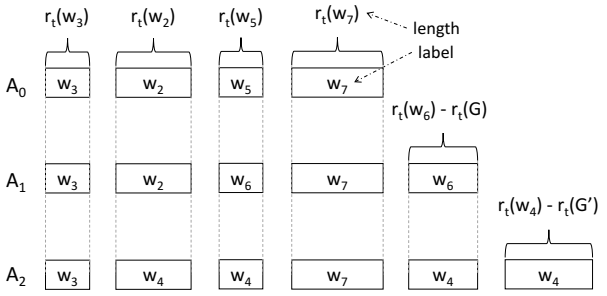


Fig. 2: An example illustrating the process used in the proof of Theorem 4. A_0 is created right after task t and worker w broke up, so the partner set of task t is $\{w_3, w_2, w_5, w_7\}$ at that time. A_1 is created after the first change in the partner set of task t, which is the substitution of $G = \{w_5\}$ with w_6, and A_2 is created after the second change in the partner set of task t, which is the substitution of $G' = \{w_2, w_6\}$ with w_4. Note that the length of the nodes in A_0 are always preserved throughout the process.

In other words, we add the new worker to A_i by dividing it into several nodes so that the node lengths in A_{i-1} are preserved. An example is provided in Fig. 2 to illustrate this process.

Let c be the number of changes occurred in the partner set of task t since her break up with worker w until the end. Then, the last node set created, A_c, can be partitioned into two sets A'_0 and B, where A'_0 contains $|A_0|$ nodes that have exactly the same lengths with the nodes in A_0, but possibly have different labels, and B is the set of newly created nodes such that (i) $l(B) + b_t^{\mathcal{M}} = \rho$. Since we assumed that (w, t) form an unhappy pair in the final matching $\mathcal{M}$, there should exist a subset $S' \subseteq \mathcal{M}(t)$ such that $r_t(w) > r_t(S')$ and $r_t(w) \leq r_t(S') + b_t^{\mathcal{M}}$. As A_c has a distinct set P of nodes that jointly correspond to each worker w' in $\mathcal{M}(t)$ (i.e., $l(P) = r_t(w')$), there should also exist a subset $\overline{S} \subseteq A'_0 \cup B$ such that (ii) $r_t(w) > l(\overline{S})$ and (iii) $r_t(w) \leq l(\overline{S}) + b_t^{\mathcal{M}}$. Then, we have

$$\begin{aligned} r_t(w) - l(\overline{S} \cap A'_0) &\leq l(\overline{S} \cap B) + b_t^{\mathcal{M}} && \text{(by (iii))} \\ \rho &< l(\overline{S} \cap B) + b_t^{\mathcal{M}} && \text{(by (7) and (ii))} \\ \rho &< \rho && \text{(by (i))} \end{aligned}$$

which is a contradiction. □

A significant result of Theorem 4 is the following corollary.

Corollary 4.1. *In proportional MCS systems, there always exists a pairwise stable matching.*

In the following theorem, we show that pairwise stability ensures a certain degree of coalitional stability in proportional MCS systems.

Theorem 5. *In proportional MCS systems, a pairwise stable matching is coalitionally 2-stable.*

Proof. We prove by contradiction. Let $\mathcal{M}$ be a pairwise stable matching in a proportional MCS system and $\langle S, t \rangle$ be an unhappy coalition in $\mathcal{M}$ such that

$$r_t(S \cup (\mathcal{M}(t) \setminus S')) > 2r_t(\mathcal{M}(t)) \tag{8}$$

where $S' \subseteq \mathcal{M}(t)$ satisfies $r_t(S) > r_t(S')$ and (i) $r_t(S) \leq b_t^{\mathcal{M}} + r_t(S')$. Thus, $\langle S, t \rangle$ breaks the coalitional 2-stability of $\mathcal{M}$ according to (5). First, note that if (ii) $r_t(\mathcal{M}(t)) \geq b_t/2$, we would have

$$\begin{aligned} r_t(S) + r_t(\mathcal{M}(t)) - r_t(S') &> 2r_t(\mathcal{M}(t)) && \text{(by (8))} \\ r_t(\mathcal{M}(t)) + b_t^{\mathcal{M}} &> 2r_t(\mathcal{M}(t)) && \text{(by (i))} \\ r_t(\mathcal{M}(t)) + b_t^{\mathcal{M}} &> b_t && \text{(by (ii))} \\ b_t &> b_t \end{aligned}$$

which is false. So, we have $r_t(\mathcal{M}(t)) < b_t/2$. Let w be any worker in S. If $r_t(w) \leq r_t(\mathcal{M}(t))$, then $\mathcal{M}$ is not pairwise stable because task t can add worker w to her partner list without removing anyone as $r_t(w) + r_t(\mathcal{M}(t)) < b_t$. Thus, we have $r_t(w) > r_t(\mathcal{M}(t))$, which also implies that $\mathcal{M}$ is not a pairwise stable matching as task t can simply replace $\mathcal{M}(t)$ with worker w to obtain a better partner set within her budget constraint (i.e., $r_t(w) \leq r_t(S) \leq b_t^{\mathcal{M}} + r_t(S') \leq b_t$). □

From Theorem 3, Theorem 4 and Theorem 5, we obtain the following corollaries.

Corollary 5.1. *In MCS systems that are both proportional and uniform, UTA algorithm always returns coalitionally 2-stable matchings.*

Corollary 5.2. *In proportional MCS systems, PSTA algorithm always returns coalitionally 2-stable matchings.*

Example: We run PSTA algorithm on the same MCS instance illustrated in Fig. 1. To make things slightly different, we assume that the workers are pushed onto the stack in line 1 in increasing order of their identifiers so that the first worker that is popped in line 3 is worker 3. As shown in Table 2, the first task in the preference list (P_3) of worker 3 is task y, so he first proposes to her (task y gets removed from P_3). Task y has enough budget, so worker 3 and task y get matched to each other and the remaining budget of task y becomes 2 (line 8-10). The next worker popped from the stack is worker 2, whose first preference is task x. Thus, worker 2 proposes to task x (task x gets removed from P_2). Since task x also has enough budget, she gets matched with worker 2, which reduces her remaining budget to 3. Next, worker 1 gets popped and proposes to the only task in his preference list: task x (task x gets removed from P_1, so $P_1 = \emptyset$). Task x does not have enough remaining budget to hire worker 1, so the algorithm finds the best set of workers among the workers in $\mathcal{M}(x) \cup \{1\} = \{1, 2\}$ that does not exceed the budget limit of task x by solving the corresponding 0-1 knapsack problem (line 13). The subset $\{1\}$ provides the highest QoS without violating the budget constraints, so worker 1 and task x will get matched to each other, and worker 2 will be set free and pushed onto the stack. In the next step, worker 2 will be popped and propose to task y (task y gets removed from P_2, so $P_2 = \emptyset$). Task y does not have sufficient remaining budget, but the algorithm, after solving the knapsack problem, will replace her current partner, worker 3, with worker 2 as this will increase the total QoS task y gets. Consequently, it will set worker 3 free and push him onto the stack. Then, it will pop him from the stack and let him propose to task x (task x gets removed from P_3, so $P_3 = \emptyset$). The budget of task x is not

