A Crowd-Sensing Framework for Allocation of Time-Constrained and Location-Based Tasks

Rebeca Estrada, Rabeb Mizouni, Hadi Otrok, *Senior Member, IEEE*, Anis Ouali, and Jamal Bentahar, *Member, IEEE*

Abstract—Thanks to the capabilities of the built-in sensors of smart devices, mobile crowd-sensing (MCS) has become a promising technique for massive data collection. In this paradigm, the service provider recruits workers (i.e., common people with smart devices) to perform sensing tasks requested by the consumers. To efficiently handle workers' recruitment and task allocation, several factors have to be considered such as the quality of the sensed data that the workers can deliver and the different tasks locations. This allocation becomes even more challenging when the MCS tries to efficiently allocate multiple tasks under limited budget, time constraints, and the uncertainty that selected workers will not be able to perform the tasks. In this paper, we propose a service computing framework for time constrained-task allocation in location based crowd-sensing systems. This framework relies on (1) a *recruitment algorithm* that implements a multi-objective task allocation algorithm based on Particle Swarm Optimization, (2) *queuing schemes* to handle efficiently the incoming sensing tasks in the server side and at the end-user side, (3) a *task delegation mechanism* to avoid delaying or declining the sensing requests due to unforeseen user context, and (4) a *reputation management component* to manage the reputation of users based on their sensing activities and task delegation. The platform goal is to efficiently determine the most appropriate set of workers to assign to each incoming task so that high quality results are returned within the requested response time. Simulations are conducted using real datasets from Foursquare[1] and Enron email social network.[2] Simulation results show that the proposed framework maximizes the aggregated quality of information, reduces the budget and response time to perform a task and increases the average recommenders' reputation and their payment.

Index Terms—Mobile crowd sensing, worker selection, particle swarm optimization (PSO)

1 Introduction

Mobile crowd-sensing (MCS) is a new paradigm in which a crowd of ordinary citizens utilize their mobile phone or smart devices to conduct complex and large-scale sensing tasks [1]. The user mobility makes MCS a versatile platform that can replace or complement current static sensing infrastructures. MCS systems benefit several applications in various areas such as community dynamics monitoring (i.e., traffic planning [2], environment monitoring [3], or public safety [4]).

The main components of an MCS system are: task manager, customers, and workers. The "workers" are enlisted to perform tasks in return for some compensation or incentive (e.g., entertainment, service, and money) [5]. The customers are the sensing task initiators. Each sensing task has its own requirements (e.g., deadline and budget), and is published on the platform to recruit mobile users to perform it. The task manager is the MCS platform that usually allocates the sensing tasks to appropriate workers.

MCS systems rely on user-contributed or crowd-source information. In other words, a task may be answered by one or multiple workers, depending on the application domain and the task requirements. Some platforms require a single user to perform a task while in others, such as Gigwalk,[3] many users are required to answer the task request to ensure the reliability of the collected information. In the particular case of location-based and time-sensitive sensing tasks, such as checking the on-shelf availability of a product in a convenience store, the users can collect the data at the precise time and location [6]. With this information, any company can reduce the cost of taking inventories, while maintaining the proper stock levels at different stores. Currently, several well-known brands and retailers are customers of Gigwalk. This suggests that the collection of location-based and time-sensitive data using MCS is a practice of growing importance.

1. https://archive.org/details/201309_foursquare_dataset_umn
2. https://snap.stanford.edu/data/email-Enron.html
3. http://www.gigwalk.com/

- *R. Estrada is with the Escuela Superior Politécnica del Litoral, ESPOL, FIEC & ReDIT Research Group, Guayaquil, Ecuador and with the Concordia Institute for Information Systems Engineering (CIISE), Concordia University, Montréal, QC, Canada. E-mail: restrada@espol.edu.ec.*
- *R. Mizouni is with the Department of ECE, Khalifa University, Abu Dhabi 127788, UAE. E-mail: rabeb.mizouni@kustar.ac.ae.*
- *H. Otrok is with the Concordia Institute for Information Systems Engineering (CIISE), Concordia University, Montréal, QC H3G 1M8, Canada, and with the Department of ECE, Khalifa University, Abu Dhabi 127788, UAE. E-mail: Hadi.Otrok@kustar.ac.ae.*
- *A. Ouali is with the Etisalat British Telecom Innovation Center (EBTIC), Khalifa University, Abu Dhabi 127788, UAE. E-mail: anis.ouali@kustar.ac.ae.*
- *J. Bentahar is with the Concordia Institute for Information Systems Engineering (CIISE), Concordia University, Montréal, QC, Canada. E-mail: bentahar@ciise.concordia.ca.*

Manuscript received 30 May 2016; revised 13 May 2017; accepted 27 June 2017. Date of publication 11 July 2017; date of current version 13 Oct. 2020. (Corresponding author: Rebeca Leonor Estrada.)
Digital Object Identifier no. 10.1109/TSC.2017.2725835

In MCS systems, selection of participants is one of the main challenges, which has an impact on the quality of the task outcome. Several approaches have been proposed to tackle this problem. They aim at selecting the best set of users to complete a task subject to several constraints (e.g., budget). However, most of these approaches are single-task oriented and do not consider the impact of the task allocation problem in a large-scale scenario. There are few solutions that address the multi-task allocation problem (e.g., TaskMe [7] and ActiveCrowd [8]). In these approaches, the relationship between the number of active participants and the number of tasks to be completed affects severely the task allocation and completion rates. Moreover, the multi-task allocation problem faces other challenges such as location dependency, diversity of quality of the sensing data, and budget constraints.

The limitations of the existing approaches are summarized as follows:

- The selection of participants is based on a single-objective optimization problem (e.g., maximizing the number of accomplished tasks, minimizing the budget [9]) assuming that the workers are willing to participate regardless how much they expect to earn. Moreover, modeling the sensing location-based task without specifying any time constraints [10], [11] is unrealistic as assumption for many crowd-sensing systems.
- Task allocation and completion rates are severely affected by limited resources (i.e., active workers) [7], [8]. Therefore, complementary components should be investigated in order to enhance the performance of the task allocation model.
- There is a lack of effective delegation mechanisms. A delegation scheme [12] with monetary incentives can be abused because the participants can misbehave. In other words, workers may compete to perform a task and once they are selected (for instance due to their high reputation), they may delegate their work to other workers and still get paid for the task (that other workers performed).
- The inability to complete tasks affects not only the participants' payment but also their reputation. For example, the approach in [13] does not pay the workers if they give wrong answers, which can discourage the workers to stay committed to the MCS system. The model should update the worker's reputation based on their performance and their successful delegation.

In this paper, our solution addresses the trade-off among quality of the sensed data, budget and time constraints for tasks that require the sensed data within a time frame because the information is useless afterwards. The main contribution of this paper is a service computing framework for the allocation management of time-constrained and location-based sensing tasks that consists of:

- a multi-objective task allocation algorithm to deal with worker selection taking into account that the workers establish their minimum wages to perform any task. This algorithm maximizes the aggregated QoI (Quality of Information)/budget ratio while minimizing the response time under scenarios with time and budget limitation, which is implemented using the Particle Swarm Optimization (PSO) technique.
- two types of queuing schemes: (1) a First In First Out (FIFO) queue implemented in the device application that allows participants to be selected to perform consecutive tasks; and (2) a priority queue in the task manager to queue arriving tasks when there is no available resources and their required response time has not expired.
- a delegation mechanism in case the workers cannot finish their allocated task. Workers may recommend a set of workers from their social network to finish their assigned task. To avoid any abuse of the system, this mechanism affects the recommender reputation based on the performance of the delegated workers.
- a systematic evaluation of workers' reputation based on their performance and the incentives/penalties from the delegation mechanism.

To evaluate the MCS system, we use event-driven simulations [14]. In other words, the functioning of the system is simulated as a discrete sequence of events over time where the occurrence of an event triggers the performance of some actions. Examples of these events include task arrival/departure or mobile user arrival/departure. In this paper, we only consider task arrival/departure events leaving the modeling of mobile user arrival/departure events for future work.

For comparison purposes, we use two benchmark models that are based on the proposed framework using different task allocation algorithms found in the literature. The first algorithm aims at maximizing the quality of information per task under budget constraints [15]. It did not consider time-constrained tasks. We implemented it using the PSO technique. The second one is the heuristic algorithm presented in [16]. We modified both algorithms to include the time constraint and budget estimation as the product of the traveled distance and worker payment. Both benchmark models are evaluated using the other components of the proposed solution, namely, the queueing schemes, delegation mechanism and reputation management. A performance comparison is carried out under two scenarios (1) an incremental scenario where the number of tasks is increased by a step of 15 tasks, which allows us to show that the proposed multi-objective task allocation algorithm can enhance the performance of the benchmark models for different number of tasks in a specific time slot; and (2) a realistic scenario, where the task arrival is modeled as a Poisson process and the required response time per task is an exponential random variable. Simulations are conducted using locations from a real dataset[4] where a large-scale scenario is generated.

The remaining of the paper is organized as follows: Section 2 presents an overview of the relevant related work. Section 3 formulates the multi-task allocation problem and discusses the challenges of such a formulation. Section 4 presents our proposed service computing framework and its components. Section 5 describes the two benchmark models used in this paper. Section 6 presents the simulation scenarios and results. Finally, Section 7 concludes the paper.

2 Related Work

Crowdsensing techniques and challenges are analyzed with a focus on resource constraints and data quality issues in

4. https://archive.org/details/201309_foursquare_dataset_umn

TABLE 1
Location-Based Task Management Solutions

Solution	Contribution	Dataset	Limitations
ProMoT [11]	Auction mechanism that maximizes the profit of the platform while providing satisfying rewards to the workers	Randomly Generated	- No time-constraints - Single-task oriented - Does not take into account workers' reputation
QOATA [15]	Single objective optimization approach to maximize the task QoI with budget constraint taking into account the workers reputation	Randomly Generated	- No time-constraints - Single-task oriented - Does not take into account workers' reputation
Budget Task [16]	Heuristic algorithm for single objective optimization problem that maximizes the task QoI taking into account the workers reputation and payment	Workers and tasks from Foursquare [22] Worker's Reputation is randomly generated	- No time-constraints - Single-task oriented - Payment does not depend on traveled distance
TaskMe [7]	Two bi-objective optimization approaches for participant selection FPMT: to maximize the total number of accomplished tasks and also to minimize the total movement distance. MPFT: to minimize total incentive payments for participants and minimize traveling distance to complete tasks	For workers D4D[23]	- Complexity - Fairness among workers - Does not guarantee the task QoI - Does not take into account workers' reputation - Not suitable for large-scale scenario - Task location are randomly generated within the mobile user area
ActiveCrowd [8]	Two greedy-enhanced genetic algorithms for optimal task allocation to minimize the total distance traveled to complete the tasks under two common situations: 1) intentional-movement-based selection for time-sensitive tasks and 2) unintentional-movement-based selection for delay-tolerant tasks	For workers and tasks D4D [23]	- Not an optimal solution - Does not guarantee the task QoI - Worker Payment is not considered - Complexity - Task/worker locations given by cell towers' location

[17]. A better understanding of resource management and QoS estimation in mobile crowdsensing can help researchers design cost-effective crowdsensing systems that can reduce the cost by fully utilizing the resource and improve the QoI for customers. Regarding task allocation and participant selection problems, the majority of existing solutions are single-task oriented. These approaches do not address the task allocation problem for a large-scale scenario where multiple heterogeneous tasks can be requested by several customers and be performed by several workers. Moreover, the participants selection procedure is based on a single optimization objective (e.g., sensing costs [9], coverage of targets of interest [18], quality or credibility of sensed data [15], [16], or revenue [10], [11]). There are few solutions that address the multi-task allocation problem taking into account several optimization objectives to find a trade-off between the most commonly used factors [7], [8].

