Task Allocation Schemes for Crowdsourcing in Opportunistic Mobile Social Networks

Xiao Chen[1], Bo Deng[2]
[1]Department of Computer Science, Texas State University, San Marcos, TX 78666
[2] Vandegrift High School, Austin, TX 78750
Email: xc10@txstate.edu, boyang.deng@gmail.com

***Abstract*—With the explosive proliferation of mobile devices, *mobile crowdsourcing* has become a new paradigm involving a crowd of mobile users to collectively take large-scale tasks required by requesters in mobile social networks (MSNs). In this paper, we work on new task allocation schemes for mobile crowdsourcing in Opportunistic Mobile Social Networks (OMSNs) which are formed opportunistically when people gather together. Our goal is to distribute the tasks to the minimum number of users using the minimum number of forwarders quickly. Our task allocation problem is related to the set cover problem which is NP-hard. To solve it, we put forward a heuristic *opportunistic task allocation approximation* (OTAA) algorithm where we use a consumer control criterion to reduce the number of consumers and a forwarder control criterion to select the best forwarder. After the analysis of two typical OMSN traces, we find it better to delegate all the tasks to the selected forwarder to reduce latency. Simulation results comparing our proposed algorithm with its variations using the two OMSN traces confirm that our proposed algorithm exhibits a high success rate, lowest latency, and uses small numbers of consumers and forwarders.**

***Index Terms*—crowdsourcing, opportunistic mobile social networks, task allocation**

I. Introduction

With the explosive proliferation of smartphones and tablets, *mobile crowdsourcing* has become a new paradigm involving a crowd of mobile users to collectively take large-scale tasks and provide sensing services required by requesters in mobile social networks (MSNs) [9]. Among the many topics in mobile crowdsourcing, task allocation or user recruitment [11], [18] is one of the most important ones. In this paper, we focus on task allocation in a special kind of mobile social network which is formed opportunistically when people gather together at conferences, social events, campus activities, etc. We refer to it as Opportunistic Mobile Social Network (OMSN).

Task allocation and user recruitment are similar. But user recruitment schemes usually consider providing incentives to the users. Here we assume that the users are motivated to take some tasks because of the payment or self-interest and thus our focus is on how to allocate the tasks to the users efficiently. Despite many task allocation or user recruitment algorithms for mobile crowdsourcing in the literature [6], [10], [11], [14], [16], [18], we are motivated to propose new ones due to the following thoughts. As we know, the majority of the current crowdsourcing implementation relies on centralized registries to recruit possible participants [13] and assumes the use of cellular networks to distribute tasks and collect data [11]. However, the soon-to-be Zettabytes of annual global IP traffic generated mostly from the mobile devices [2] has placed intense pressure on the existing cellular infrastructure [4]. Thus, researchers are seeking new paradigms to offload cellular traffic to opportunistic Device-to-Device (D2D) communication networks [3] that enable direct communication between mobile devices capable of short range communication [5] without traversing the cellular network.

To address the aforementioned issue, it is critical to devise new task allocation schemes that are fully distributed in the OMSNs. To be more specific, we will focus on the *opportunistic task allocation* (OTA) problem to distribute W tasks, i.e. sensing data or others, over the opportunistic contacts among users subject to the constraints of reducing the cost of the requester and the total latency. We assume that initially a source node s has some tasks given by the requester. We call a node with tasks a *task carrier*. Then s will allocate these tasks to the users who can perform these tasks. We assume each user u_i in the network can perform a certain number of tasks. If a user is selected to fulfil the tasks, we call him a *consumer*. Minimizing the number of consumers on a task can reduce the cost and overhead of the task requester. In addition, in the opportunistic networks, a node can only distribute tasks to another node when they enter each other's transmission range. So reducing the total task distribution latency is also important because tasks may have deadlines. To reduce latency, it is a good idea to use some active nodes that meet other nodes frequently as *forwarders*. Similarly, recruiting a small number of forwarders can reduce the cost and overhead of the task requester and the forwarders themselves. Therefore, our goal in the OTA problem is to distribute the tasks using minimum numbers of consumers and forwarders quickly in the OMSNs.