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/TMC.2020.2997280, IEEE Transactions on Mobile Computing

adequate to hire worker 3, and replacing his current partner is also not beneficial, hence worker 3 will be pushed onto the stack, again. When popped next time, since there does not remain any other task for worker 3 to propose to in his preference list, he will not be pushed onto the stack again. This will leave the stack empty, so the matching ($x \leftrightarrow 1$, $y \leftrightarrow 2$) will be returned by the algorithm, which is the same pairwise stable matching found by UTA algorithm.

Running time. Note that since the preference list of a worker (w) shrinks in size by 1 every time he is popped from the stack (line 6) until his preference list becomes empty (after which he will not be pushed onto the stack ever again), he can be pushed onto the stack at most $\mathcal{O}(m)$ times as $|P_w| \leq m$. Thus, the while loop in line 2 will iterate at most $\mathcal{O}(nm)$ times. Since the most costly operation in each iteration is solving the 0-1 knapsack problem, which takes $\mathcal{O}(n\beta)$ where $\beta = \max_{t \in \mathcal{T}} b_t$, the time complexity of PSTA algorithm is $\mathcal{O}(n^2 m \beta)$.

4.3 Heuristic Algorithm

Heuristic algorithm is a task-oriented, pseudo-polynomial time algorithm that can also be run in any type of MCS system and is designed in a way that the tasks in the system take turns at modifying the matching according to their preferences. That is, each task t, in her turn, changes her partner set to the best feasible set of workers among all the workers that are already in her partner set, or that prefer herself to their current assignments. Thus, Heuristic algorithm ensures that there is no unhappy coalition $\langle S, t \rangle$ for any $S \subseteq \mathcal{W}$ immediately after the turns of task t. It can, hence, be expected that as we increase the number of iterations/turns, there will be fewer unhappy coalitions, which will improve the coalitional stability of the matching.

The outline of Heuristic algorithm is provided in Algorithm 4. In each of the k iterations, the algorithm goes through all tasks (t) in the system (line 2) and first finds all workers that are either already matched with task t or would be better off with task t compared to their current partners (lines 3-8). Then, among these workers, it identifies the set S_{max} of workers that require a total reward of less than b_t and provide as large total QoS as possible for task t by solving the corresponding knapsack problem (line 9). Finally, it sets the workers that are presently matched with task t, but are not in S_{max} free (lines 10-12), and matches task t and the workers in S_{max} with each other after removing these workers from the partner sets of the tasks with whom they were previously matched (lines 13-18).

The fact that a task is perfectly happy (i.e., has a dissatisfaction ratio of 1) right after her turns indicates that the task that is considered the latest in the for loop in line 2 will be perfectly happy in the end, as well. This nice property of Heuristic algorithm can be used to make a different task requester happy at each (e.g., hourly, daily) assignment cycle, and to ensure that all task requesters become perfectly happy with their assignments periodically. Another useful property of Heuristic algorithm is that it allows to explore different feasible matchings that are shaped by the preference profile of a different task, which will become clearer in the toy example provided below.

Algorithm 4: Heuristic Approach($\mathcal{W}$, $\mathcal{T}$)

Input: $\mathcal{W}$: The set of workers
$\mathcal{T}$: The set of tasks
k: The number of iterations

```
1  for i ← 1 to k do
2      foreach t ∈ T do
3          S ← M(t)
4          foreach w ∈ W do
5              if t ∈ P_w and t ≻_w M(w) then
6                  S ← S ∪ {w}
7              end
8          end
9          S_max ← solve01Knapsack(b_t, R_t(S), Q_t(S))
10         foreach w' ∈ M(t) \ S_max do
11             M(w') ← ∅
12         end
13         foreach w' ∈ S_max do
14             let t' denote M(w')
15             M(t') ← M(t') \ w'
16             M(w') ← t
17         end
18         M(t) ← S_max, b_t^M ← b_t − r_t(S_max)
19     end
20 end
21 return M
```

Example: We once again utilize the MCS instance given in Fig. 1 to show how Heuristic algorithm functions. Assume that the for loop in line 2 iterates through the tasks in order of task x and task y. In the first iteration, since all workers are unmatched, task x will be assigned to the best (i.e., with the highest total QoS) subset of workers in $\{1, 2, 3\}$ with a total reward of less than 7, which is $\{2, 3\}$:

$$\mathcal{M} : x \leftrightarrow \{2, 3\}, y \leftrightarrow \emptyset$$

In this matching. worker 3 is the only one that prefers task y to task x, so when it is task y's turn, she will directly be matched with worker 3:

$$\mathcal{M} : x \leftrightarrow \{2\}, y \leftrightarrow \{3\}$$

In the next iteration, task x will be assigned to the best worker set from $\{1, 2\}$, which are the workers that are either matched with task x (i.e., worker 2) or prefer task x to their current partners (i.e., worker 1). Since her budget does not allow to hire both, worker 2 will be replaced by worker 1 who provides a higher QoS:

$$\mathcal{M} : x \leftrightarrow \{1\}, y \leftrightarrow \{3\}$$

At this point, since worker 2 prefers task y to being unassigned, task y will be assigned to the best feasible worker set from $\{2, 3\}$ in her turn, which is $\{2\}$:

$$\mathcal{M} : x \leftrightarrow \{1\}, y \leftrightarrow \{2\}$$

In this matching, both worker 2 and worker 3 prefers task x to their current partners, so the algorithm will assign task x the best feasible worker set from $\{1, 2, 3\}$, which is {2,3}:

$$\mathcal{M} : x \leftrightarrow \{2, 3\}, y \leftrightarrow \emptyset$$

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/TMC.2020.2997280, IEEE Transactions on Mobile Computing

Note that this is exactly the same as the first matching we obtained above. In fact, after this point, the algorithm will repeatedly generate the same matchings, and return the matching ($x \leftrightarrow \{2\}$, $y \leftrightarrow \{3\}$) if k is odd, and the matching ($x \leftrightarrow \{1\}$, $y \leftrightarrow \{2\}$), otherwise. The former is the optimal matching in terms of coalitional stability, as there does not exist any coalitionally stable matching in this instance (Theorem 2) and this matching contains only one coalitionally unhappy pair (worker 1 and task x). On the other hand, the latter is the same matching as the one found by UTA and PSTA algorithms and is the optimal matching in terms of pairwise stability.