Some researchers introduce redundancy to ensure a certain level of reliability in MCS systems [19] and several workers are asked to carry out the same task. Then, a technique such as majority voting [20] is applied to determine the answer for the requester. Although this solution reduces the impact of wrong answers on the final result [21], it increases the required budget to perform a given task. A trade-off between maximizing the aggregated quality of information per task while minimizing the budget per task should be investigated.

Furthermore, once the workers are selected to perform a task, any worker selected may not complete the task due to unforeseen circumstances. If the quality of information of the task is not met, then, a re-selection of workers should be carried out. Instead of performing the re-selection procedure, a delegation mechanism was proposed in [12]. Their mechanism allows a worker who cannot finish the task to recommend other worker from her/his social network to perform the task. For time-constrained tasks, the delegation is more complicated because the data should be collected within a certain time interval.

Table 1 summarizes the main contributions and limitations of five relevant approaches found in the literature related to our work.

3 Formulation of Multi-Task Allocation Problem

In this section, the model for multi-task allocation problem is presented. We consider a service computing framework where the service provider publishes the tasks for the workers. Then, each worker chooses some of the published tasks and provides the minimum payment that the worker is willing to receive from a specific range. The range payment depends on the worker reputation. The workers can visualize the tasks that satisfy some basic constraints such as the workers is within the task coverage radius and the requested payment is lower than the maximum payment per task.

3.1 Problem Formulation

The objective of the multi-task allocation problem (MTAP) is to maximize the ratio of the aggregated QoI to the required budget and response time to perform several tasks given a set of available workers. The MCS should determine the assigned tasks for each worker within their requested response time and time constraints. Workers have different reputation levels based on their historical performance in the MCS system. They are also attributed a confidence level, which represents the self-confidence that a worker has

related to the task accomplishment. For example, the battery usage level of the worker's mobile device can be used as the confidence level. In fact, a user with low battery level is less likely to be selected to perform any task. For location-based tasks, the user needs to travel a certain distance to perform the task. As the distance increases, the user needs more time to travel to the task location. This fact delays the data collection and increases the cost of performing the task (i.e., the system needs to reward the user for travelling a long distance).

While there is no definition of information quality that fits every scenario, this concept is often related to the accuracy of the information, completeness, and timeliness. Many formulations have been proposed in the literature [15], [16]. In our case, the quality of information of a worker c_j^i reflects the accuracy and timeliness of the collected data and is given by

$$c_j^i = r_j \times \beta_j \times \delta_j^i, \tag{1}$$

where r_j and β_j are the reputation and confidence of the worker to perform a given task during a given period of time. The reputation is a parameter computed by the MCS system based on the historical performance of the worker. The confidence β_j is an input parameter reflecting the worker's self-confidence to perform the task. The battery level is an example of such a parameter that could also be expressed by a combination of different other parameters. δ_j^i is a function of the distance between a worker and a given task. We use the same function given in [16], which calculates the discount to the worker's reputation as a result of his proximity to the task location

$$\delta_j^i = 1 - \max\Big(0, \min\Big[log_{DC}\Big(d_j^i\Big), 1\Big]\Big), \tag{2}$$

where DC is the city radius (i.e., 30 km) and d_j^i represents the euclidean distance between the worker location l_j and the task location l^i, which are given in GPS coordinates in the Foursquare dataset. This distance is estimated using the Haversine formula [24].

The objective function for the MTAP problem aims at maximizing the aggregated quality of information per unit of required budget and it can be formulated as

$$\max_{\mathbf{X},\mathbf{P}} \sum_{i \in T} \left[\frac{\left(\sum_{j \in W} c_j^i X_j^i\right) - C^i}{\left(\sum_{j \in W} d_j^i P_j^i\right) \times \max_{j \in W}(t_j^i)} \right], \tag{3}$$

where $\mathbf{X}$ and $\mathbf{P}$ correspond to the vectors of the variables X_j^i and P_j^i respectively. X_j^i is a binary variable that indicates the selection of a worker j to perform the task i while P_j^i corresponds to the payment per traveled kilometer received by worker j to perform the task i. C^i represents the minimum QoI required by task i. Thus, the numerator represents the aggregated quality of information per task i and the denominator is the product between the required budget and the required time to finish the task i for the selected set of workers given by vector $\mathbf{X}$. W and T are the set of workers and tasks respectively. t_j^i is the estimated time that the worker takes to reach the location of the task τ_i. This time is a calculated as the distance between the worker j and task i divided by the speed of the worker j. The sensing time is assumed to be negligible in comparison to this time.

TABLE 2
Model Parameters

Name	Description
Task Parameters	
T	Set of tasks
B^i	Maximum budget per task i
R^i	Coverage radius of task i
C^i	Minimum QoI for task i
P_{max}^i	Maximum payment allowed per traveled km for workers for task i
$P_{g,max}^i$	Maximum payment per traveled km for worker with reputation level g for task i
$P_{g,min}^i$	Minimum payment per traveled km for worker with reputation level g
DC	Radius of the city
l^i	Location of the task i
N_{max}^i	Maximum number of workers per task i
t_{max}^i	Maximum response time required by Task i
t_q^i	Queuing time for task i
W^i	Set of selected workers to perform task i
S^i	Subset of arbitrary workers in W^i that can perform task i
Worker Parameters	
r_j	reputation of the worker j
$r_j^{(k)}$	reputation of the worker j at the instant k
N_j^{max}	Maximum number of consecutive tasks per worker
W	Set of workers
c_j^i	QoI provided by the worker j to the task i
β_j	Self-Confidence to perform any task of worker j
l_j	Location of worker j
P_j^{min}	Requested payment that the worker is willing to receive per traveled km
d_j^i	Distance from task i to worker j
t_j^i	Time that the worker takes j to reach the location of task i
W_j^{SN}	Set of recommended workers from the social network of worker j
$W_j^{D,i}$	Set of delegated workers to perform the task i that worker j could not do it
N_j^T	Number of assigned task to the worker j at the time k
N_j^{TV}	Number of completed tasks with true value for worker j
N_j^{CT}	Number of completed tasks for worker j
N_j^{IT}	Number of incomplete tasks for worker j
General Parameters	
r_g^{min}	Minimum reputation for level g
r_g^{max}	Maximum reputation for level g
Output Variables	
X_j^i	Binary variable that indicates if task i is allocated to worker j
P_j^i	Value paid per traveled km to the worker j for performing the task i

3.1.1 Model Parameters

For the sake of clarity, Table 2 summarizes the notation used in this paper.

3.1.2 Model Constraints

The objective function (3) is subject to the following constraints

$$\sum_{j \in W} X_j^i \leq N_{max}^i, \qquad i \in T \tag{4}$$

$$\sum_{j \in W} c_j^i X_j^i \geq C^i, \qquad i \in T \tag{5}$$

$$t_j^i \times X_j^i \leq t_{max}^i, \qquad ; i \in T, j \in W \tag{6}$$

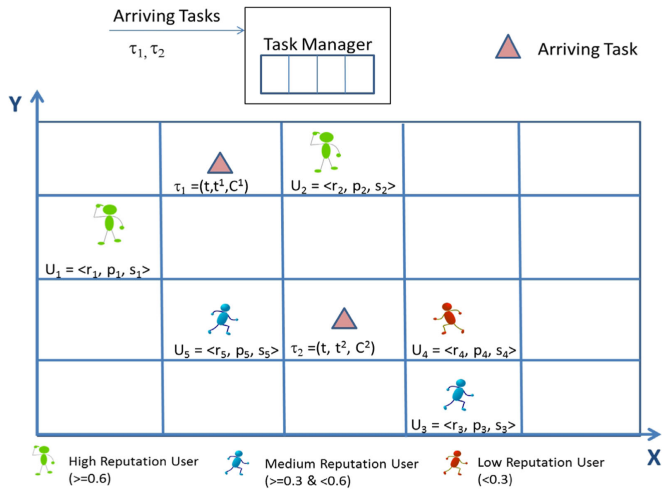


Fig. 1. Scenario with two tasks and five mobile users.

$$P_j^i \leq X_j^i P_{g,max}^i, \qquad ;i \in T, j \in W \tag{7}$$

$$P_j^i \geq X_j^i \max(P_j^{min}, P_{g,min}^i) \qquad ;i \in T, j \in W. \tag{8}$$

Constraint (4) defines the maximum number of workers that can be allocated to perform a task i. Constraints (5) and (6) ensure that the set of selected workers will satisfy the quality of information required by task i and deliver the sensed data within the required response time. Finally, constraints (7) and (8) determine the upper and lower bound for the payment of the workers with reputation g.

This model aims at maximizing the objective function (3) and is a non-linear mixed integer problem (MINLP). It could be solved by decomposing the complex problem into several subproblems for each task $i \in T$ [15] or several subproblems for each worker $j \in W$ [25].

3.2 NP-Hardness

Theorem 1. *The MTAP as shown in Eqs. (3), (4), (5), (6), (7), (8) is NP-hard.*

Proof. We prove this theorem by showing that an arbitrary instance of a NP-complete problem can be polynomially Turing reduced to an instance of the MTAP′, a simplified version of the MTAP where each worker has to be assigned to only one task. Thus, the idea is to provide an algorithm that solves the NP-complete problem in polynomial time by calling an oracle that solves the MTAP′. The candidate NP-complete problem we consider is the two-way Number Partitioning Problem (2-NPP) [26].

Given a multiset S of natural numbers, the 2-NPP consists of partitioning S into two subsets S_1 and S_2, so that the sum of the numbers in S_1 is nearly equal to the sum of the numbers in S_2. This problem can be simply solved by calling an oracle to the MTAP′ where two sensing tasks T_1 and T_2 localized in the same region l_1 have to be assigned to a set of workers sharing the same location l_2 so that d_j^i are the same for all the pairs of workers j and sensing tasks i. The total number of workers is equal to $|S|$. All the workers are paid the same price per traveled kilometer so that P_j^i are equal for all the pairs i and j. Moreover, the workers have the same speed s_j of movement so that t_j^i are also equal for all the pairs. Each number $s \in S$ is associated to a worker j who is characterized by two indistinguishable qualities c_j^1 and c_j^2 (i.e., $s = c_j^1 = c_j^2$). These numbers are linked to C^1 and C^2 as follows: $\sum_{s \in S} s = \sum_{j \in W} c_j^1 = \sum_{j \in W} c_j^2 = C^1 + C^2$.

If $\sum_{j \in W} c_j^1$ is even, then set $C^1 = C^2$, otherwise, set $C^1 = C^2 + 1$. N_{max}^1 and N_{max}^2 are set to be large enough so that constraint (4) is always satisfied. If the oracle call provides a solution, then two subsets of workers are identified so that the sum of qualities c_j^i is getting maximized, and according to constraint (5), the sum of these qualities in the two sets are nearly equal. This provides a solution to the 2-NPP since the numbers in the two sets S_1 and S_2 are mapped to c_j^i. If the call to the oracle does not provide a solution, then the oracle is called again after updating C^1 and C^2 as follows: $C^1 := C^1 + 1$ and $C^2 := C^2 - 1$. This procedure is repeated until a solution is provided. The procedure is guaranteed to terminate since the only reason of not providing a solution is the non-satisfaction of constraint (5), and each iteration is modifying C^1 and C^2 towards the satisfaction of the inequality.

