Interval Tree-Based Task Scheduling Method for Mobile Crowd Sensing Systems

Ahmed A. A. Gad-ElRab, and Almohammady S. Alsharkawy

***Abstract*—Nowadays there is an increasing demand to provide a real-time environmental information. So, the growing number of mobile devices carried by users establish a new and fast-growing sensing paradigm to satisfy this need, which is called Mobile Crowd Sensing (MCS). The MCS uses different sensing abilities to acquire local knowledge through enhanced mobile devices. In MCS, it is very important to collect high-quality sensory data that satisfies the needs of all assigned tasks and the task organizers with a minimum cost for the participants. One of the most important factors which affect the MCS cost is how to schedule different sensing tasks which must be assigned to a smartphone with the objective of minimizing sensing energy consumption while ensuring high-quality sensory data. In this paper, the problem of scheduling the tasks which have mutual sensor is formulated and a scheduling method to minimize the energy consumption by reducing the sensor utilization is proposed. The proposed method will incentive the users to participate in multiple tasks at the same time, which minimizes the total cost of the performed tasks and increases his rewards. The experimental results by using synthetic and real data show that the proposed scheduling method can minimize the energy consumption and preserve the task requirements compared to existing algorithms.**

***Index Terms*—Mobile Crowd Sensing, Task Scheduling, Time series, Intervals Tree.**

I. Introduction

Mobile Crowd Sensing (MCS) in dynamic environments is an emerging computing paradigm that utilizes everyday mobile devices to form crowd sensing networks. It allows the growing number of mobile users to share local knowledge acquired by their sensor-enhanced devices that can be used for monitoring pollution level, noise level, or traffic condition, etc. [1]. This makes mobile devices an excellent platform for sensing the environment. The new generations of mobile devices have multiple embedded sensors (e.g., GPS, an Accelerometer, a Compass, a Gyroscope, a Camera, an Ambient Light, etc.). Also, the mobile devices can easily communicate with external sensors via any of the built-in interfaces, including Bluetooth, Infrared, or Wi-Fi. [2]

The collected information by mobile devices with the support of the cloud-based (applications, services or resources) for data processing, analysis, and visualization makes MCS a good platform that can often replace static sensing infrastructures, and enable a broad range of applications such as social event detection, disaster information collection [3], public safety [4], environmental monitoring [5], [6], and [7], and traffic planning [8], etc. By considering the density of the mobile devices around the world, the MCS applications can have multiple participants in which they can be involved in the same sensing activity. For example, monitoring the environment in a disaster area or sensing the traffic information in an intersection.

Due to the nature of MCS applications, there are several issues to gather the sensory data. One of these issues is how to motivate the participants to contribute in the sensing tasks and collect a high-quality sensory data and upload it to the task requester. It is very hard to motivate the user to contribute in the MCS tasks without any rewards such as monetary, social, entertainment or game-centric rewards. So, to attract the user to perform a MCS task, the MCS system must consider the rewards and the task execution cost for the user. This paper focuses on how to minimize the execution cost of a MCS task by encouraging a user to participate in this task.

In a real case, the participant can perform more than one task at the same time, and these tasks may have common sensors. Therefore, to minimize the task execution cost, it is needed to decrease the sensor utilization which will preserve the smartphone energy. Thus, the participant needs an efficient task scheduling method to satisfy this goal.

In MCS applications which request to measure data by sensor reading, the sensed data for some sensors readings may change quickly over time (such as GPS), while the other may change slowly over time (such as temperature, humidity and light). Therefore, this feature can be employed to schedule multiple tasks which request the same sensor readings. Scheduling multiple tasks will save the energy by reducing the sensors utilization.

In this paper, the main objective is to design a MCS task scheduling method. This method will be performed on the cloud server before the task assignment, which will minimize time and energy. The cloud server will try to group a number of tasks which have common sensors, and will create a uniform task execution time line for all tasks in the same group. This method will allow the participant to perform a group of tasks with a minimal energy consumption, which will incentive the participants to contribute in many tasks. After building the uniform time line, the participants can upload the collected data for all tasks at the same time, which will reduce the data uploading time and the consumed energy from data connection.

Manuscript received November 20, 2017; revised March 10, 2018. Date of publication March 15, 2018.

Authors are with the Department of Mathematics, Faculty of Science, Al-Azhar University - Cairo, Egypt (e-mails: {asaadgad, alm.alsharkawy}@azhar.edu.eg).

Digital Object Identifier (DOI): 10.24138/jcomss.v14i1.429

The major contributions of this paper can be summarized as follows.

1) Formulating the problem of scheduling multiple MCS tasks, by considering the energy efficient, delay time, and load balance for the participant and the server.
2) Proposing a task scheduling method to minimize the energy consumption and increase the user rewards and decrease the task's payments by using intervals tree and time series [9].

The rest of this paper is organized as follows. Section 2 gives a background on mobile crowd sensing applications, and introduces the previous related works to task scheduling in MCS. Section 3 presents the overall preliminaries, assumptions, and objectives and formulates the task scheduling problem in MCS. Section 4 gives a brief description of the time series method that is used to eliminate the trends in the data, and introduce the intervals tree which is used to compute the overlapping between time instants of the tasks. Section 5 introduces the MCS task scheduling method, and presents the strategies for the implementation of the proposed method. Section 6 shows the practical experiments to evaluate the MCS task scheduling method. Section 6 concludes the paper.

II. Related Work

To encourage users for contributing in the MCS tasks, there is a lot of incentive mechanisms have been proposed. Motivating human participation and improving the quality of contribution is crucial to the success of MCS tasks. An overview of the role of motivation in crowd sensing is given in [10], [11], and [12], where different motivational factors have been discussed, such as self-efficacy, sense of community, use of contextual clues, and so on. There are several crowdsourcing platforms that provide workers with non-monetary incentives such as entertainments, social and educational opportunities [13] and [14].