Running time. The two outermost loops in line 1 and 2 will iterate k and m times, respectively. Similar to PSTA algorithm, solving the 0-1 knapsack instance in line 9 takes $\mathcal{O}(n\beta)$ where $\beta = \max_{t\in\mathcal{T}} b_t$, and is the most costly operation within the for loop in line 2. This makes the total running time of Heuristic algorithm $\mathcal{O}(knm\beta)$.

5 Simulation Results

In this section, we evaluate the performance of the proposed algorithms in different types of MCS systems.

5.1 Settings

Similar to previous work [14], [39], we utilize a taxi trip dataset [40] in a city (i.e., New York City (NYC)) to have a rather realistic geographic distribution of workers and tasks. Specifically, we randomly select a day in 2015 and then create a worker at the most recent drop-off location of each taxi that has become available between 1-2 pm on the selected day, and a task at the pick-up location of each passenger that has demanded a taxi in the next hour of the same day. Then, we use random-sampling to obtain the worker and task sets of certain size based on the experiment requirements.

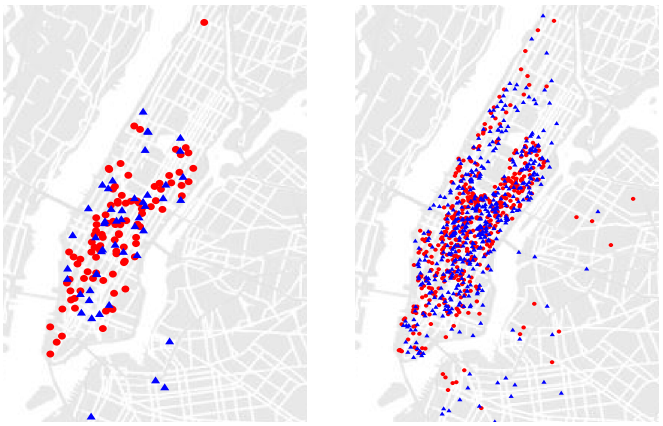
Note that each of the four types of MCS systems (i.e., proportional (P.) and non-uniform (N.U.), non-proportional (N.P.) and non-uniform, proportional and uniform (U.), non-proportional and uniform) necessitates different QoS and reward settings. Thus, we generate a unique scenario for each MCS system by integrating this information on top of the geographical information.

As the default setup for all scenarios, we sample $n = 100$ workers and $m = 50$ tasks (an instance is illustrated in Fig. 3), and randomly assign a budget for each task between $B_{min} = 100$ and $B_{max} = 1000$. Given the distance d between a worker w and a task t, we let $c_t(w) = d \times C$, where $C = 20$ denotes the cost per kilometer. Below, we describe the typical settings for each scenario.

- **Proportional (P.) and uniform (U.)**: We assign a unique QoS value v to each worker w randomly from $[1, 200]$ and let $q_t(w) = v, \forall t \in \mathcal{T}$.
  Then, the rewards are set as
  $$r_t(w) = \begin{cases} \theta_t q_t(w) & \text{if } \theta_t q_t(w) \le b_t \\ 0 & \text{otherwise} \end{cases}$$
  where θ_t is randomly selected from $[1, 5]$ for each task t.
- **Proportional (P.) and non-uniform (N.U.)**: Given a worker-task pair (w, t), we randomly assign the reward $r_t(w)$ from $[1, b_t]$ (we also examine the cases where reward values are assigned from $[1, b_t/2]$ and $[b_t/2, b_t]$), and let $q_t(w) = r_t(w)/\theta_t$, where θ_t is set as in the previous scenario, yet its value is actually arbitrary for all the algorithms considered in this scenario, unlike the previous scenario where it is unarbitrary for UTA algorithm.
- **Non-proportional (N.P.) and uniform (U.)**: For this scenario, the QoS information is produced exactly as it is in the proportional and uniform scenario. The only difference is that for each worker-task pair (w, t), the reward $r_t(w)$ is assigned randomly from $[1, b_t]$.
- **Non-proportional (N.P.) and non-uniform (N.U.)**: Given a worker-task pair (w, t), we randomly assign the reward $r_t(w)$ from $[1, b_t]$ and $q_t(w)$ from $[1, 200]$.

Given the settings described above, the preference profiles of workers and tasks can be determined by (1) and (2), respectively. (Yet it should be noted that in practice none of the algorithms requires tasks to form their preference lists, which would take $\mathcal{O}(n2^n)$ time and memory.)

Lastly, we run the simulations 100 times with a different user set in each run and present the averaged results.



(a) (b)

Fig. 3: Distribution of workers (circles) and tasks (triangles) on the NYC map with different sampling ratios: (a) 100 workers, 50 tasks; (b) 500 workers, 500 tasks.

TABLE 3: Time complexities of all algorithms considered in the simulations (* indicates the algorithms proposed in this paper).

Algorithm	Time complexity
UTA*	$\mathcal{O}(n \log n + nm)$
PSTA*	$\mathcal{O}(n^2 m\beta)$
Heuristic*	$\mathcal{O}(knm\beta)$
SJA	$\mathcal{O}(nm2^n)$
ϕ-STA	$\mathcal{O}(mn \log(mn))$
θ-STA	$\mathcal{O}(mn \log(mn))$

5.2 Algorithms in Comparison

We compare the performance of our algorithms with the following algorithms proposed in [9] and [30] (see Table 3 for a comparison of the time complexities of all algorithms).

- *Stable Job Assignment (SJA)*: This ILP-based algorithm [9] produces pairwise stable matchings solely in proportional and uniform MCS systems.

TABLE 4: Mobile crowdsensing scenarios and corresponding applicable algorithms ($*$ indicates the algorithm is applicable but has a very poor performance since it is not specifically designed for that scenario).