Our proposed OTA problem is related to the set cover problem [12] which is NP-hard. What makes our problem different and difficult are the inclusion of the time dimension and the unpredictable contacts of nodes in the OMSNs. Since our problem is NP-hard, we propose a heuristic *opportunistic task allocation approximation* (OTAA) algorithm to solve it. As there is no global information, the key to solving the problem relies on the wise decision of a task carrier x when it meets a node y opportunistically. The decision follows a *consumer control criterion* to reduce the number of consumers and a *forwarder control criterion* to control the number of forwarders and reduce latency. From the analysis of two typical OMSN traces recording node contacts at a conference and on a campus, we find it more efficient

to delegate all the tasks to one node due to the fact that many nodes meet several nodes at the same time or in a row.

The differences of our work from others and the key contributions of our work are as follows:

- We define an NP-hard task allocation problem in a distributed and opportunistic environment and propose a heuristic scheme to solve it.
- We conduct simulations to compare our proposed algorithm and its variations using two typical OMSN traces. Simulation results confirm that our proposed algorithm exhibits a high success rate, lowest latency, and uses small numbers of consumers and forwarders.

The rest of the paper is organized as follows: Section II references the related works; Section III provides preliminary information; Section IV presents our proposed algorithm; Section V describes the simulations comparing the algorithm with its variations; and the conclusion is in Section VI.

II. Related Works

There has been a lot of research on user recruitment or task allocation problems in mobile crowdsourcing [6], [10], [11], [14]–[16], [18]. The various user recruitment or task allocation algorithms proposed by the majority of these works assume a centralized registry and use the cellular network resources for communication. For example, M. Cheung et al. in [6] consider a mobile crowdsourcing platform that posts task information and propose an algorithm to help the users plan their task selections on their own. Z. He et al. in [10] propose a participant recruitment strategy for vehicle-based crowdsourcing based on predicted vehicle trajectory. S. He et al. in [14] design a local ratio based algorithm to allocate location dependent tasks by considering the spatial movement constraints of mobile users. M. Xiao et al. in [16] and Q. Zhao et al. in [18] study the task allocation issues by formulating them as online scheduling problems.

Not many papers in the literature discuss distributed task allocation over opportunistic networks. We are aware of two papers under this thread. M. Karaliopoulos et al. [11] look into the problem of user recruitment over opportunistic networks. But their focus is transferring collected data over the opportunistic network to the campaign organizer, not how the tasks are distributed to the users in the first place. G. S. Tuncay et al. [15] consider both user recruitment and data collection using opportunistic networks. But their goal is to find users to cover the sensing areas, not how to distribute the tasks to the minimum consumers quickly as in our work.

III. Preliminary

A. Problem Definition

We consider mobile crowdsourcing in OMSNs, where a requester wants to opportunistically distribute W tasks to a number of users $U = \{u_1, u_2, \cdots, u_n\}$ as soon as possible. Each user u_i in the network can consume some tasks $c(u_i), c(u_i) \geq 0$. We assume that a particular user s takes over these W tasks from the requester first. Since s may not be able to consume all of the tasks, it needs to distribute them to other nodes. A task carrier such as s can only pass on the tasks to another node when they encounter. We assume that each node has a transmission range of distance $R(R \geq 0)$ and two nodes encounter if they are within each other's transmission range. All the nodes in the network can be divided into three categories: the *consumer* that performs the tasks, the *forwarder* that helps forward the tasks and can be a consumer at the same time, and *others* that are neither a consumer nor a forwarder. The goal of the problem is to minimize the number of consumers and forwarders to reduce the cost of the requester and distribute the tasks as soon as possible.

Section	Percentage of nodes	Frequency (times)
1	100%	31
2	75%	20
3	77%	12
4	67%	6

TABLE I
The percentage and frequency that a node meets others at least five times in a row in the Infocom trace

B. Problem Hardness

Theorem 1. *The OTA problem is NP-hard.*

Proof sketch: This problem is related to the set cover problem [12], one of the Karp's 21 NP-hard problems. It is to identify the smallest sub-collection of a collection whose union equals the collection. Since our problem is to identify the smallest number of consumers who each can perform a sub-set of the total tasks, it is also NP-hard. Furthermore, what makes our problem different and difficult are the time dimension and the unpredictable node contacts in the OMSNs. That is, the consumers are identified at different times through opportunistic node encounters. □

C. Features of Opportunistic Mobile Social Networks

To facilitate our algorithm design, we explore the features of OMSNs by studying the following two typical OMSN traces from the Crawdad website [1] archiving wireless network data for the research community.

1) The Infocom trace (v. 5/29/2009): This trace has been widely used to test MSN routing algorithms [7], [17]. It records Infocom attenders' encounter history using Bluetooth devices for 4 days at the conference.