In [15] a novel MCS incentive mechanism called LBSN (location-based social network) to enhance sensing quality was proposed. This mechanism supports quality-enhanced data collection and is differed from the traditional monetary-based incentive mechanisms. The LBSN method powered model is leveraged for dynamic budgeting and proper worker selection, and a combination of multi-facet quality measurements and a multi-payment-enhanced reverse auction scheme is used to improve sensing quality. The authors in [16], reviewed the existing research, and proposed a ”5W1H” model to serve the study on incentive mechanism. Moreover, they conduct analysis on how to design an incentive mechanism with a case ”Noise Map”. Also, the authors in [17] have proposed an incentive mechanism that selects the worker candidates statically, and then dynamically selects winners after bidding. The proposed incentive mechanism includes two algorithms which are an improved two-stage auction algorithm (ITA) and a truthful online reputation updating algorithm (TORU). A novel truthful online auction mechanism has proposed in [18] that can efficiently learn to make irreversible online decisions on winner selections for new MCS systems without requiring previous knowledge of users.

Recently, incentive mechanisms for the mobile crowd sensing have been widely studied in the literature [19], [20] and [21]. [22] and [23] take the quality of collected data into account when designing the incentive mechanisms for the mobile crowd sensing. Kawajiri et. al. [24] designed incentive mechanisms to steer mobile users to collect data at certain locations, and then improve the overall quality of sensing services. According to Reddy et. al. [25], monetary incentives often increase interest in participating and reinforce good data collection habits. Monetary incentive mechanisms for MCS can be categorized into two modes: online and offline. In the online mode, the participants arrive one by one in a random order and the platform has to decide whether a task should be assigned to a participant upon her arrival based solely on the information of previous participants. However, this solution can not guarantee the same winning chance for every body because the first batch of participants is only used to train the threshold and have no chance to win. This may result in task completion delays as well as participant dropping out. Due to the limitations, more incentive mechanisms work in the offline mode, which assumes that the platform has the whole information about the users and their devices (e.g., behaviors, sensing costs) [20], [26], and [19].

In [27], the authors considered the problem of scheduling sensing tasks assigned to a smartphone with the objective of minimizing sensing energy consumption while ensuring quality of sensing for the case in which each sensing task only requests data from a single sensor. Firstly, it requests from each task organizer to assign a value which represents the quality of collected data, then uses a bell-shaped function within a value range between 0 and 1 to obtain time instants that is close to the requested quality. The main drawbacks of this work are: (1) the task requester may be unqualified to define the quality needed for the collected data, thus request to assign the quality value from the task requester is unacceptable. (2) it follows a sequential search to obtain the tasks time instants, which will cost the platform time and energy.

In this paper, to overcome the drawbacks of existing scheduling models, a new task scheduling method is proposed. This method uses time series and interval tree [9] to find the common time instant for different tasks that use common sensors. The proposed task scheduling method is similar to the task scheduling model in [9] which tries to find the common time instants for multiple tasks. But our method uses different strategy to find these common time instants and it differs from the method [27] in the following points:

1) Classifies and schedules the tasks on the cloud side before task assignment, which will minimize the energy consumption, increase the total rewards for the participants, and decrease the task payments for the task organizer.
2) Uses a mathematical method (Time Series and Trend Elimination by Differencing) Sec.IV-B to obtain the sensor changing time , which is more accurate and reliable than the method in [27].
3) Uses the interval tree to compute the overlapped interval for the tasks which will decrease the cost of obtaining a new time instant for the tasks, while the method in [27] searches sequentially to get a new time instant which has

a very high cost.

III. Task Scheduling Problem (TSP)

In this section, the tasks scheduling problem in the crowd sensing will be formulated. Here, a typical client-server crowd sensing architecture is considered where a large number of mobile devices are tasked into community-based data gathering in a specific *Area of Interest* (AoI). The task requester defines the task requirements and submits this information to a platform residing in a cloud server to distribute the task information to all users in the AoI. This information includes the specific location to collect data, sensing time, sensing interval, and the participation payments. The cloud server will undertake the task till the task completion, the cloud-based platform jobs consists of (distribute the task, select the participant, receive the collected data). The sensory data which are collected by participants are reported (e.g., through cellular networks or WiFi) to the central application cloud server. A task normally specifies multiple types of sensory data to be collected based on the application requirements. The cloud server can handle multiple sensing tasks from different task organizers at the same time, and a user device may be involved in multiple concurrent sensing tasks.

The MCS system model which was considered in this paper consisting of a platform residing in a cloud server and a set of N users, denoted as $U = \{u_1, \ldots, u_N\}$. The users execute a set of M sensing tasks, denoted as $\mathbb{T} = \{\tau_1, \ldots, \tau_M\}$ and send their sensory data to the platform, each $\tau_h \in \mathbb{T}$ has a value $V_h > 0$ to the platform, where $V_h = \sum_{i \in N} \Omega_{hi}$ is the total benefit of the platform and Ω_{hi} is evaluated through the sensing time submitted by user i. For example, Ω_{hi} may be representing the quality of the received images from the user i. Each user in the AoI may be involved in more than one task at the same time. Also, each task has a sensing time sequence $\eta_h = \{t_{h1}, t_{h2}, \ldots, t_{hs}\}$ which is a sequence of time instants at which the sensor readings are requested to be collected and s is the number of time instants for task h. When the cloud server received a new task from a task requester, the cloud server will classify the arrived task according to the requested sensors. The cloud server will create a number of classes $\hat{\mathbb{T}} = \{\hat{\tau}_1, \hat{\tau}_2, \ldots, \hat{\tau}_K\}$ where K is the number of task classes, each class has a number of tasks that have mutual sensor or sensors $\hat{\tau}_k = \{\tau_1, \ldots, \tau_p\}$, p is the number of tasks in tasks class $\hat{\tau}_k$. For each class of tasks, the cloud server will obtain a uniform time line μ, this time line will define the sensing time sequence for all the tasks in the class, $\mu_{\hat{\tau}_k} = \{v_1, v_2, \ldots, v_L\}$, where L is the number of time sequences in the uniform time line. At each time instant v_l in the uniform time line, a number r of tasks will be performed such that $1 \leq r \leq p$. The users in the AoI who carry a mobile device that are characterized by some embedded sensory capabilities (e.g. accelerometers, gyroscopes, microphones, cameras, etc.) are potentially available to undertake the execution of a class of tasks assigned by the cloud server.