MCS Type	UTA	PSTA	Heuristic	SJA	ϕ-STA	θ-STA
P. & U.	✓	✓	✓	✓	✓	*
P. & N.U.		✓	✓		✓	*
N.P. & U.	✓	✓	✓			✓
N.P. & N.U.		✓	✓			✓

- *ϕ-Stable Task Assignment (ϕ-STA)*: Proposed in [30], this approximation algorithm produces matchings that are guaranteed to be coalitionally ϕ-stable in proportional matching markets, where ϕ (≈ 1.618) denotes the golden ratio. In this algorithm, tasks (t) simply run the ϕ-approximation algorithm for the single bin removable online knapsack problem proposed in [41] to decide whether to accept (and discard some other workers if needed) or reject the proposing workers (w) using the rewards $r_t(w)$ as the weights of items and b_t as the size of the knapsack. The running time of this algorithm is $\mathcal{O}(mn\log(mn))$.
- *θ-Stable Task Assignment (θ-STA)*: This approximation algorithm is also proposed in [30] and generates coalitionally θ-stable matchings in general matching markets, where

$$\theta = \frac{1}{1 - \max_{w\in\mathcal{W}, t\in\mathcal{T}} \frac{r_t(w)}{b_t}}.$$

  It follows the classic deferred acceptance mechanism: workers make the proposals, tasks (t) that have available budget accept the incoming proposals and those that do not have available budget temporarily add the proposing worker to their partner set and then discard the workers (w') with the lowest $\frac{q_t(w')}{r_t(w')}$ ratio until the sum of rewards to be paid to the remaining workers is less than or equal to b_t. As it is shown in Table 4, although it can be run in all types of MCS systems, we do not provide results for this algorithm in proportional MCS systems due to its unpredictable and mostly poor performance in these systems. This is because it randomly selects the workers to be discarded in proportional settings as all workers (w') have the same $\frac{q_t(w')}{r_t(w')}$ ratio for each task t. The time complexity of this algorithm is also $\mathcal{O}(mn\log(mn))$.

Note that neither ϕ-STA nor θ-STA has a performance guarantee in terms of pairwise stability.

5.3 Performance metrics

We utilize the following performance metrics in the evaluations.

- *Overall user happiness*: This is calculated as

$$100 \times \left(1 - \frac{\text{\# of coalitionally unhappy pairs}}{\text{\# of all matchable worker-task pairs}}\right)$$

  and expresses the overall user happiness based on the instability of the matching. Thus, it is the main metric that defines the performance of the algorithms.
- *Outward user happiness*: This is calculated as

$$100 \times \left(1 - \frac{\text{\# of unhappy pairs}}{\text{\# of all matchable worker-task pairs}}\right)$$

  and quantifies the outward user happiness based on the one-dimensional instability of the matching. The reason it is called *outward* is that compared to coalitionally unhappy pairs, unhappy pairs are easier to notice for users, and the users forming an unhappy pair have a stronger incentive to deviate from (a subset of) their partners to each other as they do not need a collective agreement that involves other users (unlike coalitionally unhappy pairs).
- *Maximum dissatisfaction ratio*: This is the dissatisfaction of the task in the unhappy coalition with the largest incentive to deviate from the current matching, which is formally defined as

$$\delta_{max} = \max_{t\in\mathcal{T}} \delta_t.$$

  Given the maximum dissatisfaction ratio δ_{max} of a matching $\mathcal{M}$, we can say that $\mathcal{M}$ is coalitionally δ_{max}-stable and is not coalitionally $(\delta_{max}-\epsilon)$-stable for any positive real ϵ. If a matching does not have any unhappy coalition, then $\delta_{max} = 1$ by definition.
- *Running time*: We also compare the algorithms with respect to their running time, which might be critical for MCS systems with strict time constraints.

5.4 Results

We first look at the performance of the proposed algorithms in proportional and non-uniform scenario. Fig. 4 shows the performance of algorithms with different number of workers. First, note that Heuristic algorithm (which is run with $k = 3$ as default) usually performs the best and produces optimal assignments in terms of both overall and outward user happiness when the number of workers is larger than 200. It is interesting that it achieves very similar overall and outward user happiness scores, which indicates that it yields matchings in which most of the coalitionally unhappy pairs are also unhappy pairs. Second, we see that despite having a better upper bound (UB) in terms of maximum dissatisfaction ratio (i.e., the upper bounds for ϕ-STA and PSTA are, respectively, ϕ (≈ 1.618) and 2, while Heuristic algorithm is unbounded), ϕ-STA mostly performs much worse than our algorithms. Besides, its performance gets worse as the number of workers increases and it even produces matchings with as low as 10% overall user happiness. Since the system is proportional, PSTA algorithm always achieves 100% outward user happiness (Theorem 4), but we still include it in all figures for completeness. Also, it outperforms Heuristic algorithm when the number of workers is small and generally achieves a maximum dissatisfaction ratio that is much smaller than its upper bound and very close to the optimal value (i.e., 1).

Fig. 5 shows the performance of algorithms with varying number of tasks. As for the performance of Heuristic algorithm, we see a trend that is similar to what we have seen in Fig. 4. It performs worse when the number of workers and tasks are close to each other. On the other hand, ϕ-STA and PSTA algorithms always have a better performance with increased number of tasks. This is because they are worker-oriented algorithms where the proposals are made by workers, so the increase in the number of tasks reduces the competition among workers and results in improved

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/TMC.2020.2997280, IEEE Transactions on Mobile Computing

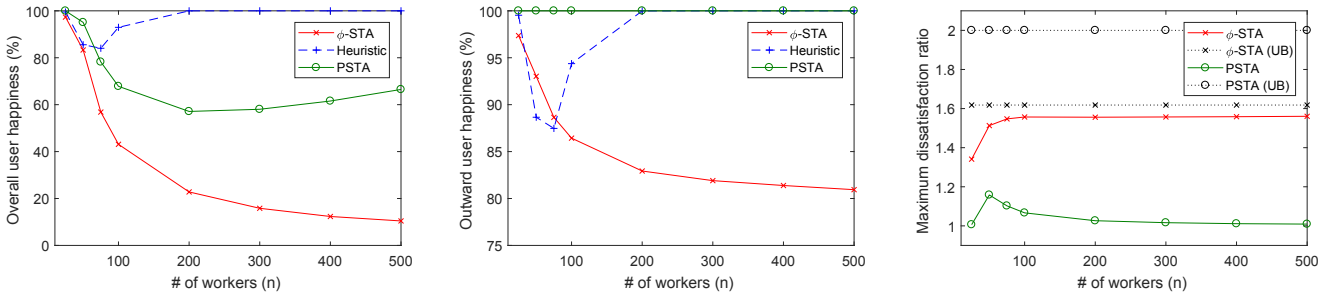


Fig. 4: Performance comparison of algorithms in proportional and non-uniform scenario with varying number of workers ($m = 50$ tasks).