The proposed procedure runs in $\mathcal{O}(\sum_{j \in W} c_j^1)$ as there are at most $C^1 + C^2$ calls to the oracle. Therefore, the reduction is polynomial since constructing the 2-NPP solution is a simple mapping of the quantities c_j^i of the workers conducting task t_i to the subset S_i. Thus, the NP-hardness of MTAP′ follows from the fact that 2-NPP $\preceq_p$ MTAP′, where $\preceq_p$ is the polynomial Turing reduction. Since the MTAP is harder than the MTAP′ because in the MTAP a worker can be assigned to 0 or many tasks, which increases the number of possible combinations, we conclude that the MTAP is NP-hard. □

TABLE 3
Worker Parameters and Metrics

Worker	r_j	P_j^{min}	s_j	d_j^1	d_j^2	t_j^1	t_j^2	c_j^1	c_j^2
U_1	0.85	5	1	1.41	2.2	1.41	2.2	0.76	0.65
U_2	0.8	4.2	0.75	1	2	1.33	2.67	0.8	0.64
U_3	0.5	2.5	0.25	3.61	1.41	14.44	5.64	0.31	0.45
U_4	0.25	0.5	1	2.82	1	2.82	1	0.17	0.25
U_5	0.55	2.7	0.5	2	1	4	2	0.44	0.55

3.3 Motivating Example

Let's consider a simple MCS system with five available workers $(U_1, .., U_5)$ and two arriving tasks (τ_1, τ_2) in a two dimensional (2D) area as illustrated in Fig. 1. This simplified example is only for illustrative purpose; real scenarios are more complex and highly scalable involving considerable number of tasks and workers. Workers might have different reputations (e.g., high, medium and low reputation users). The question is how to allocate these workers to each task so that they can maximize the aggregated quality of information per unit of used budget and time to execute each task. The required quality of information are 1 and 0.9 for tasks τ_1, and τ_2 respectively while response time is 5 minutes for both tasks.

Table 3 presents the worker parameters, namely reputation, speed, minimum payment to receive as well as the

TABLE 4
Set of Workers to Perform Task τ_1

Workers	QoI	Agg QoI	Budget	Response Time	$\frac{AggQoI}{B\times T}$
$[U_1, U_2]$	1.56	0.56	11.25	1.41	0.04
$[U_1, U_2, U_4]$	1.74	0.74	12.66	2.82	0.02
$[U_1, U_2, U_4, U_5]$	2.18	1.18	18.06	4.00	0.02
$[U_1, U_2, U_5]$	2.00	1	16.65	4.00	0.02
$[U_2, U_4, U_5]$	1.41	0.41	11.01	4.00	0.01
$[U_1, U_4, U_5]$	1.38	0.38	13.86	4.00	0.01
$[U_2, U_5]$	1.24	0.24	9.60	4.00	0.01

estimated distance to reach the task locations and the estimated quality of information that each worker can contribute to each task using Eq. (1). We include a minimum payment per worker to represent the willingness of the worker to perform a task. This minimum payment depends on the corresponding range of the worker reputation $\langle P^i_{g,min}, P^i_{g,max}\rangle$. We assume that the range payment for each reputation level are the same for both tasks. For instance, high reputation users can select values in the range between 3.5 and 5 while medium and low reputation users can select values from [2, 3.5) and [0.5, 2) respectively. For the motivating example, we assume that the worker confidence is equal to 1 for any task.

Table 4 presents the worker combinations that meet the requirements for task τ_1. We start with task τ_1 because it requires higher QoI. Then, the problem is to select disjoint set of mobile users to solve each task since they cannot perform both tasks at the same time.

Selecting the first option for task τ_1 could mean that only three workers can be allocated to task τ_2. Then, only one combination of these workers meet the QoI of task τ_2, which is shown in the scenario I in Table 5. If the model only maximizes the QoI per task under budget constraint, then, this combination could be selected as a solution. However, this implies that the worker with higher response time do not get paid and the worker satisfaction is decreased.

In the second scenario of Table 5, the task manager allows each user to keep a local queue of tasks to be carried out sequentially. Thus, the users U_1 and U_2 that were allocated to perform task τ_1, can also perform task τ_2 after finishing task τ_1. This scenario presents the worker combinations including users U_1 and U_2 taking into consideration the total time that these users need to reach the location of task τ_2 while their payments are estimated using the traveled distance from the location of task τ_1 to the location of task τ_2. In this case, the first option is the one that maximizes the ratio between aggregated QoI divided by the product of budget and response time and also guarantee the worker payment. This means that workers U_1 and U_2 will perform two consecutive tasks while workers U_4 and U_5 will carry out one task. In addition, all the workers performing tasks will receive their payment owing to the fact that they can deliver the sensed information within the requested time. Worker U_3 is not selected because the sensed data cannot be delivered to the task manager within the required response time. This allows the MCS system to meet the requirements of task τ_2 and to enhance the worker satisfaction.

In summary, the MTAP model proposed assigns all tasks to the available workers in one step. We demonstrated by means of an example that this might not be the best strategy and it would be better if some workers are allowed to execute several tasks sequentially. The following section presents an enhanced task manager framework that deals with the single task allocation at a time and allows the workers to be selected to perform more than one task. The workers are considered for the allocation of new tasks as long as they can reach the new task location within its required time and they do not reach their maximum number of tasks allowed by the system.

TABLE 5
Set of Workers to Perform Task τ_2 under Two Scenarios

Scenario I: U_1 and U_2 are not included					
Workers	QoI	Agg QoI	Budget	Response Time	$\frac{AggQoI}{B\times T}$
$[U_3, U_4, U_5]$	0.92	0.02	6.73	5.64	0.0006
Scenario II: U_1 and U_2 are included					
Workers	QoI	Agg QoI	Budget	Response Time	$\frac{AggQoI}{B\times T}$
$[U_1, U_2, U_4, U_5]$	2.06	1.16	23.44	4.26	0.012
$[U_1, U_4, U_5]$	1.45	0.55	14.2	3.61	0.011
$[U_2, U_4, U_5]$	1.41	0.51	12.44	4.26	0.009
$[U_1, U_2, U_4]$	1.51	0.61	20.74	4.26	0.006
$[U_1, U_5]$	1.2	0.3	13.7	3.6	0.006
$[U_2, U_5]$	1.16	0.26	11.94	4.26	0.005

4 Service Computing Framework for Task Management in MCS System

In this section, we present the proposed framework for task management in the MCS system. The main idea is that the MCS task manager can be considered as a queuing system with N servers (mobile users), which can process the same task in parallel and at the same time several tasks can be allocated for service if there are some available servers. Thus, our MCS system can be seen as discrete-event system where the system state $S(t)$ is defined by the number of tasks in service and the number of queued tasks $(T_s(t), T_q(t))$. The task arrival process is a Poisson process while the service time per server is deterministic.

4.1 Multi-Objective Task Allocation Model

Here, we present the optimization problem for the worker selection per task that maximizes the ratio between the aggregated quality of information, the product of the budget,and the execution time under the constraints to meet the quality of information with the limited budget and response time. This means that our objective function for every task i is

$$\max_{\mathbf{X},\mathbf{P}} \frac{\left(\sum_{j\in W} c^i_j X^i_j\right) - C^i}{\left(\sum_{j\in W} d^i_j P^i_j\right) \times \max_{j\in W}(t^i_j)}. \tag{9}$$

The denominator in (9) corresponds to the task budget multiplied by the response time to gather the information from the allocated workers to the task (i.e., the maximum time of the allocated workers to perform a task). The budget is estimated as the payment paid per traveled kilometer to the worker, P^i_j, multiplied by the traveled distance per worker, d^i_j. The second term in the denominator is introduced to reduce the total required time to collect the sensed information.

In our model, a spatial task i is represented as a tuple of the form $\langle l^i; C^i; R^i; B^i; P^i_{max}; t^i_{max}\rangle$. These parameters corresponds to the task location, minimum expected QoI, coverage radius, budget, maximum payment per traveled kilometer, and maximum response time respectively. Workers are represented by tuples of the form $\langle l_j; r_j; \beta_j; P^{min}_j; s_j\rangle$, which represent their location, reputation, confidence to perform a task, minimum payment they are willing to receive and finally their speed.

In summary, we want to maximize the aggregated QoI/budget ratio while minimizing the time to collect the information about a given task from the workers taking into account the worker's willingness to perform this task. Our objective function is a multi-objective function. Particle Swarm Optimization has been used to solve several complex optimization problems with multi-objective function. Moreover, PSO has been proven to obtain a satisfying solution while speeding up the optimization process in comparison to other evolutionary-based optimization algorithms [27]. Therefore, we propose to solve the optimization problem using this technique.

4.1.1 Model Constraints

The constraints for our task allocation sub-problem are:

- Maximum Budget per task i

$$\sum_{j\in W} P^i_j d^i_j \leq B^i. \tag{10}$$

- Minimum required Quality of Information

$$\sum_{j\in W} c^i_j X^i_j \geq C^i. \tag{11}$$

- Maximum response time

$$t^i_j \times X^i_j \leq (t^i_{max} - t^i_q), \tag{12}$$

and the constraints (7) and (8) given for MTAP model.

4.1.2 PSO-Based Multi-Objective Task Allocation Algorithm (PSO-MOA)

We propose to solve the worker selection for each task defined by Eq. (9) using PSO, which is a population-based search approach and depends on information sharing among the population members to enhance the search processes using a combination of deterministic and probabilistic rules. PSO algorithm uses two vectors that determine the position and velocity of each particle n at each iteration k. These two vectors are updated based on the memory gained by each particle. The position y^k_n and velocity v^k_n of a particle n at each iteration k are updated as follows:

$$y^k_n = y^{k-1}_n + \delta_t v^{k-1}_n, \tag{13}$$

$$v^k_n = \omega v^{k-1}_n + c_1 r_1 (p^{local}_{k-1} - y^{k-1}_n) + c_2 r_2 (p^{global}_{k-1} - y^{k-1}_n), \tag{14}$$

where δ_t is the time step value typically considered as unit [28], p^{local}_{k-1} and p^{global}_{k-1} are the best ever position of particle n and the best global position of the entire swarm so far, and r_1 and r_2 represent random numbers from interval [0,1]. The parameters ω, c_1 and c_2 are the configuration parameters that determine the PSO convergence. The first term is related to the particle inertia ω, which is used to control the exploration abilities of the swarm. Large inertia values produce higher velocity updates allowing the algorithm to explore the search space globally. Conversely, small inertia values force the velocity to concentrate in a local region of the search space. Parameters c_1 and c_2 are known as the cognitive scaling and social scaling factors. Thus, the second and third terms are associated with cognitive knowledge that each "particle" has experienced and the social interactions among "particles" in the population respectively [29].