A. Problem Formulation

The objective function of TSP is to minimize the consumed energy from sensing data by the task's sensors. The minimization will be issued by decreasing the number of sensing times while persevering the quality of sensing. This objective function is formulated as follows.

$$\min_{l \in L} \quad v_l \tag{1}$$

Subject to:

$$\eta_h = \{t_{h1}, t_{h2}, \ldots, t_{hs}\}, \quad \forall h \in \hat{\tau}_k \tag{2}$$

$$v_l = t_{hs}, \quad \forall h \in \hat{\tau}_k \text{ and } s \leq L \tag{3}$$

Constraint 2 means that minimizing the v_l without affecting the original time instants for each task in the tasks class, in other words, after minimizing the v_l, each task in the class will have a number of time instants equal to the requested sensing time instants. Constraint 3 means that each time instant in v_l will have a corresponding time instant in the requested set of time instants of all tasks in class $\hat{\tau}_k$.

The minimization comes from finding the overlapped time instants and obtaining a new time instant that fits to all overlapped tasks without influencing the quality of the collected data. Here, the quality of data means that the number of sensor readings must equal to the desired number which is requested by the task organizer and the readings in the new time instant must equal to the readings in the original time instant.

IV. The Proposed MCS Tasks Scheduling Method

To solve the scheduling problem in MCS, a new task scheduling called *Interval Tree-Based Task Scheduling* method (ITBTS) is proposed. In this section, the ITBTS method, will be described in details. The task organizer should provide the platform by a tuple of task information and requirements sensors list, sensing time instants.

A. Basic Idea

The proposed task scheduling method ITBTS depends on unifying the sensing time instants of several tasks which have common sensors. In case of two tasks, for example, to unify the sensing time instant of these two tasks, the cloud server will try to find a new time instant whereas the sensing quality does not be affected. So, the cloud server firstly will define a β_x, value which represents the stability time for a sensor x. For example, when measuring the temperature in a city, the temperature may change after an amount of time, the value changing time β will be called sensor value stability time. The cloud server will obtain the value of β by using the previous measures of the sensor in each AoI and analyze these measures by using the time series analysis methods. Then, trend elimination by differencing will be used to obtain the value of β. Once the value of β is obtained, it can be used to unify the time instants for several tasks that have mutual sensors by using the intervals tree.

In the rest of this section, time series model and interval tree will be described in details, then the proposed ITBTS will be introduced.

B. Time Series and Trend Elimination by Differencing

A time series is a collection of data recorded over a period of time, weekly, monthly, quarterly, or yearly. A time series is a set of observations y_t, each one being recorded at a specific time t. Time series plots can reveal patterns such as random, trends, level shifts, periods or cycles, unusual observations, or a combination of patterns [9]. Patterns commonly found in time series data, a trend is evolutionary movement, either upward or downward, in the value of the variable. Trends may be long-term or more dynamic and of relatively short duration. Seasonality is the component of time series behavior that repeats on a regular basis, such as each year. Sometimes the data will be smoothed to make identification of the patterns more obvious Fig.1 shows the pattern in the temperature time series.

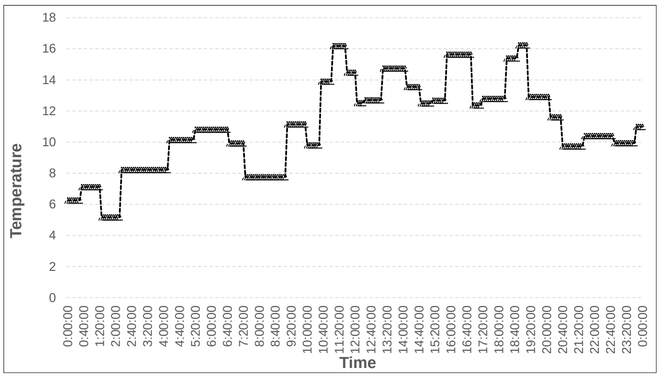

Fig. 1: Temperature changing over the time

One of the most used approaches to remove trend is by differencing the data [28] and [9]; that is, applying the difference operator to the original time series to obtain a new time series. To eliminate the trend term by differencing, the lag-1 difference operator $\bigtriangledown$ is defined as follows.

$$\bigtriangledown X_t = X_t - X_{t-1} = (1 - B)X_t, \tag{4}$$

where B is the backward shift operator, so

$$BX_t = X_{t-1} \tag{5}$$

Powers of the operators B and $\bigtriangledown$ are defined in the obvious way, i.e., $B^j(X_t) = X_{t-j}$ and $\bigtriangledown^j(X_t) = \bigtriangledown(\bigtriangledown^{j-1}(X_t))$, $j \geq 1$, with $\bigtriangledown^0(X_t) = X_t$. Polynomials in B and $\bigtriangledown$ are manipulated in precisely the same way as polynomial functions of real variables. For example,

$$\begin{aligned} \bigtriangledown^2 X_t &= \bigtriangledown(\bigtriangledown(X_t)) \\ &= v(1-B)(1-B)X_t = (1-2B+B^2)X_t \\ &= X_t - 2X_{t-1} + X_{t-2} \end{aligned} \tag{6}$$

In this paper, the data differencing method is used to calculate the sensor's value stability time β for each sensor in the MCS systems. β represents the minimum time which the sensor readings do not change, Here, smaller β means that the sensor readings change quickly over time (such as GPS and sounds), while larger β means that sensor readings change slowly over time (such as temperature and humidity). To obtain the value of β for a sensor x in a specific AoI, the cloud server will apply the time series data differencing method to compute the smallest difference that have a small value ϵ, then it will compute the time of the smallest difference which is called the sensory data stability time β. By using the time series data differencing method, the system can obtain the minimum and maximum data differencing value for the dataset [29].