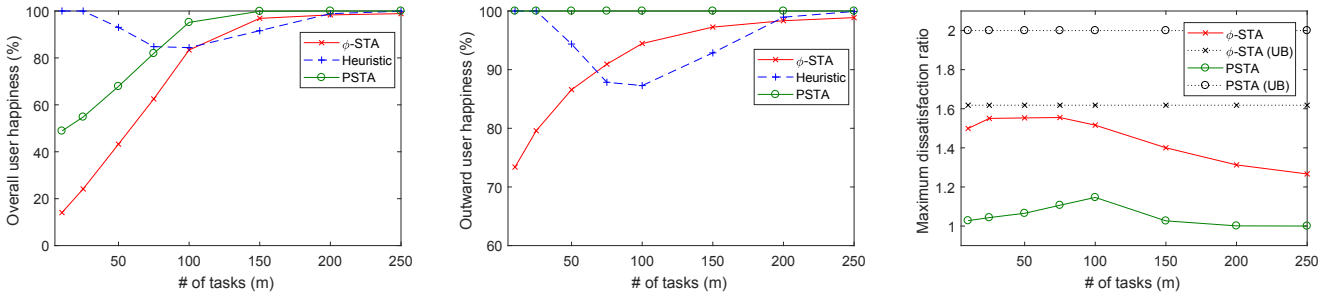


Fig. 5: Performance comparison of algorithms in proportional and non-uniform scenario with varying number of tasks ($n = 100$ workers)

user happiness. Here, PSTA always performs better than ϕ-STA, and it also outperforms Heuristic algorithm when the number of tasks get larger than that of workers.

In Fig. 6a, we show the performance of Heuristic algorithm when it is run with different k values. We see that even with a few number of iterations, it achieves about 95% overall user happiness, and that increasing the number of iterations continues to improve the performance even up until 100 iterations, but it may not be worth the increase to be seen in the runtime, which is linear to the number of iterations.

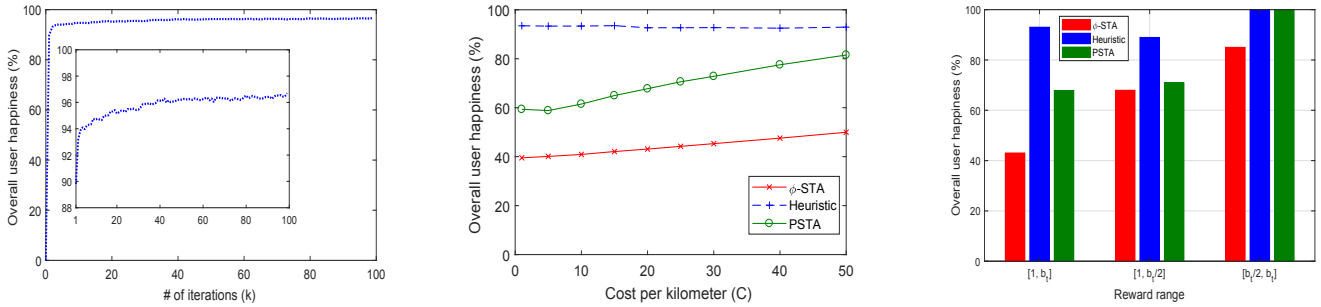
In Fig. 6b, we look at the impact of C (cost per kilometer) on the performance of the algorithms. Note that an increased C value can be interpreted as having very selective workers who do not even find most of the tasks acceptable. Thus, it deescalates the competition among workers and results in an improvement in the performance of worker-oriented algorithms (ϕ-STA and PSTA). Conversely, it escalates the competition among tasks as there will be less number of eligible workers for each task, so the performance of task-oriented Heuristic algorithm gets slightly worse. The similar outcomes are also seen in Fig. 6c where we look at the impact of reward ranges. When we lower the reward range from $[1, b_t]$ to $[1, b_t/2]$, there will be fewer workers eligible for each task, so the performance of ϕ-STA and PSTA algorithms gets better, while that of Heuristic algorithm gets worse due to the same reasons pointed out above. However, when the reward range is made $[b_t/2, b_t]$, the performance of all algorithms get better (PSTA and Heuristic algorithms even achieve optimal overall user happiness) because tasks can now hire at most 2 workers, making it an almost competition-free setting for both workers and tasks.

Next, we analyze the performance of algorithms in non-proportional and non-uniform scenario in Fig. 7. Note that since the system is non-proportional, PSTA algorithm fails to achieve perfect outward user happiness most of the times, but still manages to outperform the others in terms of outward user happiness. The performance of Heuristic algorithm seems similar to its performance in proportional and non-uniform scenario: it achieves higher than 85% overall/outward user happiness regardless of the number of workers/tasks and performs the worst when the number of workers and tasks are similar. However, in this scenario, our algorithms are usually outperformed by θ-STA algorithm in terms of overall user happiness. In fact, θ-STA algorithm has a quite reliable performance in this scenario as its overall user happiness score never drops below 95%.

Fig. 8 and 9 show the performance comparison of algorithms in proportional and uniform scenario for varying number of workers and tasks, respectively. Here, Heuristic algorithm always outperforms all the other algorithms in terms of overall user happiness by steadily achieving very close to optimal scores ($\geq$ 97%) regardless of worker and task counts. On the other hand, ϕ-STA algorithm always has the poorest performance in terms of all metrics considered here. Since the system is both proportional and uniform, UTA and PSTA algorithms are guaranteed to achieve perfect outward user happiness due to Theorems 3 and 4, respectively. It is remarkable that the increase in the number of tasks improves the performance of UTA and PSTA algorithms in terms of overall user happiness, while it has a detrimental effect on them in terms of maximum dissatisfaction ratio.

In Fig. 10, we look at the performance of algorithms in non-proportional and uniform scenario. We first observe that UTA algorithm generally has the worst performance in terms of overall user happiness, yet it is the only algorithm that ensures a perfect outward user happiness. Second, 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/TMC.2020.2997280, IEEE Transactions on Mobile Computing



(a) (b) (c)

Fig. 6: (a) Performance of Heuristic algorithm with increasing number of iterations; (b-c) Performance comparison of algorithms with varying cost per kilometer values and reward ranges (all in proportional and non-uniform scenario with $m = 50$ tasks and $n = 100$ workers).