Many PSO variants update the inertia parameter ω using different functions [29]. For simplicity, we consider the following

$$\omega^n_k = \omega_o \times \frac{(1 - \sqrt{(p^{local}_k - y^k_n)^2 + (p^{global}_k - y^k_n)^2})}{\max(\sqrt{p^{local}_k - y^k_n)^2 + (p^{global}_k - y^k_n)^2})}. \tag{15}$$

According to [28], the convergence of PSO is guaranteed if the following set of stability conditions are met

$$0 \leq (c_1 + c_2) \leq 4 \text{ and } \frac{c_1 + c_2}{2} - 1 \leq \omega \leq 1.$$

For our PSO-based multi-objective task allocation algorithm, the position particle **Y** in the search space is given by two vectors **(X, P)**, which represent the allocation of the task i to worker j and price per worker respectively.

PSO algorithm is formulated as an unconstrained optimizer. One way to accommodate constraints is to augment the objective function with penalties proportional to the degree of constraint infeasibility. In our PSO algorithm, a penalty parameter-less scheme [30] is used to accommodate the constraints, where the penalties are based on the average of the objective function and the level of violation of each constraint during each iteration. According to [28], the penalty coefficients pc_l are determined by

$$pc_l = |\overline{f}(y)| \frac{\overline{g_l}(y)}{\sum_{j=1}^{PC} [\overline{g}(y)]^2}, \tag{16}$$

where l indicates a particular constraint, $\overline{f}(y)$ is the average objective function, $\overline{g}(y)$ is the average level of l_{th} constraint violation over the current population and PC is the number of penalty coefficients, which also corresponds to the total number of constraints [28]. Thus, the fitness function is defined by

$$f'(y) = \begin{cases} f(y^k_n), & \text{if } y^k_n \text{ is feasible} \\ f(y^k_n) + \sum_{l=1}^{PC} pc_l \widehat{g}(y^k_n), & \text{otherwise.} \end{cases} \tag{17}$$

and $\widehat{g}(y^k_n)$ is determined as follows:

$$\widehat{g}(y^k_n) = \max\big(0, g_j(y^k_n)\big). \tag{18}$$

Accordingly, the average of the fitness function for any population is approximately equal to $\overline{f}(y) + |\overline{f}(y)|$. Since we formulate our model as a maximization problem and PSO is defined to solve a minimization problem, we modify our objective function from (9) to

$$f(\mathbf{X}, P) = Z - \left(\frac{\left(\sum_{j \in W} c_j^i X_j^i\right) - C^i}{\left(\sum_{j \in W} d_j^i P_j^i\right) \times \max_j(t_j)} \right), \qquad (19)$$

where Z is a large number, which is computed as the objective function (9) under the worst case scenario with the minimum response time and maximum budget that a task can allow. The resulting values is then multiplied by 100 to ensure Z to be large enough. The fitness function of our minimization problem is given by

$$f'(x) = \begin{cases} f(\mathbf{X}, P), & \text{for feasible solutions} \\ f(\mathbf{X}, P) + \sum_{l=1}^{PC} pc_l \times \widehat{g}(X, P), & \text{otherwise} \end{cases}. \qquad (20)$$

where model constraints are included in $\sum_{l=1}^{PC} pc_l \widehat{g}(X, P)$ to penalize unfeasible solutions. Algorithm 1 presents the PSO-based multi-objective task allocation algorithm. The random function in the Algorithm 1 returns a random number between 0 and 1.

Algorithm 1. PSO-MOA Algorithm

Data: Worker Locations (l_j),
Worker Demands (P_j^{min}),
Task Location (l^i),
Maximum Budget per Task B^i,
Task Maximum Price P_{max}^i
Coverage radius d^i
Required Time t^i)
Result: Set of worker allocated to the task and the price to be paid per worker (X_j^i, P_j^i).
begin
Generate initial swarm with the particle positions $Y_j^i = (X_j^i, P_j^i)$ and velocities randomly v_j^i;
Evaluate Fitness Function;
Determine first global best of the swarm;
while $k \leq MaxIteration$ **do**
Update Position using Eq. (13);
Evaluate Fitness Function;
Determine best local for each particle;
Determine best global in the swarm and update the best global;
Update the inertia parameter w using Eq. (15);
Update velocity using Eq. (14);
end
end

PSO Parameters Settings and Convergence Analysis. The convergence analysis of the proposed PSO algorithm using different values of cognition and social behavior factors (c_1, c_2) is shown in Fig. 2. It can be observed that the best objective value was given for the setting c_1=2 and c_2 = 1.5 after 400 iterations. Therefore, we set the parameters to those values for the rest of our simulations.

4.2 Queuing Schemes

4.2.1 Task Queuing in the Device Application

Our framework proposes to have a local queue in the worker's device with a maximum number of consecutive tasks that can be allocated to him, N_j^{max}. Thus, several tasks can be assigned to a worker and they are going to be performed in the sequential order that they were assigned to the worker (i.e., FIFO queue). The main idea is to reduce the number of rejected tasks in a scenario with reduced number of workers within the task coverage area. Thus, the following constraint is added to our task allocation model

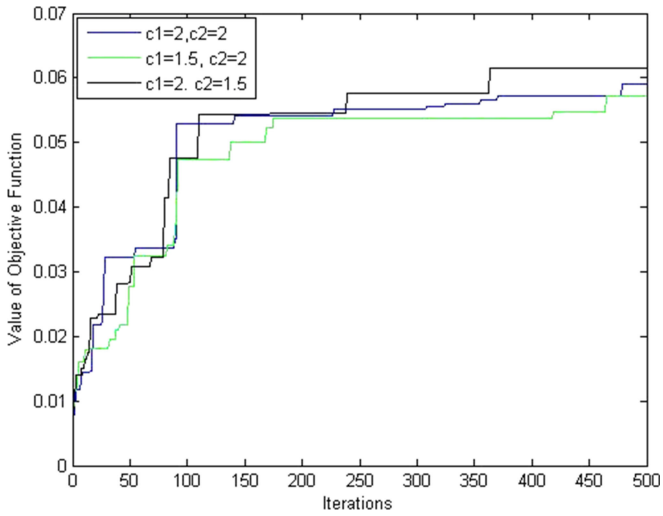


Fig. 2. Convergence Analysis for different settings of c_1, c_2.

$$X_j^i + N_j^T \leq N_j^{max} \qquad ; j \in W, \qquad (21)$$

where N_j^T is the number of assigned task to the worker j at the time k without being processed and N_j^{max} is the size of the worker local queue. For convenience, we assume that all the workers have a local queue with the same size, (i.e., $N_j^{max} = N_w^T; \quad \forall j \in W$).

4.2.2 Priority-Based Task Queuing in Task Manager

Our framework also uses a priority-based queuing system in the task manager. If there are no available workers to perform the task, then, the task is queued until the potential workers are released from their current assigned task (i.e., after finishing their assigned tasks) and can perform the queued task. Otherwise, the task is rejected. The priority is defined by the remaining response time. Thus, a low remaining response time task should be assigned first over the tasks with higher time. Doing so, the model always selects the task that needs to be served first according to the remaining response time. If the remaining response time is zero, then, the task is eliminated from the queue and it is marked as an unsuccessful queued task. The task allocation algorithm complements the queuing scheme since it aims at minimizing the response time per task. This means that the workers are able to finish rapidly their current tasks and they can be assigned to the new arriving or queued tasks.

4.3 Task Delegation Mechanism

The proposed task delegation mechanism is intended to avoid the QoI decrease when some workers are not able to finish their assigned task(s) due to unpredictable circumstances. In our proposal, a worker who cannot finish a task is able to recommend one or a worker set W_j^{SN} from the social network to perform the task. Since the worker's social network may be unknown to the MCS system, the task manager determines the appropriate set W_j^D from the set of

recommended workers W_j^{SN}, who are registered in the system and can satisfy the following requirements:

- The sum of the QoI of selected workers should be at least equal to QoI of the recommender.

$$\sum_{h \in W_j^{D,i}} c_h^i \geq c_j^i, \qquad ; i \in T, \tag{22}$$

where $W_j^{D,i}$ represents the selected delegated workers for task i obtained from the set of recommended workers W_j^{SN} by the worker j. This means that $W_j^{D,i}$ is a subset of the intersection of the set of workers registered to MCS system and the set of delegated workers from the social network of worker j ($W_j^{D,i} \subseteq W_j^{D,i} \cap W$) that comply with the requirements of task i. The index h is used to represent workers other than j.

- The maximum budget to be allocated to the delegated workers should be less or equal to the remaining budget.

$$\sum_{h \in W_j^{D,i}} P_h^i d_h^i \leq B^i - \sum_{k \in W^i \setminus j} P_k^i d_k^i, \tag{23}$$

where W^i is the set of workers selected to perform the task i by the task manager.

- When delegation is allowed, the delegated workers may be located outside the area of coverage. However, they should reach the task location within the remaining response time. This constraint is given by

$$t_h^i \leq t_{max}^i - \max_{k \in W^i} t_k^i. \tag{24}$$

- The probability of not finishing a task for the delegated workers is exponentially reduced (e.g., p^2).
- Only one delegation level is taken into account, which means delegated workers are not allowed to delegate.
- The model incentivizes the recommender by increasing their reputation as linear function of the reputation of the delegated workers. Doing so, the user avoids getting bad reputation for future tasks and the delegation becomes useful for time-constrained tasks.

4.4 Reputation Management

The worker reputation should be updated every time that a given number of tasks are completed in the system. In this procedure, only the workers that have been allocated to tasks are considered regardless of the task completion or delegation. Our framework updates the worker reputation each time that five tasks are completed. The reputation of the participating worker j at time k is estimated as follows:

$$r_j^k = min\left(1, \frac{N_j^{TV}}{N_j^{CT} + N_j^{IT}} + \sum_{l \in W_j^{D_{k,k-1}}} \alpha * r_l^{k-1}\right), \tag{25}$$

where the first term corresponds to the worker performance evaluation in the MCS system. This means how well is the worker performing his assigned tasks. N_j^{TV}, N_j^{CT} and N_j^{IT} indicates the number of completed tasks with true value, number of completed tasks and number of incomplete task respectively for the worker j. The proposed framework considers that a task is answered with a true value by the workers if their answers are equal to the estimated ground truth value by the system, which is given in (26). The second term is related to the delegation mechanism and $W_j^{D_{k,k-1}}$ corresponds to the set of delegated workers by the worker j during the reputation update periods between k and $k-1$. This term is a linear function of the a given parameter α multiplied by the reputation of the delegated workers in the previous period. The value of α can be positive or negative and depends on the task completion of the delegated worker. If the delegated worker did not finish the task this parameter is negative, otherwise it is positive. r_l^{k-1} represents the reputation of worker l in the previous period $k-1$.

We assume that the ground truth of the task (i.e., the correct answer to the location-based sensing task) is binary as in [16], which is reasonable for many real-world situations (e.g., whether the road work at a particular location has been completed, on-shelf availability of a product in a convenience store, etc). In practice, the correct answer of a task i is unknown to the task requester which makes the performance evaluation of the participating workers difficult. To overcome this limitation, the authors in [16] propose to estimate the ground truth from the collected data o_j^i using the majority voting system [20]. This means that the ground truth is the most common response (vote) given by the selected workers (voters) regardless the worker's reputation. Our MCS system uses instead the weighted voting system [31] based on the idea that not all workers have the same influence over the estimated ground truth. In other words, workers with high reputation are more reliable to give good answers than the ones with low reputation. Thus, the estimated ground truth O^i of the task i is given by

$$O^i = \lfloor \frac{\sum_{j \in W^i} X_j^i r_j o_j^i}{\sum_{j \in W^i} X_j^i r_j} - \frac{1}{2} \rfloor + 1. \tag{26}$$

In other words, $O^i = 1$ if the average outcome is greater than $\frac{1}{2}$, and is 0 otherwise. This mechanism allows the model to evaluate the performance of the participating workers. Thus, if the worker response o_j^i is equal to the estimated ground truth O^i. If so, it is assumed that the worker completed the task i with a true value. We keep the historical worker performance for the allocated tasks to update their reputation over time. However, if the ground truth of the sensing tasks are not binaries, it would be good to use other mechanism to estimate the ground truth of the tasks. Regardless the mechanism used to determine the ground truth, the MCS framework needs to know if the reported value of the worker is close to the ground truth of a task. If so, it is assumed that the worker completed the task successfully.