C. Representing Time Instants of Tasks by Intervals Tree

An interval tree is an ordered data structure whose nodes represent the intervals and are therefore characterized by a start value and an end value. A typical application example is when there is a number of available intervals and another set of query intervals, for which verifying the overlap with the given intervals is required.

The interval tree structure comes into play to provide a more efficient solution than the naive approach of applying a brutal force strategy and compare each query range with all the others and check if, according to the values of the relative bounds, there is an overlap (total or partial) between them. Fig.2 shows the intervals and the overlap between them.

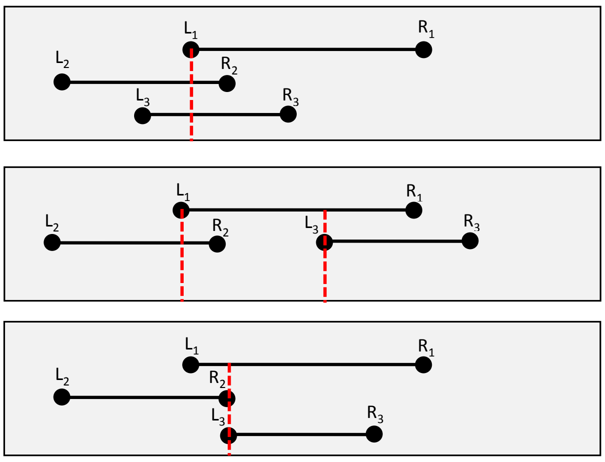

Fig. 2: Overlapped intervals for three classes of tasks

For each class of tasks on the cloud server it will be stored in an interval tree. This is an augmentation of the Binary Search Tree (BST) data structure. An interval tree has a leaf node for every elementary interval. On the top of these leaves, a complete binary tree is built. Each internal node of the tree stores, as its key, the integer that separates the elementary intervals in its left and right subtrees. The leaf nodes do not store any keys, and represent the elementary intervals. So, by using the intervals tree, the cloud server can store all the tasks time instants in the intervals tree which will contain all the overlapped intervals for all the tasks in the current class. The overlapped times will be used next to define a new time instant for all overlapped tasks. The interval tree running time cost for a query time is $O(K + logn)$, the preprocessing time is $O(nlogn)$, and the space is $O(n)$ where n is the number of intervals in the collection and K is the running time.

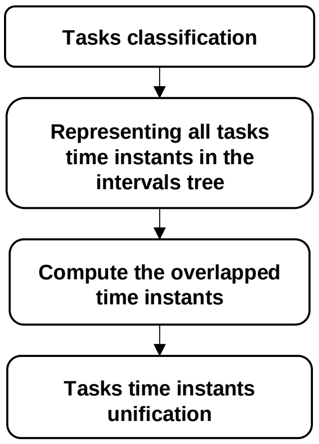
Tasks classification
Representing all tasks time instants in the intervals tree
Compute the overlapped time instants
Tasks time instants unification

Fig. 3: ITBTS phases

D. The proposed method

The proposed task scheduling method, ITBTS, consists of three phases, Fig.3:

1) Task classification phase, in this phase the cloud server classifies each task based on the required sensors.
2) Overlapping time computation phase, in this phase the cloud server finds the overlapped time instants of a task by using the intervals tree.
3) Unify the tasks time instants computation phase by finding the overlapped time and compute the new time instant of a task. These three phases are described as follows.

1) Task Classification Phase: In the proposed task scheduling ITBTS, all tasks will be scheduled before task assignment. So, in this phase, when a task organizer requests collecting data from a specific AoI, the cloud server will classify the received task and add it to a certain task class based on its required sensors. All tasks in the task class k must have a common sensor. For example, task i requests to collect data (pictures with locations) about an event from 1pm to 8pm, the participants should capture a picture every 10 minutes, and task j requests to collect data in the same location from 3pm to 10pm, the required data is sound records every 5 minutes or (10 times per hour) with its location and light level. So, the platform will try to fit the two tasks to work together and reduce the GPS usage. When a new task arrived on the platform, it will be added to its class. For each task class, the participant can select to perform partial or full class based on his available resources

2) Overlapping Time Instants Computation Phase: In this phase, when the task classification completed, the cloud server will start to find the overlapped times for all tasks. Each task has a number of time instants, for each time instant the cloud server will convert the time instant to span subinterval. The span subinterval will have a start and an end time, the start time is the current time instant - β and the end time is the current time instant + β, as shown in Fig.4. After obtaining all the spans subintervals of all tasks, the cloud server will build their intervals tree. The intervals tree will contain the information about the overlapped times of all tasks in the current task class.

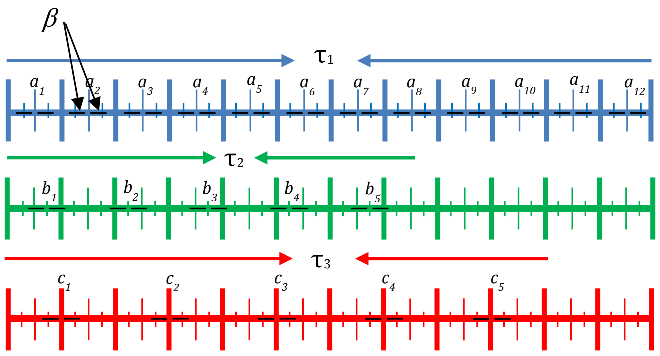

Fig. 4: The time instants for 3 MCS tasks

For each time instant of the tasks, if there is an overlapped between more than one task, then the cloud server will obtain the new time instant for the overlapped tasks. The overlapped will be on the right or left side of the current time instant for the overlapped tasks, if there is no overlapping between all tasks in the class, the cloud server will search for an overlap between a number of tasks such that $o \approxeq O$, where O is the number of tasks in the current class, and o is the number of tasks in the overlap. Fig.5 shows the overlapping between four intervals.