2) The upb/hyccups trace (v. 10/17/2016): This trace gathers information of device encounters at the University Politehnica of Bucharest for 63 days using an application called HYCCUPS Tracer.

To study the properties of the contacts, in the Infocom trace, we divided the whole trace into four sections. In each section, we calculated the percentage of nodes that meet other nodes at least five times in a row and the number of times that this occurs. The results are shown in Table I. We can see that in the first part of the conference, all nodes meet other nodes at least five times in a row and that happens on an average of 31 times in that section. In the next three sections, these two numbers become smaller, which match the

real situation of a conference. Usually at the beginning of the conference, each participant meets a lot of people at the same time and then he meets fewer and fewer people concurrently in the next few days due to different schedules or early departure of some attendants. For the whole trace, on an average of 60 times, 100% of the people meet other people at least five times in a row and on an average of 12 times, 90% of the people meet other people at least eight times in a row. We see the similar pattern in the Hyccups trace. On an average of six times in the whole trace, 42% of the nodes meet other nodes at least five times in a row as a possible result of attending classes or participating in activities on campus. Thus, due to such high frequencies of nodes meeting many other nodes concurrently in the traces, in order to reduce latency, we conclude that it would be a good idea to just let the task carrier distribute the tasks rather than split the tasks to multiple nodes.

IV. Task Allocation Approximation Algorithm

Since the OTA problem is NP-hard, we propose a heuristic opportunistic task allocation approximation (OTAA) algorithm shown in Fig. 1 to solve it. It has the following phases.

A. Initialization

Initially the source node s has W tasks. The remaining tasks carried by s denoted by $r(s)$ is equal to W. Each node x has a *consumption capability* of $c(x)$ tasks known by itself and then known by other nodes when they encounter and exchange information. How active a node meeting other nodes is depicted by its *activity value* $a(x)$, which is the total number of contacts node x had with all the other nodes in the history we observe. Each node x also keeps an *activity threshold* $\tau(x)$ that records the highest node activity value x has seen so far. The activity threshold is used to control the number of forwarders. In the beginning, node x sets its activity threshold $\tau(x)$ to its activity value $a(x)$. As long as not all of the tasks are allocated, the distribution process continues. When a node x and a node y meet, they will carry out the following actions.

B. Exchanging and Updating Node Consumption List

Each node x in the network has a node consumption list $\{c_x(u_1), c_x(u_2), \cdots, c_x(u_N)\}$, recording the number of tasks node u_i can consume known to x denoted by $c_x(u_i)$. When x and y meet, they exchange and update each node's consumption capability to the minimum known to them. This is because after a node consumes some tasks, the number of tasks it can further consume will be reduced. The exchange and update is to propagate each node's newest remaining consumption capacity to the other nodes in the network to help them make decisions in the following task consumption phase. If node x is a task carrier, it goes to the next two phases.

C. Task Consumption

In this phase, a task carrier x will decide whether itself and y are eligible to be consumers or not. There are two conditions that a node can become a consumer. One is that it has the capability $c(x)$ to consume the remaining tasks $r(x)$ carried by x, i.e. $r(x) \leq c(x)$. And the second is the criterion to control the number of consumers. That is, node x should select nodes that have high consumption capability. To identify such a node, node x compares it with the C most *capable nodes* it has met so far. We denote this set of the

Algorithm OTAA: Opportunistic Task Allocation Approximation Algorithm

Require: Each node x consumes $c(x)$ tasks; The source node s has W tasks: $r(s) = W$;
1: /* initialize each node x's activity threshold to its activity value*/
2: $\tau(x) = a(x)$;
3: **while** not all of the tasks are allocated **do**
4: /* On contact between a node x and a node y */
5: x and y update each element in the node consumption list to $c_x(u_i) = c_y(u_i) = min(c_x(u_i), c_y(u_i))$;
6: **if** x is a task carrier **then**
7: /* Tasks Consumption */
8: $D(x) = \{$the most capable friends selected by x$\}$;
9: **if** $r(x) <= c(x) || c(x) > c_{avg}(D(x))$ **then**
10: /* x consumes $\min(c(x), r(x))$ tasks */
11: $r(x) = r(x) - \min(c(x), r(x))$;
12: $c(x) = c(x) - \min(c(x), r(x))$;
13: /* update y's knowledge about x's consumption capability */
14: $c_y(x) = c(x)$;
15: **end if**
16: **if** $r(x) > 0$ **then**
17: **if** $r(x) <= c(y) || c(y) > c_{avg}(D(x))$ **then**
18: /* y consumes $\min(c(y), r(x))$ tasks */
19: $r(x) = r(x) - \min(c(y), r(x))$;
20: $c(y) = c(y) - \min(c(y), r(x))$;
21: /* update x's knowledge about y's consumption capability */
22: $c_x(y) = c(y)$;
23: **end if**
24: **end if**
25: /* Task reassignment */
26: **if** $r(x) > 0$ **then**
27: **if** $\tau(x) < a(y)$ **then**
28: r(y) = r(y) + r(x);
29: /* x improves its activity threshold to $a(y)$ */
30: $\tau(x) = a(y)$;
31: **if** $r(y) > 0$ **then**
32: it becomes a new task carrier x and repeats the process;
33: **end if**
34: **end if**
35: **end if**
36: **end if**
37: **end while**