4.5 Running Example

Based on the motivating example presented in Section 3.3, let's suppose now that two new tasks (τ_3, τ_4) arrive to the system one minute after the arrival of task τ_1. The minimum quality of information for both tasks is 1 and the maximum response time is 10 and 5 respectively. In this example, we limit the maximum number of tasks per worker to 2. After 1 minute, the conditions of the workers are shown in Fig. 3.

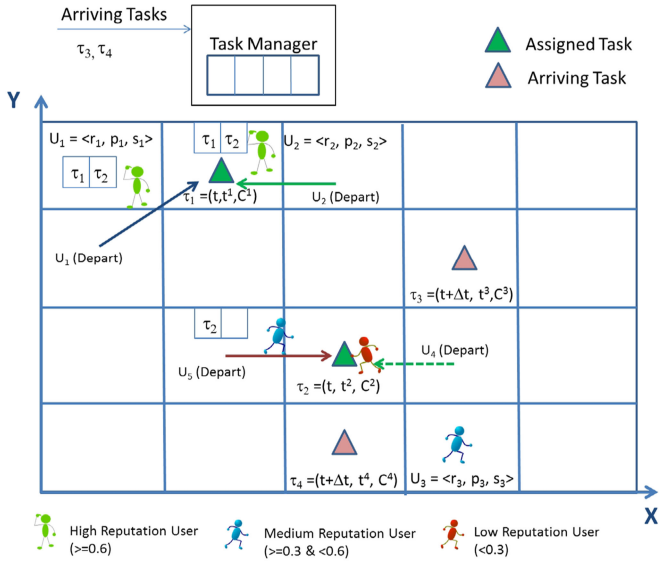


Fig. 3. Scenario with four tasks (2 already assigned and 2 new arriving tasks).

As shown in Table 3, users U_1 and U_2 are still going to the location of their first assigned task (i.e., τ_1) and keep the two tasks in their local queue, which means that they cannot be considered for the allocation task process of the new tasks. Table 6 shows the distances, required time and QoI for each worker regarding the new tasks.

Although both tasks have the same QoI requirement, the algorithm selects task τ_4 to allocate the workers first because of its lowest response time (5 minutes).

One combination meets the requirements of time and QoI as shown in Table 7. After the allocation of this combination to task τ_4, the allocation of task τ_3 is analyzed with the workers having space in their local queue. Only users U_3 and U_4 can be considered for task τ_3 because worker U_5 queue is full. The algorithm also should consider that user U_4 needs some time to finish task τ_4 and the traveled distance to perform task τ_3 (i.e., starting from the location of task τ_4). After running the algorithm for task τ_3, there is no combination that meet the required QoI. However, since the response time of the task is long enough and there are some potential workers that can be selected in a future time, our framework queues task τ_3 in the task manager queue until any other user can be allocated to task τ_3. For example, user U_2 and U_1 can be allocated to another task after 0.33 and 0.41 minutes from the arrival of task τ_3.

After the worker selection is performed for a given task, there is always a probability p that the worker might not be able to finish his/her assigned task. In such scenario, the task manager should wait until all workers deliver their sensed data to see if the QoI is satisfied. When the QoI is deprived, the task manager checks for the recommendations of the worker who did not finish the task and select the delegated workers that can perform the corresponding task within the remaining response time and budget. If the worker did not recommend anyone, this affects the worker reputation since the task is marked as not completed on his historical performance. Otherwise, the task manager rewards or penalizes the reputation based on the recommended participants' performance.

TABLE 6
Worker Metrics for Tasks τ_3 and τ_4

Worker	d_j^3	d_j^4	t_j^3	t_j^4	c_j^3	c_j^4
U_3	2	1	8	4	0.40	0.5
U_4	1.41	1	1.41	1	0.25	0.25
U_5	1.41	1	3.82	3	0.49	0.55

TABLE 7
Set of Workers to Perform Task τ_4

Workers	QoI	Agg QoI	Budget	Response Time	$\frac{AggQoI}{B \times T}$
$[U_3, U_4, U_5]$	1.3	0.3	5.7	4	0.01

5 Benchmark Models and Performance Metrics

In this section, we present two benchmark models (PSO-QoI and QoI-Heu) as well as the performance metrics used to validate and evaluate the proposed framework. The benchmark models use different task allocation algorithms. Also, we extended them using the other components from our framework, namely, queueing schemes, delegation mechanism and reputation management.

5.1 Benchmark Models

5.1.1 QoI Aware PSO-Based Algorithm (PSO-QoI)

This benchmark model aims at maximizing the total quality of information under budget constraints [15]. The authors did not consider time-constrained tasks. To have a fair comparison, we modified their model to include the time constraints. The objective function for this optimization problem is given by

$$\max_{\mathbf{X},\mathbf{P}} \sum_{j \in W} c_j^i X_j^i. \tag{27}$$

We propose to solve the optimization problem under the same constraints (7-8,10,11,12) as in our model using PSO technique. Therefore, the PSO algorithm is similar to the algorithm in 1 but with a different fitness function.

5.1.2 QoI Aware Heuristic Algorithm with Budget Constraints (QoI-Heu)

In [16], a heuristic algorithm, which aims at maximizing the QoI of one task under budget constraint, was proposed. Thus, we modified their algorithm to solve the problem of allocating a location-based task i to a set of candidate workers W subject to a budget limit of $B^i \in R^+$ within the response time $t_{max}^i \in R^+$. The Algorithm 2 presents the modified version of the algorithm in [16].

This algorithm should provides close results to the PSO-QoI algorithm since both algorithms aim at maximizing the QoI per task.

5.2 Metrics

- *Task allocation rate.* This metric represents the percentage of tasks being effectively allocated to workers

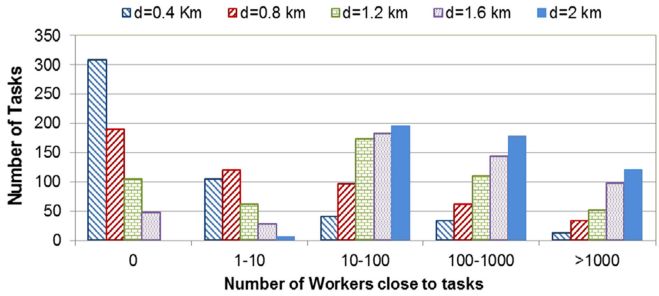


Fig. 4. Range of Workers close to tasks vs Number of task.

$$\overline{\phi_T} = \frac{T_{assigned}}{|T|}, \tag{28}$$

where $T_{assigned}$ is the number of tasks that are effectively allocated and performed within their respective response time.

- *Average Response time per task*. It indicates the average time to perform a location-based task

$$\overline{t_T} = \frac{\sum_{i\in T} \max_{j\in W}(t_j^i)}{T_{assigned}}. \tag{29}$$

- *Average QoI Satisfaction per Task*. This metric measures the average satisfaction of the quality of information over the set of tasks in a given instant

$$\overline{S_{QoI}} = \frac{\sum_{i\in T} max\left(1, \sum_{j\in W} c_j^i X_j^i - C^i\right)}{T_{assigned}}. \tag{30}$$

- *Average Payment per Worker*. It indicates the average payment received by the worker per traveled kilometer and it can be expressed as follows:

$$\overline{P_W} = \frac{\sum_{i\in T}\sum_{j\in W} d_j^i P_j^i}{\sum_{i\in T}\sum_{j\in W} d_j^i X_j^i}. \tag{31}$$

- *Average Estimation Error Rate*. This metric measures how effective is an approach in finding credible workers for a location-based task. The average estimation error rate ϵ is the ratio of incorrect responses (i.e., the number of answers provided by the worker that differ from the ground truth of the assigned tasks) to the total number of assigned tasks. This metric is given by

$$\overline{\epsilon} = \frac{\sum_{i\in T}\left(\frac{\sum_{j\in W} X_j^i \mathbf{1}|_{o_j^i \neq O_i}}{\sum_{j\in W} X_j^i}\right)}{T_{assigned}}. \tag{32}$$

- *Average Budget per Task*. This metric measures the average budget used per task and is given by

$$\overline{B_T} = \frac{\sum_{i\in T}\sum_{j\in W} P_j^i d_j^i}{T_{assigned}}. \tag{33}$$

- *Average Reputation per Worker*. It measures the average reputation of the participating workers in the MCS system

TABLE 8
Workers Distribution per Reputation Level

Number Workers	Initial Distribution		
	Rep_L	Rep_M	Rep_H
200	27	72	101
400	53	141	206
600	70	228	302
800	102	301	397
1,000	118	384	498

$$\overline{R_W} = \frac{\sum_{i\in T}\sum_{j\in W} r_j^k X_j^i}{\sum_{i\in T}\sum_{j\in W} X_j^i}. \tag{34}$$

- *Effective Crowd Size*. This metric measures the number of participating workers in the MCS system

$$\overline{SIZE} = \sum_{i\in T}\sum_{j\in W} X_j^i. \tag{35}$$

6 Simulation Results

For our simulations, we used a real dataset of an existing application: Foursquare. Specifically, two files from this dataset are used: 1) the venues' file that represents the task locations and 2) the users' file that corresponds to the workers' locations. Moreover, we extracted two subsets: 500 venues and 16,836 users to represent the tasks and workers in our model. These subsets allows us to construct realistic spatial crowd sensing scenario to demonstrate that the proposed approach outperforms existing approaches.

Algorithm 2. Heuristic Algorithm (QoI-Heu)

Data: A set of workers W, a spatial task i
Result: Worker selected to the task i and the price to be paid per worker $(X_j^i, P_j^i.)$
begin
  for $i \leftarrow 1$ **to** $|W|$ **do**
    Compute c_j^i according to Eq. (1);
  end
  Rank workers in descending order of their c_j^i ;
  for $j \leftarrow 1$ **to** $min(|W^i|, \lfloor \frac{B}{P_{H,j}^{min}} \rfloor)$ **do**
    $J \leftarrow \min|W^i|, \lfloor \frac{B - P_{H,j}^{max}}{P_{M,j}^{max}} \rfloor$;
    Select a set of workers, W^i who satisfy $d(l_j(t), l_i) \leq R^i$;
    Select a subset,$S^i \in W^i$ workers who satisfy:
    $1 : r_j(t) \geq Th_{ML}$, and;
    $2 : \max_j t_j^i \leq t_i^{req}$;
    (subject to actual availability) with ties broken arbitrarily;
  end
  $P_i = argmax_{S^i} C_{S^i}^i$;
end

We first identify the distribution of the workers in the vicinity of the tasks as the task coverage radius increased from 0.4 to 2 km, which is shown in Fig. 4).