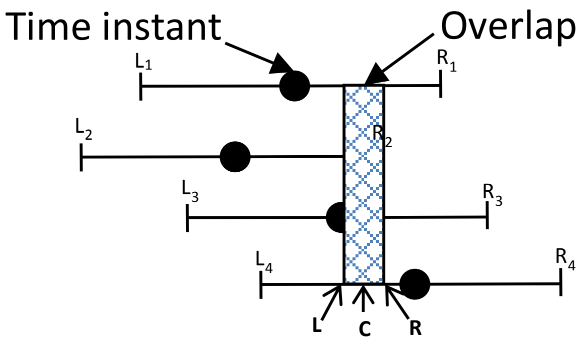

Fig. 5: Overlapped intervals

3) Common Time instants Computation Phase: In this phase, to compute the common time instant, ρ_j, for the overlapped tasks, the cloud server will compute the three distances D_l, D_r, and D_c which represent the distance between all time instants of a task and the left, right, and center of the overlapped area such that.

$$D_l = \sum_{i=1}^{o} dl_i \tag{7}$$

$$D_r = \sum_{i=1}^{o} dr_i \tag{8}$$

$$D_c = \sum_{i=1}^{o} dc_i \tag{9}$$

where D_l is the distance between the tasks time instants and the left border of the overlapped area, D_r is the distance between the tasks time instants and the right border of the overlapped area, and D_c is the distance between the tasks time instants and the center of the overlapped area. The common time instant ρ_j will be obtained by selecting the smallest distance from $D_l, D_r,$ and D_c, as follows

$$\rho_j = \begin{cases} L & \text{if } D_l \leq D_c \text{ \& } D_l \leq D_r \\ C & \text{if } D_c \leq D_l \text{ \& } D_c \leq D_r \\ R & \text{if } D_r \leq D_l \text{ \& } D_r \leq D_c \end{cases} \tag{10}$$

where $L, C,$ and R are the left, center and right of the overlapped intervals.

The previous steps will be performed for each time instant in the current task class. When computing all the common time instants, then the task class is ready to be assigned to the participants in the AoI. The participant has the choice to perform all the tasks in the class or select some tasks from the class. For each time instant that has an overlap, the MCS application will store the sensor reading for all the tasks in the current time instant which will minimize the sensor utilization and minimize the energy consumption. Alg.1, shows the steps of the proposed task scheduling method, ITBTS.

Algorithm 1: TASK SCHEDULING METHOD ITBTS

```
   Input: Number of tasks $N$
   Output: a uniform time line for each class of tasks
1  $UL \leftarrow \phi$ ;
2  $D = 0$ ;
3  $\Theta = 0$ ;
4  add the task to class according to the requested data ;
5  K = number of classes ;
6  for $v \leftarrow 0$ to $K$ do
7      create an interval for each time instant ;
8      insert all intervals to the intervals tree ;
9      foreach Time instant do
10         obtain the overlapped tasks ;
11         compute $D_l, D_r,$ and $D_c$ ;
12         if $D_l \leq D_c$ & $D_l \leq D_r$ then
13             $\rho_j = L$ ;
14         else if $D_c \leq D_l$ & $D_c \leq D_r$ then
15             $\rho_j = C$;
16         else
17             $\rho_j = R$ ;
18         add $\rho_j$ to the time line $TL$
19 return $TL$ ;
```

V. PERFORMANCE EVALUATION

In this section, the simulation results and the performance of the proposed ITBTS will be introduced and discussed. In this simulation, the OMNET++ simulator [30] have been simulating the proposed method. A discipline algorithm is used as the baseline for performance comparison with both the proposed task scheduling method ITBTS and the Minimum Energy Single-sensor task Scheduling (MESS) method in [27]. With the discipline scheduling algorithm, a task is executed at the same time instants which are requested by the task organizer. Here, many metrics are used for performance evaluation as the energy consumption, bandwidth, average shift time, sensor readings and tasks consistency. Table-I, shows the simulation parameters. In the rest of this section, firstly, the full description of a real scenario will be presented. Secondly, the performance metrics are introduced. Finally, the scenario results will be presented and discussed.

TABLE I: SIMULATION PARAMETERS

Parameter	Value
# of Tasks	5-50
Sensors	Temperature, Humidity and light
Sensors readings rate	5, 10, 15, 20 minutes
Wifi power consumption	545.07 mJ [31]
Temperature power consumption	50 mJ
Humidity power consumption	50 mJ
light power consumption	15 mJ
4G communication 2KB	1500 mJ [31]

A. Performance Metrics

Here, to evaluate the proposed ITBTS, the following metrics are used.

1) *Energy consumption*: which represents the consumed energy by the participants when performing a task class.
2) *Sensor readings*: which is the number of sensor readings actually performed for a class of tasks.
3) *Consistency tasks*: which is the percentage of time instants that have overlapping interval.
4) *Bandwidth*: which is the data rate that is needed to upload the collected sensors' readings.
5) *Average shift time*: which represents the difference between the unified time instants and the requested time instants.

B. Results and Discussion

The proposed method ITBTS lack to obtain the data stability value β to schedule the MCS tasks. The value of β will be computed by applying the data differencing equation to the data set in [29]. By using this method, we have obtained the minimum changing time equal to 20 minutes and the maximum changing time equal 115 minutes. So, the cloud server will use the minimum changing time $\beta = 20$ minutes to schedule the MCS tasks. The value of β will be used to add left and right the tasks time instant to allow the platform to choose a new time instant which its value will be equal to the value of the original time instant, which allows scheduling more than one time instant and performs them by only one sensor reading.