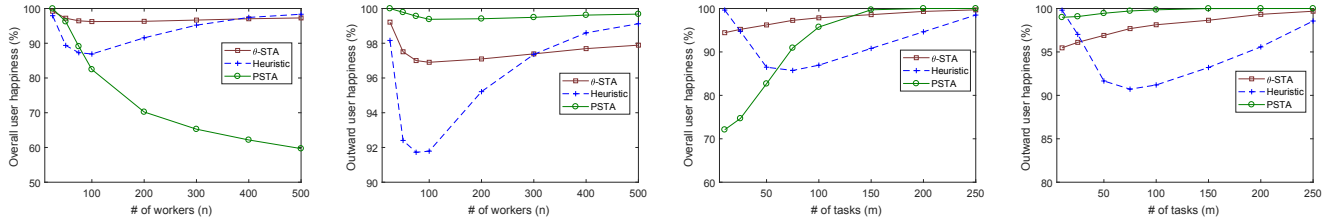


Fig. 7: Performance comparison of algorithms in non-proportional and non-uniform scenario with varying number of workers and tasks.

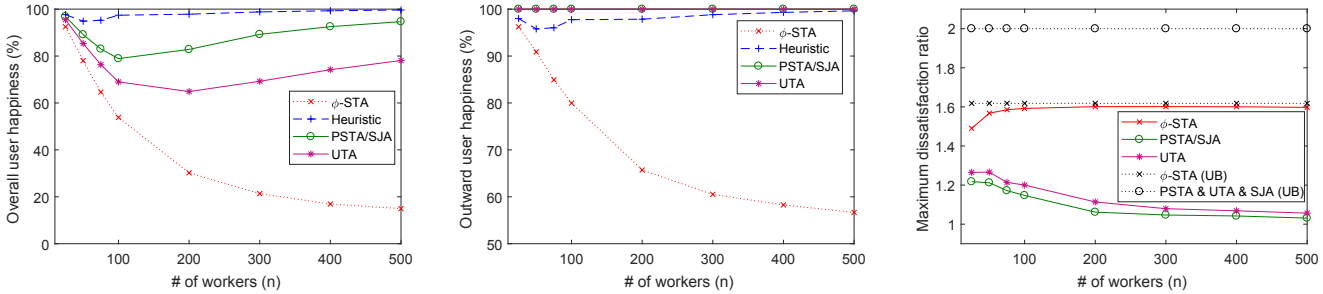


Fig. 8: Performance comparison of algorithms in proportional and uniform scenario with varying number of workers ($m = 50$ tasks).

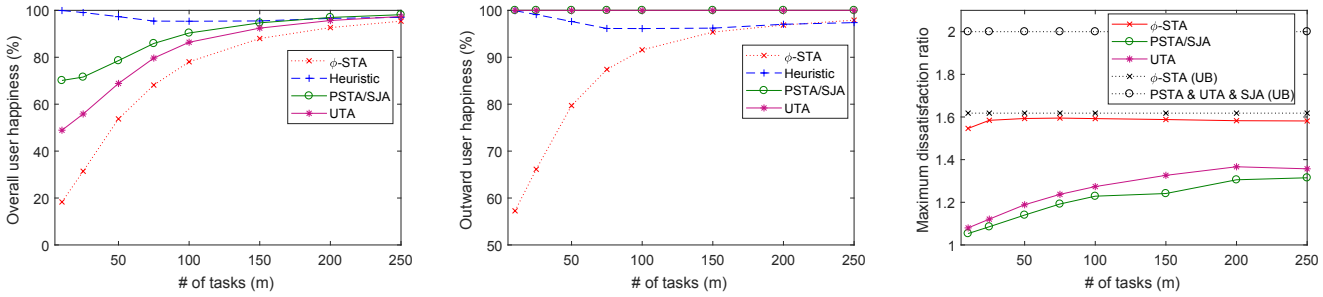


Fig. 9: Performance comparison of algorithms in proportional and uniform scenario with varying number of tasks ($n = 100$ workers).

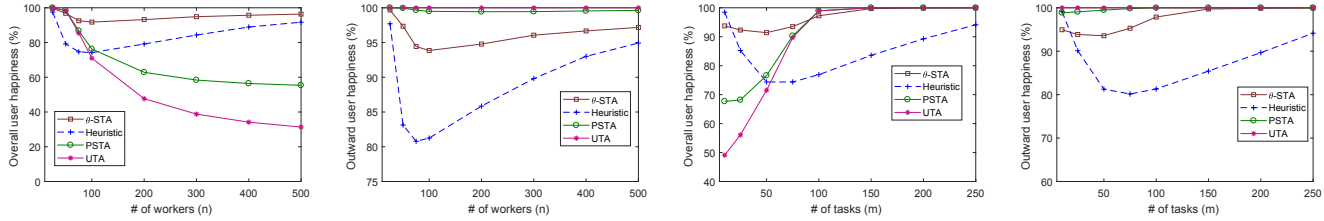
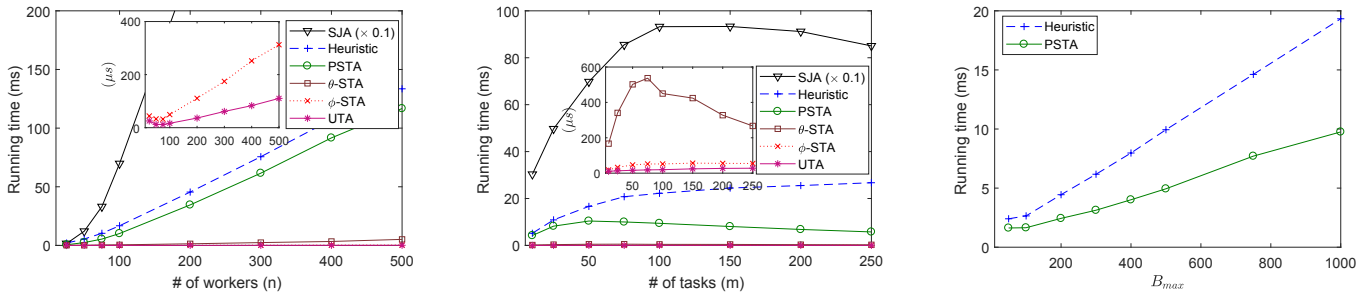


Fig. 10: Performance comparison of algorithms in non-proportional and uniform scenario with varying number of workers and tasks.