For coverage radius equal to 2 km, it is observed that for 100 tasks it is possible to find between 10-1,000 workers in the vicinity of 100 tasks. Therefore, we select a set of 1,000 workers for our simulations and generate their initial reputation randomly. Table 8 shows the workers distribution per reputation level according to their initial reputation.

TABLE 9
Realistic Scenario Parameters

Name	Description	Value
μ	Required Response time mean	30 min
λ	Task arrival rate	1 Task/minute
N^i_{max}	Maximum number of worker per task i	1 or 5
N^{max}_j	Maximum number of task per worker j	1 or 5
B^i	Maximum Budget per Task	100
P^i_{max}	Maximum Price per Task	5
R^i	Task Coverage Radius	2-5 km
Q	MCS Task Manager Queue Size	0 or 5
r_j	Worker Reputation	0 - 1
β_j	Worker Self-Confidence	0.7 - 1
s_j	Worker Speed	10 - 50 km/h
p	Probability of not finishing a task per worker	0.2

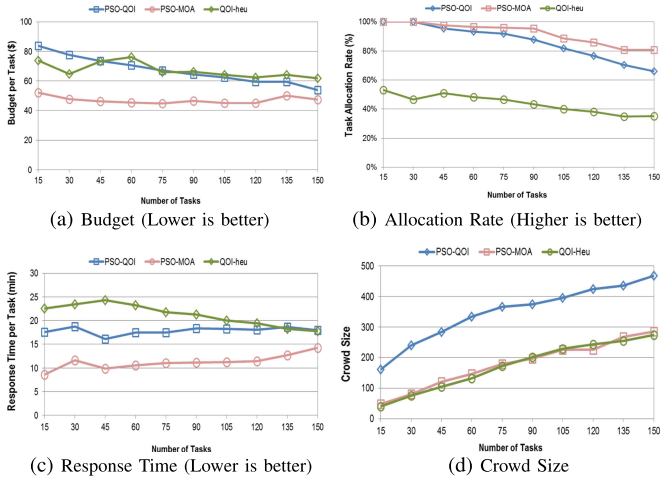
We run the simulations under two different scenarios: an incremental scenario and a realistic scenario.

a) Incremental Scenario. This scenario starts with 15 tasks up to 150 tasks with incremental steps of 15 tasks. This scenario is used to prove that the proposed multi-objective task allocation algorithm can enhance the performance of the benchmark task allocation algorithms for different number of tasks. We modified the three-stage strategy used in [16] to obtain the numerical results. In the first stage, the tasks are sorted according to three parameters: required QoI, the number of workers and requested response time instead of only QoI. The first two parameters are used to sort the tasks in descending order while the response time is used to sort them in ascending order. In such way, the system gives priority to the tasks with short response time. In the second stage, workers are selected if they meet the conditions to perform the task, such as they are located within the coverage area defined by the task and the time to reach the task location is lower than required response time. For this scenario, the performance metrics are presented as a function of the number of tasks in Section 6.1 and the benefits of the worker queue in Section 6.2.1.

b) Realistic scenario. The task arrival process is considered as Poisson Process with parameter λ. Each task identifies the required time to gather the sensed information from the assigned workers. We use an exponential random variable with mean μ to generate the required response time for the tasks. We run extensive event-driven simulations over 1,000 iterations to get to steady state conditions with the parameters described in Table 9.

We assume that workers can finish a task with certain probability $1 - p$. If they do not finish a task, their reputation is affected over the time. The initial worker's reputation is randomly generated. Then, after each five completed tasks, the worker reputation is calculated as the ratio between their completed task with correct answer and his total allocated tasks plus the reward or penalty from the delegation mechanism.

The initial worker self-confidence β_j is assumed to be the percentage of his phone battery ($\beta_j \in [0, 1]$). In our simulations, the confidence value is a decreasing function of the number of tasks completed by the worker due to the battery consumption needed to perform the task and send the collected data to the server. For convenience, the worker confidence during a given period k is estimated as follows:



(a) Budget (Lower is better) (b) Allocation Rate (Higher is better)
(c) Response Time (Lower is better) (d) Crowd Size

Fig. 5. Performance under an incremental scenario (one task per worker).

$$\beta_j(k) = (\beta_j)^{T^k_j}, \tag{36}$$

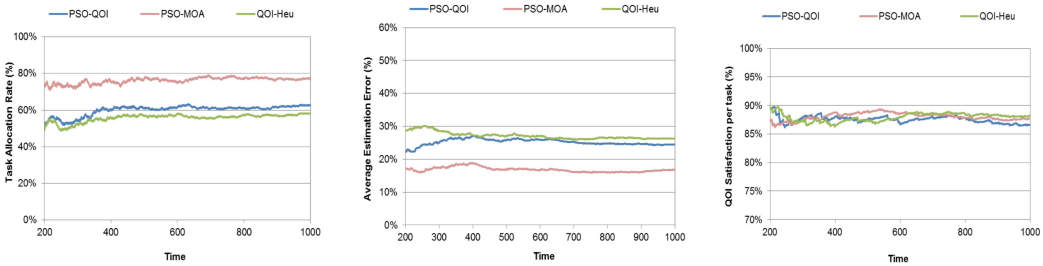
where T^k_j is the number of completed tasks by the worker j during the period k and β_j is the initial measured phone battery level. The battery consumption function is only an example and other functions could be used depending on the technical features and the applications running on the device.

Our event-driven simulation are carried out as follows: At the initial state, one task arrival event is generated. The arrival time follows an exponential distribution. Then, the process starts selecting the event that occurs first (minimum time of occurrence) and according to the type of event (task arrival or departure), several actions are performed.

In the case of arrival, the worker selection for the task starts. If there are available workers, then, the workers that maximize our multi-objective function are selected. Otherwise, the MCS platform queues the task until these workers are available. Once the task is assigned to a set of workers, the type of event is changed to departure and the event time is updated to the service starting time plus the estimated response time from the algorithm and another arrival task event is generated.

In the case of departure, the MCS platform verifies that the selected workers have sent the sensed data. If the QoI of the task is higher than the MCS customer requirements, then, the number of completed tasks is increased by one and the workers' payment and reputation are updated. Otherwise, the MCS platform checks for delegation proposal by the worker who did not finish the task. If the worker did not recommend any other worker, the task is considered as incomplete because the MCS customer won't pay for the sensed data that does not comply with the required quality of information. In the case of delegation, the event time is updated to the total service time estimated by the delegation mechanism. Finally, the MCS platform always checks the queue to allocate the queued tasks as soon as potential workers are available.

For the realistic scenario, the performance metrics over the time are presented in Section 6.1. Then, we show how the delegation mechanism can improve the worker satisfaction without depriving the worker reputation in Section 6.3. In addition, we analyze the impact of each component of



(a) Task Allocation Rate (b) Estimation Error (Lower is better) (c) QoI per Task (Higher is better)

Fig. 6. Performance under a realistic scenario (one task per worker).

the framework individually over the performance metrics in Section 6.4.

6.1 Performance Analysis

Fig. 5 shows the task allocation rate, budget and response time for the three models, where the worker can perform only one task at a time but several workers can be involved in the execution of one task.

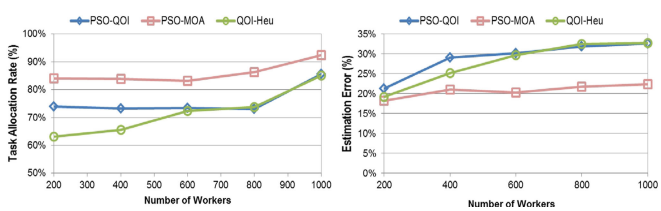
For the case of 15 tasks, PSO-MOA model requires 50 percent less budget than the benchmark models PSO-QoI and QoI-heu and it achieves a task allocation rate of 100 percent. QoI-Heu has a lower task allocation rate (around 55 percent) due to the fact that this model recruits more workers to perform each task, as shown in Fig. 5d. PSO-MOA model also reduces the time to collect the sensed data from the workers by 55 and 40 percent in comparison to the time required by the PSO-QoI and QoI-Heu models respectively.

Fig. 6 depicts the performance metrics for the realistic scenario. In fact, the proposed framework increases the task allocation rate and reduces the estimation error while keeping the same level of QoI satisfaction per task in comparison with the benchmark models. In particular, PSO-MOA model presents a task allocation rate approximately 20 and 25 percent higher than PSO-QoI and QoI-heu models respectively while the estimation error is around 10 percent less than the other two models and the average worker reputation is 10 percent higher than the benchmark models.

In summary, the PSO-MOA model outperforms the benchmark models under the incremental and the realistic scenarios.

TABLE 10
Running Time (sec)

No. Tasks	PSO-QoI	PSO-MOA	QoI-Heu
1	2.1	1.5	0.2
30	28.39	31.47	0.94
60	51.78	60.20	1.78
90	80.22	97.47	3.17



(a) Task Allocation Rate (b) Estimation Error Rate

Fig. 7. Impact of number of available workers.

6.1.1 Complexity

Table 10 presents the running time for the incremental scenario with a number of available workers equal to 1,000. As expected, the running time increases as the number of workers increases. The running time of the three-stage algorithm using PSO for the task allocation is considerably higher than the one using the heuristic algorithm. In the incremental scenario, PSO-QoI algorithm requires an average of 1 minute and 20 seconds to find the worker selection while PSO-MOA requires 1 minute and half approximately for the case of 90 tasks. Nevertheless, it should be noticed from Fig. 5c that the average response time per task plus the running time of the three-stage algorithm is still lower than the response time of the benchmark models. For the realistic scenario, the running time is given by the first row in Table 10 since each task is allocated upon its arrival.

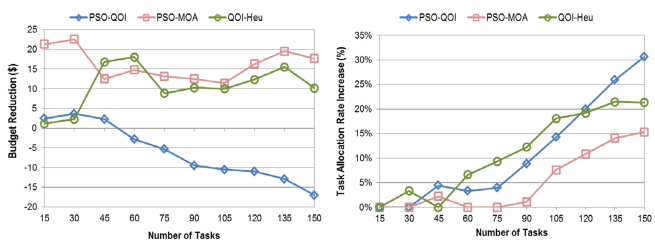
6.1.2 Impact of Crowd Size

Fig. 7 depicts the task allocation rate and the estimation error rate versus the number of available workers for the realistic scenario under steady state conditions.

It can be noticed that our framework presents the highest task allocation rate (85 -92 percent) with the lowest estimation error rate (15 - 22 percent). In particular, QoI-Heu algorithm has similar behavior as the PSO-QoI algorithm when the number of available workers is higher than 600. This means that the QoI-Heu model indeed approximates the results of the PSO-QoI model if the number of available workers is large enough; otherwise, the heuristic algorithm does not perform well.

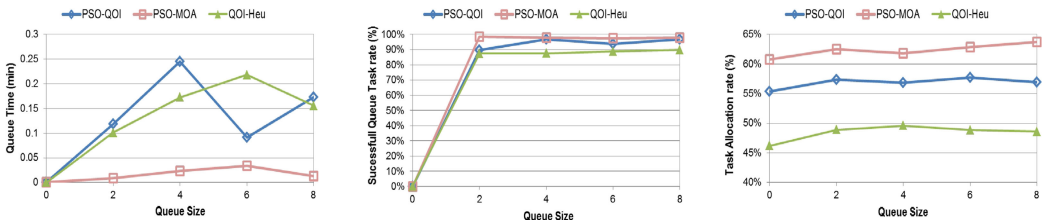
6.2 Queuing Schemes

In the proposed framework, two types of queue are used: a worker queue and the task manager queue as explained in Section 4.2.