The experimental results for the consumed energy by the three different MCS task scheduling methods are shown in Fig.6 and Fig.7. As shown in Fig.6, the consumed energy by the proposed ITBTS is less than both the Baseline and

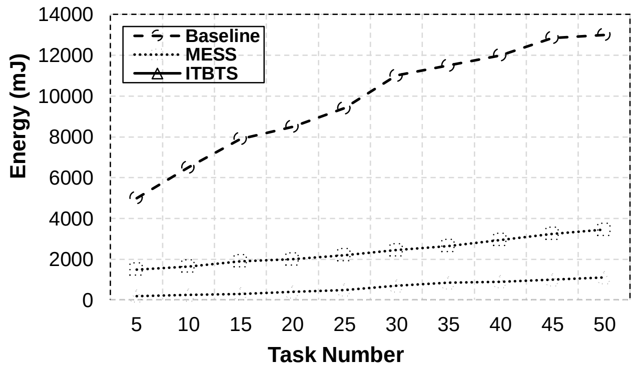

Fig. 6: The consumed energy by different # of tasks

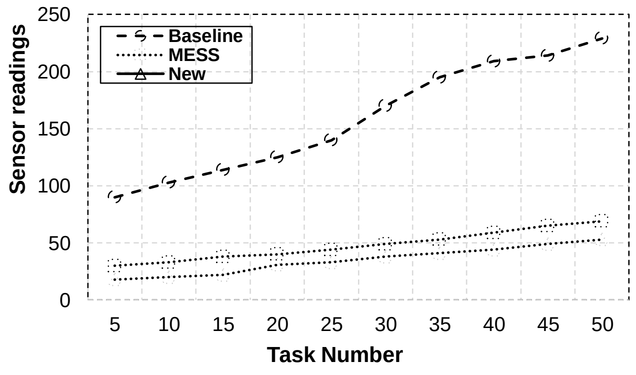

Fig. 8: # of sensor readings performed by different # of tasks

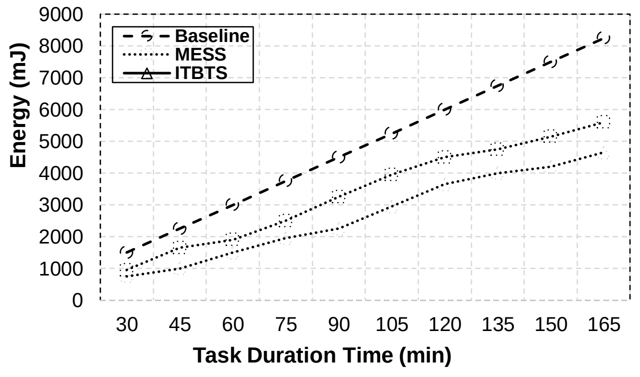

Fig. 7: The consumed energy by tasks with different duration time

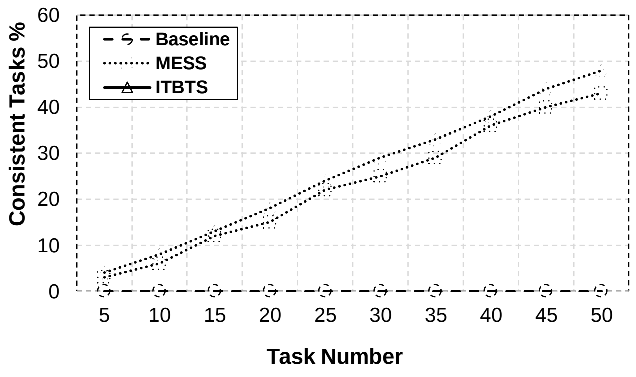

Fig. 9: The % of consistent tasks when changing the # of tasks

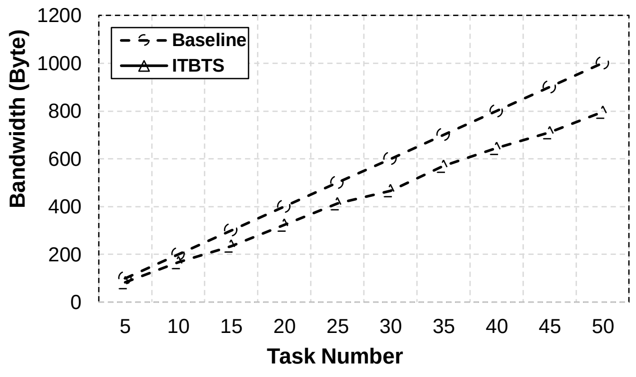

Fig. 10: The consumed bandwidth for uploading the collected data with different # of tasks

MESS methods, which is proved that ITBTS can minimize sensor utilization. The minimization comes from detecting the overlapped time between the tasks time instants after adding the obtained sensor reading stability value β, creating a uniform time instants line for all the tasks in the same class, and performing sensing at the new time instants. Whenever the sensor reads a new value, the MCS application assign this value to all overlapped tasks, which saves the energy of mobile units with the participants. Fig.7 shows the consumed energy with different task duration per minutes. As shown in Fig.7, the consumed energy by the proposed ITBTS is less than both the Baseline and MESS methods, which proves that ITBTS is not influence the changes of the task duration and it still can minimize the sensor usage.

Fig.8 and Fig.9, show the sensor readings and the consistency tasks against the number of tasks. As shown in Fig.8, the result indicates that the proposed ITBTS can minimize the sensor utilization by detecting the overlapped time between different tasks, replace the time instants for all overlapped tasks by a one time instant, and retrieve the sensor reading at this time, then it will assign the sensor reading to all overlapped tasks. As shown in Fig.9, the percent of task consistency between different tasks for the proposed ITBTS is higher than the MESS and the baseline methods. This is because the proposed ITBTS uses the time instants unification of the overlapped tasks which will preserve the participant device resources such as energy, bandwidth, and CPU.

Fig.10 and Fig.11, show the consumed bandwidth and energy from uploading the collected data to the cloud server. As shown in these figures, the consumption of both bandwidth and energy decreases as the number of tasks increases. This is because the proposed ITBTS uses the time instants unification of the overlapped tasks. As a result, the proposed ITBTS can minimize the participant device resources such as energy, bandwidth, and CPU.