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/TMC.2020.2997280, IEEE Transactions on Mobile Computing



(a) (b) (c)

Fig. 11: (a-b) Comparison of algorithms in terms of running time; (c) Impact of B_{max} on the running time of Heuristic and PSTA algorithms.

performance of Heuristic algorithm is significantly worse in this scenario compared to that in the others. In fact, this is the only scenario where it achieves less than 85% overall user happiness. Similar to the non-proportional and non-uniform scenario, θ-STA algorithm usually manages to deliver a higher overall user happiness score than the others when there are more workers than tasks, but its performance is also worse in this scenario. Besides, it is significantly outperformed by PSTA and UTA algorithms in terms of outward user happiness for the most part.

Lastly, in Fig. 11a-b, we compare the running time of all algorithms. We only provide the results for the proportional and uniform scenario as all of the algorithms can be run in this scenario and those that are also run in different scenarios have an almost indistinguishable running time in all scenarios they have been used with. First, note that the objective of SJA algorithm is to obtain assignments with perfect outward user happiness in proportional and uniform systems. Our UTA algorithm achieves the same in not only proportional and uniform systems, but also in non-proportional and uniform systems in an extremely shorter time (by a few orders of magnitude). Furthermore, it has the lowest running time among all, which is consistent with its superior time complexity $(\mathcal{O}(n \log n + mn))$. Our other algorithms (Heuristic and PSTA) are also much more time-efficient than SJA algorithm, though they have notably larger running time compared to the rest of the algorithms. Nonetheless, it should be noted that we have assumed a pre-scaling will have to be made in case budgets and rewards are not defined as integers. Therefore, we used relatively large budget (up to $B_{max} = 1000$) and hence reward values in all experiments. If pre-scaling can be avoided (or the task requesters in the system have a rather limited budget), the running time of Heuristic and PSTA algorithms will decrease linearly with reduced budget values as shown in Fig. 11c. Finally, we observe that increasing the number of tasks after when there are equal number of workers and tasks in the system ($n = m = 100$) either reduces the running time or the increase in the running time of all algorithms. This is because when there are more tasks than workers, workers will be able to find their stable partners in a shorter time as they are more likely to be matched with a task that is at the top of their preference lists.

6 Conclusion

In this paper, we study the stable task assignment problem in MCS systems. Different from the classic stable matching problem, the task requesters in an MCS system may recruit multiple workers within their budget ranges to reinforce the quality of the sensed data. This makes the generic stability definitions obsolete and the existing approaches to find stable matchings inapplicable in MCS systems. To address this problem, we first define the stability conditions peculiar to MCS systems, and provide the existence and hardness results for stable task assignments in different types of MCS systems. Then, we introduce three different stable task assignment algorithms, namely UTA, PSTA, and Heuristic. We prove that UTA and PSTA algorithms always produce pairwise stable task assignments in uniform and proportional MCS systems, respectively. Finally, we evaluate the performance of the proposed algorithms in terms of user happiness through extensive simulations. The results show that our algorithms significantly outperform the state-of-the-art stable task assignment algorithms in most scenarios. Specifically, PSTA and Heuristic algorithms usually achieve the highest outward and overall user happiness, respectively.

In our future work, we will address the problem of many-to-one stable task assignment with non-additive task preferences where the total QoS that two workers provide for a task may be less or more than the sum of their individual QoS considering the overlapping, redundant skills of workers and the benefit of cooperation, respectively.

Acknowledgments

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

References

[1] F. Khan, A. U. Rehman, J. Zheng, M. A. Jan, and M. Alam, "Mobile crowdsensing: A survey on privacy-preservation, task management, assignment models, and incentives mechanisms," *Future Generation Computer Systems*, vol. 100, pp. 456–472, 2019.

[2] B. Guo, Y. Liu, W. Wu, Z. Yu, and Q. Han, "Activecrowd: A framework for optimized multitask allocation in mobile crowdsensing systems," *IEEE Trans. Human-Machine Systems*, vol. 47, no. 3, pp. 392–403, 2017. [Online]. Available: https://doi.org/10.1109/THMS.2016.2599489

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

[4] J. Wang, J. Tang, G. Xue, and D. Yang, "Towards energy-efficient task scheduling on smartphones in mobile crowd sensing systems," *Computer Networks*, vol. 115, pp. 100–109, 2017.

[5] T. Hu, M. Xiao, C. Hu, G. Gao, and B. Wang, "A qos-sensitive task assignment algorithm for mobile crowdsensing," *Pervasive and Mobile Computing*, vol. 41, pp. 333–342, 2017.

[6] J. Wang, L. Wang, Y. Wang, D. Zhang, and L. Kong, "Task allocation in mobile crowd sensing: State-of-the-art and future opportunities," *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 3747–3757, 2018.

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

[8] V. T. Ho, S.-S. Wong, and C. H. Lee, "A tale of passion: Linking job passion and cognitive engagement to employee work performance," *Jour. of Management Studies*, vol. 48, no. 1, pp. 26–47, 2011.

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

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

[11] M. Abououf, R. Mizouni, S. Singh, H. Otrok, and A. Ouali, "Multi-worker multi-task selection framework in mobile crowd sourcing," *Journal of Network and Computer Applications*, vol. 130, pp. 52–62, 2019.

[12] T. Liu, Y. Zhu, and L. Huang, "Tgba: A two-phase group buying based auction mechanism for recruiting workers in mobile crowd sensing," *Computer Networks*, vol. 149, pp. 56–75, 2019.

[13] L. Wang, Z. Yu, D. Zhang, B. Guo, and C. H. Liu, "Heterogeneous multi-task assignment in mobile crowdsensing using spatiotemporal correlation," *IEEE Transactions on Mobile Computing*, vol. 18, no. 1, pp. 84–97, 2018.

[14] Y. Yang, W. Liu, E. Wang, and J. Wu, "A prediction-based user selection framework for heterogeneous mobile crowdsensing," *IEEE Transactions on Mobile Computing*, 2018.

[15] M. Xiao, J. Wu, S. Zhang, and J. Yu, "Secret-sharing-based secure user recruitment protocol for mobile crowdsensing," in *IEEE INFOCOM*, 2017, pp. 1–9.

[16] Z. Wang, J. Hu, R. Lv, J. Wei, Q. Wang, D. Yang, and H. Qi, "Personalized privacy-preserving task allocation for mobile crowdsensing," *IEEE Transactions on Mobile Computing*, vol. 18, no. 6, pp. 1330–1341, 2018.