(a) Budget Reduction (b) Task Allocation Rate Increase

Fig. 8. Impact of multiple tasks allocation per worker under an incremental scenario (From 1 Task to 5 Tasks per worker).



(a) Queue Time (b) Successful Queued Task Rate (c) Task Allocation Rate

Fig. 9. Impact of size of the priority queue.

6.2.1 Worker Queue

Here, the budget reduction and task allocation rate gain for the incremental scenario is presented in Fig. 8. The budget reduction is the difference between the budgets spent for the cases without and with the local worker queue while the task allocation rate gain is the difference between the task allocation rate for the cases with and without the local worker queue. In Fig. 8a, the PSO-MOA model has the highest budget reduction (25-40 percent) while the QoI-heu model has a budget reduction between 10 and 26 percent of its original budget. PSO-QoI model presents a budget reduction of 5 percent when the number of tasks is less than 50 tasks. For more than 50 tasks, PSO-QoI model presents a budget increase and is an increasing function of the number of tasks. From Fig. 8b, it can be observed that the PSO-MOA model has an increase of the allocation rate between 0 and 15 percent of the original task allocation rate (Fig. 7a). For the case of 150 tasks, the multiple tasks allocation mechanism allows the PSO-MOA model to improve the task allocation rate to 95 percent while PSO-QoI model presents a task allocation rate gain of 30 percent of its original task allocation rate (i.e., 62 percent).

6.2.2 Priority Queue

Fig. 9 presents the impact of the queue size in the task manager side. In particular, Figs. 9a and b present the average waiting time and the successful completion rate for those tasks being queued due to the unavailability of workers to perform them upon their arrival. As we can see, our model has higher rate of successful number of tasks that have been queued and completed than the other two benchmark models. Moreover, the average waiting time is lower than the benchmark models. Both benchmark models present higher waiting time in queue that the PSO-MOA model while their rate of task being successful queued are lower in comparison to our model (see Fig. 9c).

As expected, the priority queue scheme is indeed leveraged by our multi-objective task allocation algorithm. This is owing to the fact that our algorithm also aims at minimizing the response time to perform every task that were already allocated. Therefore, the workers will be available to perform the queued tasks without causing long waiting times and having a high completion task rate.

TABLE 11
Social Graph for Different Number of Workers

No. Workers	Size of social graph	Avg. Known People
200	2,704	13
400	7,506	18
600	17,956	29
800	26,886	33
1,000	34,776	34

TABLE 12
Delegation Mechanism versus Worker's Re-Selection

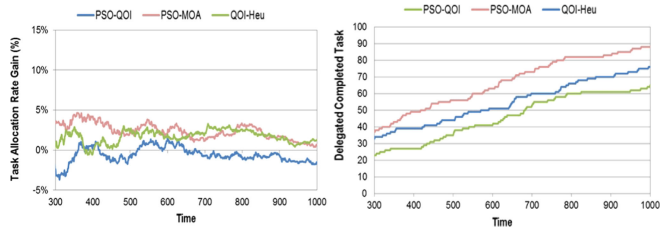
Iteration:	600			1,000		
Model	**Alloc. Rate %**	**Resp. Time (min)**	**Budget $**	**Alloc. Rate %**	**Resp. Time (min)**	**Budget $**
PSO-MOA	80	6	10.2	84	6	10
PSO-MOA-ReSe	81	8.2	11.2	85	9.5	12
PSO-MOA-De	83	7	9.4	85	7.5	9.8

6.3 Delegation Mechanism

The impact of the delegation mechanism over several performance metrics is presented in this section. Although the Foursquare dataset includes social graph information about its users, we could not use it because we extracted a subset of users and its corresponding social network was very limited (approximately two people known per worker) for delegation purposes. Instead, we decided to use the social network from the Enron e-mail dataset.[5] The practice of combining two different datasets is used successfully on other papers [32]. In fact, the two datasets are complementing each other. The Enron email communication network covers all the email communication within a dataset of around half million emails. Nodes of the Enron network are email addresses and if an address i sent at least one email to address j, the graph contains an undirected edge from i to j. The number of nodes is 36,692. Each worker from our subset is mapped to one e-mail address of the Enron e-mail dataset. Table 11 shows the average number of known people per worker after the mapping according to the size of the worker subset. The probability of dropping a task (p) is set to 0.2.

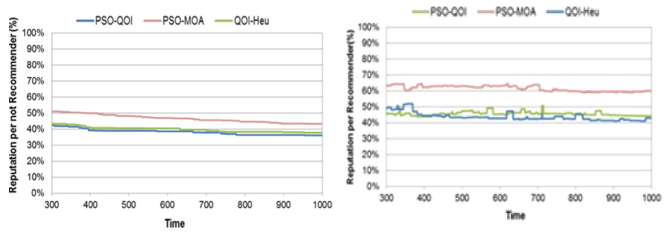
First, we run simulation using PSO-MOA with neither delegation nor reselection (PSO-MOA), with the delegation mechanism (PSO-MOA-DE) and with worker reselection by the MCS system (PSO-MO-ReSe). Table 12 shows the average results for three models at iterations 600 and 1,000. As it can be observed, the task allocation rate for the model that attempts to enhance the task allocation rate of the original model (PSO-MOA) using the delegation scheme presents values similar to the values using the worker re-selection by the MCS system and both schemes enhanced the task allocation rate of the original model. However, the response time and budget are increased for the model using the reselection compared to the one with the delegation mechanism. This is

5. https://snap.stanford.edu/data/email-Enron.html



(a) Gain Task allocation rate (b) Completed delegated tasks

Fig. 10. Impact of the delegation mechanism.



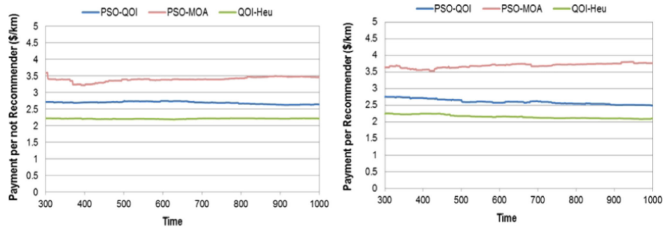
(a) Worker who does not recommend (b) Recommender

Fig. 11. Reputation per worker.

because the MCS system does not select workers among the full set of workers, but instead relies on the worker's recommendation and reduces its search to a subset of workers.

Fig. 10a presents the task allocation rate gain when the delegation mechanism is incorporated to the three models and a worker can be assigned to only one task at a time. It can be noticed the highest task allocation rate gain is usually obtained using the PSO-MOA model. Fig. 10b depicts the number of delegated and completed tasks for each model and shows that the proposed model is able to delegate more tasks than the benchmark models. Fig. 11 presents the average reputation per workers. In particular, these figures show the average reputation for workers who do not recommend and recommenders. It is worth noticing that the proposed delegation mechanism can effectively incentivize a worker to recommend other workers from his social network and allows them to increase their reputation and thanks to this increase they can keep their payment higher than in the other two models (see Fig. 12).

The benchmark models fail to incentivize through the proposed delegation mechanism owing to the fact that these two models require more workers to carry out one task and there is less available workers that can be reached under a recommendation.



(a) Worker who does not recommend (b) Recommender

Fig. 12. Payment per worker.

In summary, the proposed delegation mechanism is able to incentivize workers, guaranteeing their payment by means of keeping their high reputation level. In particular, our model increases the number of tasks being delegated owing to the fact that our model uses less number of workers per task. Therefore, there is a high probability that one worker who does not finish his/her task can affect the total required QOI per task.

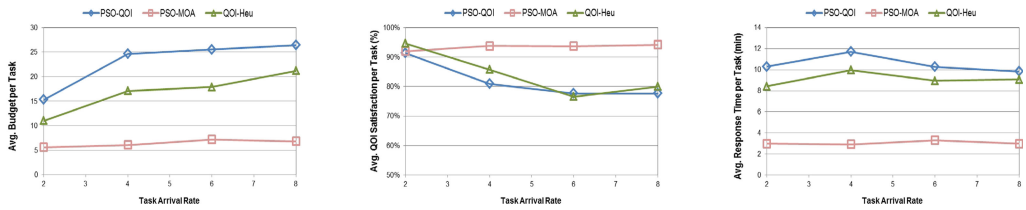
6.4 Evaluation of Each Component of the Realistic Framework

Table 13 presents the impact of each component of the solution on the performance metrics for the three models. The first column indicates the name of the metric. Then, the five consecutive columns present the results for each model. The first column under each model indicates the case where only one task can be carried out per worker without any additional component. The second to fourth columns correspond to the results for each model considering only one component of the proposed framework and the fifth column combines all the components.

Each component leads to an enhancement over the performance metrics in comparison with the model that allocates just one task per worker. In particular, the proposed framework (PSO-MOA) using all components can allocate around 81 percent of the requested tasks with a QoI satisfaction of 90 percent and an estimation error of 19 percent. PSO-QoI model allocates 70 percent of the requested tasks with a QoI satisfaction of 90 percent and an estimation error of 32 percent. The heuristic model allocates 64 percent of the requested tasks with a QoI task satisfaction of 94 percent and an estimation error of 32 percent. As expected, the proposed framework reduces the budget and response time per task around 70 and 50 percent in comparison with the two benchmark models respectively.

TABLE 13
Impact of Each Component over the Performance Metrics

Performance	PSO-QoI					PSO-MOA					QoI-Heu				
Metrics	One Task	Priority Queue	Multipe Tasks	Deleg	All	One Task	Priority Queue	Multipe Tasks	Deleg	All	One Task	Priority Queue	Multipe Tasks	Deleg	All
Estimation Error (%)	28	28	27	31	32	18	19	19	20	19	28	27	29	33	32
Average Reputation (%)	51	51	50	50	51	63	65	62	61	65	49	51	51	49	49
Task Allocation Rate (%)	58	58	69	70	70	79	78	79	80	81	55	59	60	67	64
QoI Satisfaction (%)	86	87	86	90	90	89	89	89	89	90	88	87	88	94	94
Budget ($)	37.40	35.32	33.63	39.04	36.85	11.61	12.72	12.00	11.51	11.88	28.58	27.954	28.55	29.46	30.23
Time (min)	18.79	18.81	16.93	21.36	21.53	5.64	7.87	5.66	5.96	8.32	17.42	18.43	17.00	20.39	22.09
Running Time (min)	1.15	1.18	1.17	1.06	1.11	1.42	1.55	1.50	1.46	1.41	0.05	0.04	0.05	0.04	0.05



(a) Budget (b) QOI Satisfacion (c) Time

Fig. 13. Performance metrics for variable task arrival rate.

Finally, the running times for PSO-based models are higher than the heuristic model (1.1 and 1.5 minutes for the PSO-QoI and PSO-MOA models respectively). In particular, PSO-MOA model presents the total response time including the response time and running time lower than the one for the QoI-Heu model. Moreover, the running time for the PSO-based algorithms can be further reduced using a PSO variant. However, this is out of the scope of the paper and it can be investigated as future work.