Fig.12 shows the average time instants shift time against the number of tasks. As shown in Fig.8, the difference in the proposed ITBTS is less than the difference obtained by the MESS method. This confirms that the proposed ITBTS

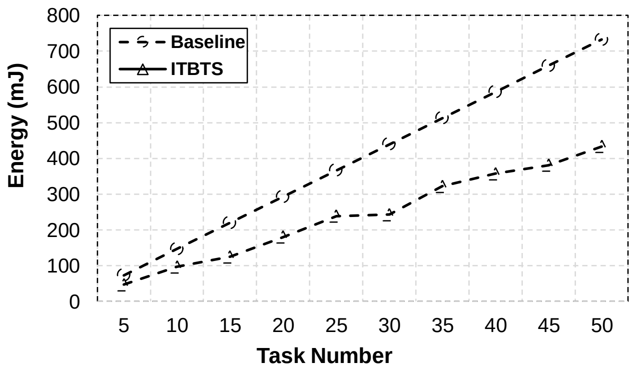

Fig. 11: The consumed energy from uploading the collected data with different # of tasks

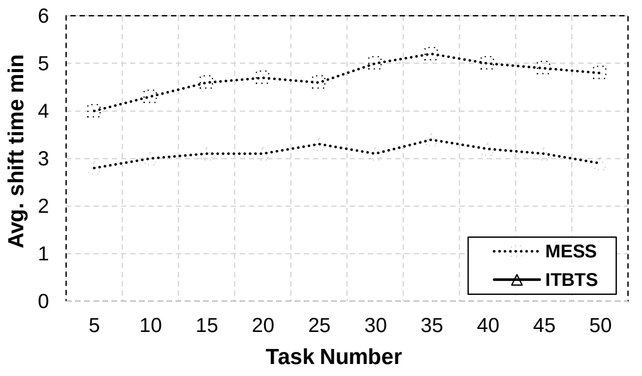

Fig. 12: The average time instants shift time in different # of tasks

can collect more accurate and reliable data than the MESS method, this is because the obtained time instants are close as possible to the requested time instants. Also, the accuracy of the obtained time instants is based on the value of β which is computed mathematically from the previous sensor reading in the same AoI.

VI. Conclusions

This paper addresses the problem of task scheduling problem for MCS systems. To solve this problem a novel MCS task scheduling method called interval tree-based task scheduling method, ITBTS was proposed. ITBTS is based on attracting the participants in the Area of Interest by scheduling the tasks that have a common sensor to minimize the total users resources consumption when they perform a class of tasks. ITBTS uses the data differencing in the time series analysis and interval tree to compute each sensor reading changing time and then uses this value to find a new time instant for the overlapped tasks. In addition, ITBTS classifies all the tasks according to their requested sensors and the intervals tree has been used to represent the time instants for each task class. In this tree, the overlapped tasks time instants will change to an appropriate common time instant for all tasks. The results of conducted simulation indicate that the proposed task scheduling method preserves the participants devices energy compared with the MESS and baseline methods.

In future works, the proposed task scheduling method will be developed to be adaptable with Multi-Sensor Tasks.

References

[1] B. Guo, Z. Yu, X. Zhou, and D. Zhang. From participatory sensing to mobile crowd sensing. In *2014 IEEE International Conference on Pervasive Computing and Communication Workshops (PERCOM WORKSHOPS)*, pages 593–598, Budapest, Hungary, March 2014.

[2] S. Greengard. *The Internet of Things*. The MIT Press Essential Knowledge series, Cambridge, Massachusetts, USA, 2015.

[3] J. Radianti, J. Dugdale, J. J. Gonzalez, and O. Granmo. Smartphone sensing platform for emergency management. In *ISCRAM 2014 Conference Proceedings 11th International Conference on Information Systems for Crisis Response and Management*, pages 379–383, Pennsylvania, USA, May 2014.

[4] E. Aubry, T. Silverston, A. Lahmadi, and O. Festor. Crowdout: A mobile crowdsourcing service for road safety in digital cities. In *2014 IEEE International Conference on Pervasive Computing and Communication Workshops (PERCOM WORKSHOPS)*, pages 86–91, Budapest, Hungary, March 2014.

[5] A. Longo, M. Zappatore, and M. A. Bochicchio. Collaborative learning from mobile crowd sensing: A case study in electromagnetic monitoring. In *2015 IEEE Global Engineering Education Conference (EDUCON)*, pages 742–750, Tallinn, Estonia, March 2015.

[6] N. D. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, and A. T. Campbell. A survey of mobile phone sensing. *IEEE Communications Magazine*, 48(9):140–150, 2010.

[7] M. Zappatore, A. Longo, and M. A. Bochicchio. Crowd-sensing our smart cities: a platform for noise monitoring and acoustic urban planning. *Journal of Communications Software and Systems*, 13(2):53–67, 2017.

[8] Shaohan Hu, Lu Su, Hengchang Liu, Hongyan Wang, and Tarek F. Abdelzaher. Smartroad: Smartphone-based crowd sensing for traffic regulator detection and identification. *ACM Trans. Sen. Netw.*, 11(4):1–27, 2015.

[9] D. C. Montgomery, C. L. Jennings, and M. Kulahci. *Introduction to Time Series Analysis and Forecasting*. John Wiley & Sons, 2015.

[10] J. Goncalves, S. Hosio, J. Rogstadius, E. Karapanos, and V. Kostakos. Motivating participation and improving quality of contribution in ubiquitous crowd sourcing. *Computer Networks*, 90(1):34 – 48, 2015.

[11] L. Pournajaf, D. A. Garcia-Ulloa, L. Xiong, and V. Sunderam. Participant privacy in mobile crowd sensing task management: A survey of methods and challenges. *ACM SIGMOD Record*, 44(4):23–34, 2016.