Fig. 1. The opportunistic task allocation approximation (OTAA) algorithm

most capable nodes by $D(x)$. When x selects nodes for this set, it picks the first C nodes that consume most tasks and whose meeting frequencies with x are above the average meeting frequency of x and all of the other nodes. So the consumer control criterion is: **a node is eligible to be a consumer if its consumption capability is greater than the average consumption capability of the nodes in** $D(x)$. If node x and/or y become consumers, the remaining tasks $r(x)$ and x's and/or y's remaining consumption capability $c(x)$ and/or $c(y)$ are subtracted by the consumption number. Also x's consumption capacity $c(x)$ is updated in y's node consumption list to reflect y's knowledge about x's remaining consumption capacity and vice versa. If in any of these cases, $r(x)$ becomes zero after the deduction, the distribution task of x is done. But no matter whether x and/or y become consumers or not, as long as x is a task carrier, it goes to the next phase.

D. Task Reassignment

In this phase, node x will decide whether to make y a forwarder. In order to reduce latency, node x should choose an active node that meets other nodes frequently. To identify such a node, it uses the forwarder control criteron: **A node is eligible to be a forwarder if its activity value** $a(y)$ **is above** x**'s activity threshold** $\tau(x)$. A node x's activity threshold $\tau(x)$ records the most active node that x has seen so far. Initially, it is set to x's active value and after making y as a forwarder, x will improve its $\tau(x)$ to y's activity value $a(y)$. This condition is enlightened by the delegation forwarding in [8] to control the number of forwarders in the network so as to reduce the distribution cost. Also, based on the node contact pattern in the OMSNs, if y is selected to be a forwarder, it is a good idea for x to delegate all its tasks to y. After that, y becomes a new task carrier and the distribution process continues.

V. Simulations

A. Algorithms Compared

In this section, we evaluate the performance of our proposed scheme OTAA by comparing it with its variations using a custom simulator written in Matlab.

1) *The Opportunistic Task Allocation Approximation Algorithm* (OTAA): our proposed algorithm with a consumer control criterion and a forwarder control criterion and where a node carrier delegates all its tasks to a node with the highest activity value it has seen so far.
2) *The variation of OTAA* (Hold): a variation of OTAA with a consumer control criterion and where only the source node distributes the tasks. No forwarder control criterion is used.
3) *The variation of OTAA* (Split): a variation of OTAA with a consumer control criterion and a forwarder control criterion and where a node carrier splits the tasks evenly between itself and the node with the highest activity value it has seen so far.
4) *The variation of OTAA* (SplitAny): a variation of OTAA with a consumer control criterion and a forwarder control criterion and where a node carrier splits the tasks evenly between itself and any node it meets.
5) *The Wait Algorithm* (Wait): where only the source node distributes the tasks to whatever nodes it meets. No consumer and forwarder control criteria are used.