[17] J. Li, Z. Cai, J. Wang, M. Han, and Y. Li, "Truthful incentive mechanisms for geographical position conflicting mobile crowdsensing systems," *IEEE Transactions on Computational Social Systems*, vol. 5, no. 2, pp. 324–334, 2018.

[18] 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. Vehicular Technology*, vol. 65, no. 9, pp. 7698–7707, 2016.

[19] Z. Zheng, F. Wu, X. Gao, H. Zhu, S. Tang, and G. Chen, "A budget feasible incentive mechanism for weighted coverage maximization in mobile crowdsensing," *IEEE Transactions on Mobile Computing*, vol. 16, no. 9, pp. 2392–2407, 2016.

[20] Y. Wen, J. Shi, Q. Zhang, X. Tian, Z. Huang, H. Yu, Y. Cheng, and X. Shen, "Quality-driven auction-based incentive mechanism for mobile crowd sensing," *IEEE Trans. Vehicular Technology*, vol. 64, no. 9, pp. 4203–4214, 2015.

[21] H. Wang, S. Guo, J. Cao, and M. Guo, "Melody: A long-term dynamic quality-aware incentive mechanism for crowdsourcing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 4, pp. 901–914, 2018.

[22] H. Gao, C. H. Liu, J. Tang, D. Yang, P. Hui, and W. Wang, "Online quality-aware incentive mechanism for mobile crowd sensing with extra bonus," *IEEE Trans. Mob. Comput.*, vol. 18, no. 11, pp. 2589–2603, 2019.

[23] 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.

[24] Y. Chen, Y. Xiong, Q. Wang, X. Yin, and B. Li, "Ensuring minimum spectrum requirement in matching-based spectrum allocation," *IEEE Transactions on Mobile Computing*, vol. 17, no. 9, pp. 2028–2040, 2018.

[25] Y. Zhong, L. Gao, T. Wang, S. Gong, B. Zou, and D. Yu, "Achieving stable and optimal passenger-driver matching in ride-sharing system," in *2018 IEEE 15th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*. IEEE, 2018, pp. 125–133.

[26] "National resident matching program," 2018. [Online]. Available: http://www.nrmp.org

[27] R. Zhang, X. Cheng, and L. Yang, "Flexible energy management protocol for cooperative ev-to-ev charging," *IEEE Transactions on Intelligent Transportation Systems*, 2018.

[28] Robert W. Irving and Paul Leather and Dan Gusfield, "An efficient algorithm for the "optimal" stable marriage," *J. ACM*, vol. 34, no. 3, pp. 532–543, 1987.

[29] A. Ismaili, N. Hamada, Y. Zhang, T. Suzuki, and M. Yokoo, "Weighted matching markets with budget constraints," *J. Artif. Intell. Res.*, vol. 65, pp. 393–421, 2019.

[30] Y. Kawase and A. Iwasaki, "Approximately stable matchings with budget constraints," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018, pp. 1113–1120.

[31] A. Eryilmaz and R. Srikant, "Fair resource allocation in wireless networks using queue-length-based scheduling and congestion control," *IEEE/ACM Trans. Netw.*, vol. 15, no. 6, pp. 1333–1344, 2007.

[32] F. Zabini, A. Bazzi, and B. M. Masini, "Throughput versus fairness tradeoff analysis," in *Proceedings of IEEE International Conference on Communications, ICC 2013, Budapest, Hungary, June 9-13, 2013*, 2013, pp. 5131–5136.

[33] T. Lan, D. T. H. Kao, M. Chiang, and A. Sabharwal, "An axiomatic theory of fairness in network resource allocation," in *INFOCOM 2010. 29th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 15-19 March 2010, San Diego, CA, USA*, 2010, pp. 1343–1351.

[34] H. Ahmed, K. P. Jagannathan, and S. Bhashyam, "Queue-aware optimal resource allocation for the LTE downlink with best M subband feedback," *IEEE Trans. Wireless Communications*, vol. 14, no. 9, pp. 4923–4933, 2015.

[35] J. Liu, A. Proutière, Y. Yi, M. Chiang, and H. V. Poor, "Stability, fairness, and performance: a flow-level study on nonconvex and time-varying rate regions," *IEEE Trans. Inf. Theory*, vol. 55, no. 8, pp. 3437–3456, 2009.

[36] Wikipedia contributors, "Knapsack problem — Wikipedia, the free encyclopedia," https://en.wikipedia.org/w/index.php?title=Knapsack_problem&oldid=942139230, 2020, [Online; accessed 3-March-2020].

[37] Waze, 2018. [Online]. Available: https://www.waze.com/

[38] X. Yin, Y. Chen, and B. Li, "Task assignment with guaranteed quality for crowdsourcing platforms," in *Quality of Service (IWQoS), IEEE/ACM 25th International Symposium on*, 2017, pp. 1–10.

[39] E. Wang, Y. Yang, J. Wu, W. Liu, and X. Wang, "An efficient prediction-based user recruitment for mobile crowdsensing," *IEEE Transactions on Mobile Computing*, vol. 17, no. 1, pp. 16–28, 2017.

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

[41] K. Iwama and S. Taketomi, "Removable online knapsack problems," in *International Colloquium on Automata, Languages, and Programming*. Springer, 2002, pp. 293–305.

Fatih Yucel (M'17) 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 joined MoWiNG lab in Fall 2017. He is working on development of stable task assignment algorithms for mobile crowd sensing applications. He is a member of IEEE.

Murat Yuksel (SM'11) received the BS degree in computer engineering from Ege University, Izmir, Turkey, in 1996, and the MS and PhD degrees in computer science from RPI, in 1999 and 2002, respectively. He is an associate professor with the ECE Department, University of Central Florida (UCF), Orlando, Florida. Prior to UCF, he was with the CSE Department, University of Nevada, Reno (UNR), Reno, Nevada, as a faculty member until 2016. He worked as a software engineer with Pepperdata, Sunnyvale, California, and a visiting researcher with AT&T Labs and the Los Alamos National Lab. His research interests include networked, wireless, and computer systems with a recent focus on big-data networking, UAV networks, optical wireless, public safety communications, device-to-device protocols, economics of cyber-security and cybersharing, routing economics, network management, and network architectures. He has been on the editorial board of Computer Networks, and published more than 100 papers in peer-reviewed journals and conferences. He is a senior member of the IEEE, and a senior and life member of the ACM..

Eyuphan Bulut (M'08) 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 Assistant 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 and he is a member of IEEE and ACM.