6.5 Impact of Variable Task Arrival Rate

Here, the performance of the three models when the MCS system reaches the steady state conditions is presented. Fig. 13 shows the quality of information, the budget and the average response time per task for variable task arrival rate λ (i.e., 2-8 tasks/min). For this scenario, the task manager queue size Q was set up to 5, the worker queue size is equal to 2, the delegation mechanism is used and available number of workers is equal to 1,000. As expected, our framework can effectively reduce the budget and response time. It can be noticed that as the task arrival rate increases, the QoI for the benchmark models decreases, while in our model slightly increases.

In summary, our solution reduces the budget by 5 to 35 percent and accommodates around 20 percent more tasks than the benchmark models. Moreover, its response time is lower than the response time of the QPI-Heu and PSO-QoI models. It is also shown that the MCS framework encourages the workers through the delegation mechanism by keeping the level of workers' reputation high. Finally, the queuing schemes allow the framework to further increase the task allocation rate without compromising the required response time. Therefore, our MCS framework increases the average quality of information per task, reduces the budget and minimizes the response time.

7 Conclusion

A service computing framework for task management in MCS systems is introduced. It consists of a multi-objective task allocation algorithm, queuing schemes and a delegation mechanism. The multi-objective task allocation algorithm was implemented using PSO to find a compromise between the aggregated quality of information/budget ratio and the response time. This algorithm was compared with two algorithms: one meta-heuristic (PSO-QoI) and one heuristic (QoI-Heu) that aim at maximizing the QoI per task under an incremental scenario and a realistic scenario. For the incremental scenario, our solution reduces the budget between 5 and 35 percent, accommodates around 20 percent more tasks than the benchmark models while requiring less time for the data collection. For the realistic scenario, the proposed framework provides incentives to the workers through the delegation mechanism. In fact, our model presents an average recommenders' reputation higher than 58 percent while the other two models present a reputation lower than 50 percent. Furthermore, the queuing schemes allow the proposed framework to further increase the allocation rate without compromising the required response time. As future work, we propose to investigate the optimization of the workers' route when allocated to several tasks. We also propose to investigate the implementation of other delegation mechanisms and the analysis of the framework performance under mobile users arrival/departure processes to determine when the engagement strategies are needed to guarantee the QoI per task.

Acknowledgments

This work has been partially funded by Khalifa University Internal Research Level 1, fund code 210068.

References

[1] B. Guo, et al., "Mobile crowd sensing and computing: The review of an emerging human-powered sensing paradigm," *ACM Comput. Surveys*, vol. 48, no. 1, pp. 7:1–7:31, 2015.

[2] V. Coric and M. Gruteser, "Crowdsensing maps of on-street parking spaces," in *Proc. IEEE Int. Conf. Distrib. Comput. Sensor Syst.*, May 2013, pp. 115–122.

[3] N. Maisonneuve, M. Stevens, M. Niessen, and L. Steels, "Noisetube: Measuring and mapping noise pollution with mobile phones," in *Proc. Inf. Technol. Environmental Eng.*, 2009, pp. 215–228.

[4] B. Guo, H. Chen, Z. Yu, X. Xie, S. Huangfu, and D. Zhang, "FlierMeet: A mobile crowdsensing system for cross-space public information reposting, tagging, and sharing," *IEEE Trans. Mobile Comput.*, vol. 14, no. 10, pp. 2020–2033, Oct. 2015.

[5] X. Zhang, et al., "Incentives for mobile crowd sensing: A survey," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 54–67, Jan./Mar. 2016.

[6] M. H. Cheung, R. Southwell, F. Hou, and J. Huang, "Distributed time-sensitive task selection in mobile crowdsensing," in *Proc. 16th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, 2015, pp. 157–166.

[7] Y. Liu, B. Guo, Y. Wang, W. Wu, Z. Yu, and D. Zhang, "TaskMe: Multi-task allocation in mobile crowd sensing," in *Proc. ACM Int. Joint Conf. Pervasive Ubiquitous Comput.*, 2016, pp. 403–414.

[8] 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-Mach. Syst.*, vol. 47, no. 3, pp. 392–403, Jun. 2017.

[9] I. Koutsopoulos, "Optimal incentive-driven design of participatory sensing systems," in *Proc. IEEE INFOCOM*, Apr. 2013, pp. 1402–1410.

[10] N. Do, C.-H. Hsu, and N. Venkatasubramanian, "CrowdMAC: A crowdsourcing system for mobile access," in *Proc. 13th Int. Middleware Conf.*, 2012, pp. 1–20.

[11] H. Shah-Mansouri and V. Wong, "Profit maximization in mobile crowdsourcing: A truthful auction mechanism," in *Proc. IEEE Int. Conf. Commun.*, 2015, pp. 3216–3221.

[12] H. Yu, C. Miao, Z. Shen, C. Leung, Y. Chen, and Q. Yang, "Efficient task sub-delegation for crowdsourcing," in *Proc. 29th AAAI Conf. Artif. Intell.*, 2015, pp. 1305–1312.
[13] D. W. Barowy, C. Curtsinger, E. D. Berger, and A. McGregor, "AutoMan: A platform for integrating human-based and digital computation," *Commun. ACM*, vol. 59, no. 6, pp. 102–109, 2016.
[14] J. Banks, J. Carson, B. L. Nelson, and D. Nicol, *Discrete-Event System Simulation*, 4th ed. Englewood Cliffs, NJ, USA: Prentice Hall, Dec. 2004.
[15] C. Zhou, C.-K. Tham, and M. Motani, "QOATA: Qoi-aware task allocation scheme for mobile crowdsensing under limited budget," in *Proc. IEEE International Conference on Intelligent Sensors, Sensor Networks and Information Processing*, Apr. 2015, pp. 1–6.
[16] H. Yu, C. Miao, Z. Shen, and C. Leung, "Quality and budget aware task allocation for spatial crowdsourcing," in *Proc. Int. Conf. Autonomous Agents Multiagent Syst.*, 2015, pp. 1689–1690.
[17] J. Liu, H. Shen, and X. Zhang, "A survey of mobile crowdsensing techniques: A critical component for the internet of things," in *Proc. 25th Int. Conf. Comput. Commun. Netw.*, Aug. 2016, pp. 1–6.
[18] L. Jaimes, I. Vergara-Laurens, and M. Labrador, "A location-based incentive mechanism for participatory sensing systems with budget constraints," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun.*, Mar. 2012, pp. 103–108.
[19] L. Mo, et al., "Optimizing plurality for human intelligence tasks," in *Proc. 22nd ACM Int. Conf. Inf. Knowl. Manage.*, 2013, pp. 1929–1938.
[20] L. Varshney, J. Rhim, K. Varshney, and V. Goyal, "Categorical decision making by people, committees, and crowds," in *Proc. Inf. Theory Appl. Workshop*, Feb. 2011, pp. 1–10.
[21] J. Surowiecki, *The Wisdom of Crowds*. New York, NY, USA: Anchor, 2005.
[22] Foursquare dataset, 2013. [Online]. Available: https://archive.org/details/201309_foursquare_dataset_umn
[23] V. D. Blondel, et al., "Data for development: The D4D challenge on mobile phone data," *CoRR*, 2012. [Online]. Available: http://arxiv.org/abs/1210.0137
[24] G. A. Korn and T. M. Korn, *Mathematical Handbook for Scientists and Engineers: Definitions, Theorems, and Formulas for Reference and Review*. Mineola, NY, USA: Dover Publications, 2000.
[25] S. He, D.-H. Shin, J. Zhang, and J. Chen, "Toward optimal allocation of location dependent tasks in crowdsensing," in *Proc. IEEE INFOCOM*, Apr. 2014, pp. 745–753.
[26] M. R. Garey and D. S. Johnson, *Computers and Intractability; A Guide to the Theory of NP-Completeness*. San Francisco, CA, USA: Freeman, 1979.
[27] E. Elbeltagi, T. Hegazy, and D. Grierson, "Comparison among five evolutionary-based optimization algorithms," *Adv. Eng. Inform.*, vol. 19, no. 1, pp. 43–53, 2005.
[28] Perez. R. and K. Behdinan, "Particle swarm optimization in structural design," in *Swarm Intelligence: Focus on Ant and Particle Swarm Optimization*. Vienna, Austria: Itech Education and Publishing, 2007, pp. 532–555.
[29] D. Bratton and J. Kennedy, "Defining a standard for particle swarm optimization," in *Proc. IEEE Swarm Intell. Symp.*, 2007, pp. 120–127.
[30] F. Lobo, *Parameter Setting in Evolutionary Algorithms*. Berlin, Germany: Springer, 2007.
[31] P. Tannenbaum, *Excursions in Modern Mathematics*. Cambridge, U.K.: Pearson, 2013.
[32] O. A. Wahab, J. Bentahar, H. Otrok, and A. Mourad, "Optimal load distribution for the detection of VM-based DDoS attacks in the cloud," *IEEE Trans. Serv. Comput.*, 2017.

Rebeca Estrada received the BSc degree in computer engineering from ESPOL, in 1995, the master's degree with specialization in telecommunication from ITESM, Monterrey, México, in 1998, and the engineering doctoral degree from École de Technologie Supérieure, University of Quebec, in 2014. She holds an associate professor position in the department of Electrical Engineering and Computer Science at Escuela Superior Politécnica del Litoral (ESPOL), Guayaquil, Ecuador. Currently, she is leading a research group (ReDIT) with focus on Networking and Technological infrastructure at ESPOL. Her research work is oriented to resource allocation in mobile crowd-sensing systems, cloud computing systems and two-tier wireless network.

Rabeb Mizouni received the MSc and PhD degrees in electrical and computer engineering from Concordia University, Montreal, Canada, in 2002 and 2007, respectively. She is an assistant professor in electrical and computer engineering with Khalifa University. Currently, she is interested in the deployment of context aware mobile applications, crowd sensing, software product line and cloud computing.

Hadi Otrok received the PhD degree in ECE from Concordia University. He holds an associate professor position in the Department of ECE, Khalifa University, an affiliate associate professor in the Concordia Institute for Information Systems Engineering, Concordia University, Montreal, Canada, and an affiliate associate professor in the electrical department at cole de Technologie Suprieure (ETS), Montreal, Canada. He is an associate editor at: *Ad-Hoc Networks* (Elsevier), the *IEEE Communications Letters*, *Wireless Communications and Mobile Computing* (Wiley). He co-chaired several committees at various IEEE conferences. He is an expert in the domain of computer and network security, web services, ad hoc networks, application of game theory, and cloud security. He is a senior member of the IEEE.

Anis Ouali received the BSc degree in computer engineering from LEcole Nationale des Sciences de l'Informatique (ENSI), Tunisia, in 2000, the MSc degree in computer science from Universite du Quebec A Montreal (UQAM), Canada, in 2004, and the PhD degree from the Electrical and Computer Engineering Department, Concordia University, Montreal, Canada, in 2011. His research interests include P2P networks for video streaming, distributed computing and content adaptation. He joined EBTIC in 2010 and is currently working in the network optimization team which focuses on solving network design related problem.

Jamal Bentahar received the PhD degree in computer science and software engineering from Laval University, Canada, in 2005. He is a full professor with Concordia Institute for Information Systems Engineering, Faculty of Engineering and Computer Science, Concordia University, Canada. From 2005 to 2006, he was a postdoctoral fellow with Laval University, and then Simon Fraser University, Canada. His research interests include services computing, applied game theory, computational logics, model checking, multi-agent systems, and software engineering. He is a member of the IEEE.