B. Simulation Setup

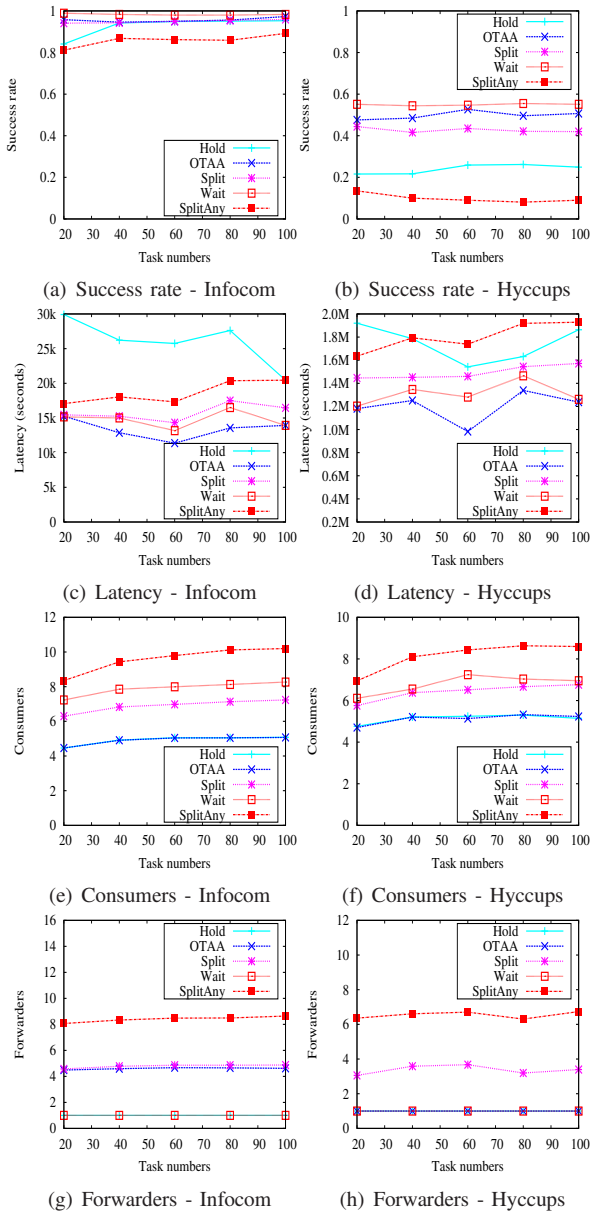
We used the following metrics to evaluate the performance of the algorithms.

- *Success rate*: the ratio of the number of successful task distributions and the total number of task distributions.
- *Consumers*: the number of consumers recruited to perform the tasks.
- *Forwarders*: the number of forwarders that help distribute the tasks.
- *Latency*: the total distribution time counted from the beginning to the end of the task distribution.

In our simulations, we divided each trace time into two intervals. The first part of the trace is used to obtain the initial value, e.g. activity value, of each node. The second part is used for node contacts in the simulation. In each experiment, we randomly generated a source node s, set the total number of tasks W to be 100, randomly generated each node's consumption capability uniformly in the range of $[0, W/4]$, and set the number of most capable nodes C a task carrier used in the consumer control criterion to 5. We ran each algorithm 1000 times and averaged the results.

C. Simulation Results

The simulation results using the Infocom trace are shown in Fig. 2 (a)(c)(e)(g) and the Hyccups trace in Fig. 2 (b)(d)(f)(h). In both traces, we can see that Wait has the highest allocation success rate and low latency (due to the node contact pattern in the traces), high number of consumers (because it does not use the consumer criterion to reduce the number of consumers), and the lowest number of forwarders (only the source itself). Same as Wait, Hold also just has the source as its forwarder. But different from Wait, Hold uses the consumer criterion and only chooses a node as a consumer if its consumption capability is above the average consumption capability of the most capable nodes known to the deciding node. So it uses fewer consumers than Wait. However, because of its selectiveness, it incurs larger latency and lower success rate. To reduce latency and improve the success rate, OTAA and Split pass all or half of the tasks to the most active node a task carrier has seen so far. Thus, both of them have lower latency and higher success rate than Hold because now the tasks are carried by the more active nodes. Comparing OTAA with Split, they have no difference in the number of consumers as demonstrated in the overlapped curves in Fig. 2 (e) and (f) because they use the same consumer selection criterion. But OTAA has fewer forwarders than Split because only one node rather than two carries the tasks in OTAA after the meeting of two nodes. Furthermore, OTAA has higher success rate and lower latency than



(a) Success rate - Infocom (b) Success rate - Hyccups

(c) Latency - Infocom (d) Latency - Hyccups

(e) Consumers - Infocom (f) Consumers - Hyccups

(g) Forwarders - Infocom (h) Forwarders - Hyccups

Fig. 2. Comparison of the algorithms using the Infocom trace in (a)(c)(e)(g) and Hyccups trace in (b)(d)(f)(h) when the number of tasks W is 100

Split because Split splits tasks but OTAA does not. This again is due to the node contact pattern in the traces. The worst scheme is SplitAny, which not only has the highest number of forwarders because it is not selective, the highest number of consumers and the lowest success rate because it separates tasks to too many nodes. Overall, after comparing these algorithms, the best algorithm that has high success rate, lowest latency, small numbers of consumers and forwarders is OTAA. Though using the consumer selection criterion in OTAA makes its success rate a little lower than that of Wait, the result of delegating all the tasks to the most active node known to the decider in OTAA is more significant to bring down the latency in addition to the reduction in the number of consumers compared with Wait.