[12] R. I. Ogie. Adopting incentive mechanisms for large-scale participation in mobile crowdsensing: from literature review to a conceptual framework. *Human-centric Computing and Information Sciences*, 6(1):1–24, 2016.

[13] J. P. Rula and F. E. Bustamante. Crowdsensing under (soft) control. In *2015 IEEE Conference on Computer Communications (INFOCOM)*, pages 2236–2244, Kowloon, Hong Kong, April 2015.

[14] G. Yang, S. He, Z. Shi, and J. Chen. Promoting cooperation by the social incentive mechanism in mobile crowdsensing. *IEEE Communications Magazine*, 55(3):86–92, 2017.

[15] B. Guo, H. Chen, Z. Yu, W. Nan, X. Xie, D. Zhang, and X. Zhou. Taskme: Toward a dynamic and quality-enhanced incentive mechanism for mobile crowd sensing. *International Journal of Human-Computer Studies*, 102(1):14–26, 2017.

[16] P. Ma and D. Tao. 5w1h model for incentive mechanism in mobile crowd sensing. In *2016 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-TW)*, pages 1–2, May 2016.

[17] Y. Wang, Z. Cai, G. Yin, Y. Gao, X. Tong, and G. Wu. An incentive mechanism with privacy protection in mobile crowdsourcing systems. *Computer Networks*, 102(1):157 – 171, 2016.

[18] X. Chen, M. Liu, Y. Zhou, Z. Li, S. Chen, and X. He. A truthful incentive mechanism for online recruitment in mobile crowd sensing system. *Sensors*, 17(1), 2017.

[19] L. G. Jaimes, I. Vergara-Laurens, and M. A. Labrador. A location-based incentive mechanism for participatory sensing systems with budget constraints. In *2012 IEEE International Conference on Pervasive Computing and Communications*, pages 103–108, Lugano, Switzerland, March 2012.

[20] Z. Feng, Y. Zhu, Q. Zhang, L. M. Ni, and A. V. Vasilakos. Trac: Truthful auction for location-aware collaborative sensing in mobile crowdsourcing. In *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*, pages 1231–1239, April 2014.

[21] L. Gao, F. Hou, and J. Huang. Providing long-term participation incentive in participatory sensing. In *2015 IEEE Conference on Computer Communications (INFOCOM)*, pages 2803–2811, Kowloon, Hong Kong, April 2015.
[22] H. Jin, L. Su, D. Chen, K. Nahrstedt, and J. Xu. Quality of information aware incentive mechanisms for mobile crowd sensing systems. In *Proceedings of the 16th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, MobiHoc '15, pages 167–176, Hangzhou, China, 2015. ACM.
[23] D. Peng, F. Wu, and G. Chen. Pay as how well you do: A quality based incentive mechanism for crowdsensing. In *Proceedings of the 16th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, MobiHoc '15, pages 177–186, Hangzhou, China, 2015. ACM.
[24] R. Kawajiri, M. Shimosaka, and H. Kashima. Steered crowdsensing: Incentive design towards quality-oriented place-centric crowdsensing. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '14, pages 691–701, Seattle, Washington, 2014. ACM.
[25] S. Reddy, D. Estrin, M. Hansen, and M. Srivastava. Examining micro-payments for participatory sensing data collections. In *the 12th ACM International Conference on Ubiquitous Computing*, ACM, 2010, pages 33–36, Copenhagen, Denmark, 2010.
[26] J. S. Lee and B. Hoh. Dynamic pricing incentive for participatory sensing. *Pervasive Mobile Computing*, 6(6):693–708, 2010.
[27] J. Wang, J. Tang, G. Xue, and D. Yang. Towards energy-efficient task scheduling on smartphones in mobile crowd sensing systems. *Computer Networks*, 115(4):100 – 109, 2017.
[28] P. J. Brockwell and R. A. Davis. *Introduction to Time Series and Forecasting*. springer, 2016.
[29] M. A. Alswailim, H. S. Hassanein, and M. Zulkernine. CRAWDAD dataset queensu/crowd_temperature (v. 2015-11-20): derived from roma/taxi (v. 2014-07-17). Downloaded from http://crawdad.org/queensu/crowd_temperature/20151120, November 2015.
[30] A. Varga and R. Hornig. An overview of the omnet++ simulation environment. In *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops*, Simutools '08, pages 1–10, Marseille, France, March 2008.
[31] K. Lin, A. Kansal, D. Lymberopoulos, and F. Zhao. Energy-accuracy trade-off for continuous mobile device location. In *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*, MobiSys '10, pages 285–298, San Francisco, California, USA, 2010. ACM.

Ahmed A. A. Gad-ElRab is an Associate Professor of ubiquitous and mobile computing at the Department of Mathematics, Faculty of Science, Al-Azhar University,Cairo, Egypt. He received B.S. Degree in Computer Science, from Faculty of Science, Alexandria University, Egypt in 1999. He received MS. Degree in Computer Science from Faculty of Science, Cairo University, Egypt in 2008. He received his Ph.D. at the Nara Institute of Science and Technology (NAIST), a national corporation university located in NARA, Japan in 2012. He received the NAIST best PhD student award in March 2012, he received the outperformance Award from Graduate School of Information Science, NAIST in March 2012, and he received 2011 IPSJ Yamashita Memorial Award (given to only one or two papers among all papers presented in one year in each IPSJ SIG, Japan. His research interests include cloud computing, mobile computing, Internet of Things applications, smart home, data science, sensor networks, dynamic distributed systems, and mobile crowd sensing.

Almohammady S. Alsharkawy received the Masters degree in the context awareness computing from the Department of Mathematics, Faculty of Science, Al-Azhar University - Cairo, Egypt, in 2015. He is active in research, mobile computing, context awareness, IoT, and mobile crowd sensing. In 2015 Almohammady has started to study at the Department of Mathematics, Faculty of Science, Al-Azhar University - Cairo, Egypt PhD degree with topic of mobile crowd sensing management.