In summary, we have the following conclusions. First, the consumer control criterion can bring down the number of consumers but it has lower success rate and higher latency due to its selectiveness of the consumers. Second, the forwarder control criterion can decrease the number of forwarders. Third, due to the contact pattern in situations like conferences and campus, it is a good idea to delegate the tasks to one node to increase the success rate and decrease the latency.

VI. Conclusion

In this paper, we have worked on the task allocation schemes for mobile crowdsourcing in OMSNs. Since the defined problem is NP-hard, we have put forward a heuristic OTAA algorithm where we have applied a consumer control criterion to reduce the number of consumers and a forwarder control criterion to select the best forwarder, and after the analysis of two typical OMSN traces, we have decided to delegate all the tasks to the selected forwarder to reduce latency. Simulation results comparing OTAA with its variations using the two OMSN traces have confirmed that our proposed algorithm has a high success rate, lowest latency, and uses small numbers of consumers and forwarders. In the future, we will explore scheme improvement by considering node social features.

References

[1] Crawdad. https://crawdad.org.

[2] The Zettabyte Era: Trends and Analysis. http://www.cisco.com/c/en/us/solutions.

[3] R. Alkurd, R. Shubair, and I. Abualhaol. Survey on device-to-device communications: Challenges and design issues. In *IEEE International New Circuits and Systems Conference*, 2014.

[4] A. Asadi, Q. Wang, and V. Mancuso. A survey on device-to-device communication in cellular networks. *IEEE Communication Surveys & Tutorials*, 16(4), 2014.

[5] A. Balasubramanian, R. Mahajan, A. Venkataramani, B. N. Levine, and J. Zahorjan. Interactive wifi connectivity for moving vehicles. In *ACM SIGCOMM*, 2008.

[6] M. H. Cheung, R. Southwell, F. Hou, and J. Huang. Distributed timesensitive task selection in mobile crowdsensing. In *ACM MobiHoc*, 2015.

[7] X. Deng, L. Chang, J. Tao, J. Pan, and J. Wang. Social profile-based multicast routing scheme for delay-tolerant networks. In *IEEE ICC*, 2013.

[8] V. Erramilli, M. Crovella, A. Chaintreau, and C. Diot. Delegation forwarding. In *the 9th ACM international symposium on Mobile ad hoc networking and computing*, pages 251–260, 2008.

[9] R. K. Ganti, F. Ye, and H. Lei. Mobile crowdsensing: Current state and future challenges. *IEEE Communications Magazine*, 49:32–39, 2011.

[10] Z. He, J. Cao, and X. Liu. High quality participant recruitment in vehicle-based crowdsourcing using predictable mobility. In *IEEE INFOCOM*, 2015.

[11] M. Karaliopoulos, O. Telelis, and I. Koutsopoulos. User recruitment for mobile crowdsensing over opportunistic networks. In *IEEE INFOCOM*, 2015.

[12] R. M. Karp. Reducibility Among Combinatorial Problems. *Complexity of Computer Computations*, pages 85–103, 1972.

[13] S. Reddy, D. Estrin, and M. Srivastava. Recruitment Framework for Participatory Sensing Data Collections. In *International Conference on Pervasive Computing*, 2010.

[14] J. Zhang S. He, D.-H. Shin and J. Chen. Toward optimal allocation of location dependent tasks in crowdsensing. In *IEEE INFOCOM*, 2014.

[15] G. S. Tuncay, G. Benincasa, and A. Helmy. Participant recruitment and data collection framework for opportunistic sensing: a comparative analysis. In *ACM MobiCom workshop on Challenged networks*, 2013.

[16] M. Xiao, J. Wu, L. Huang, Y. Wang, and C. Liu. Multi-task assignment for crowdsensing in mobile social networks. In *IEEE INFOCOM*, 2015.

[17] Y. Xu and X Chen. Social-similarity-based multicast algorithm in impromptu mobile social networks. In *IEEE Globecom*, 2014.

[18] Q. Zhao, Y. Zhu, H. Zhu, J. Cao, G. Xue, and B. Li. Fair energy-efficient sensing task allocation in participatory sensing with smartphones. In *IEEE INFOCOM*, 2014.