Correlated Friends' Impacts in Social-crowdsensing

Wei Chang
Department of Computer Science
Saint Joseph's University, USA
wchang@sju.edu

Wei-Shih Yang
Department of Mathematics
Temple University, USA
ws.yang@temple.edu

Jie Wu
Department of Computer and Information Sciences
Temple University, USA
jiewu@temple.edu

ABSTRACT

Social sensing is a typical application of the crowdsourcing system. With the consideration of system timeliness, flexibility, and stability, it could not be more natural to build a self-organized, distributed, and cross-platform crowdsourcing system. Social-crowdsensing (SC) is the first attempt. In SC, a huge sensing task is gradually partitioned into smaller pieces, and the pieces are propagated to potential workers via stochastic social contacts. During these contacts, allocating the workload is a critical problem, which affects the work's completion time and system resource utilization. By analyzing real data, we notice that the times of social contact occurrences are partially correlated. Whether it is necessary to purposely incorporate workers' correlation into the decision-making phase of workload allocation becomes an open question. In this paper, we systematically study the impacts of users' correlated behaviors.

CCS CONCEPTS

•**Networks** → **Mobile networks;** •**Mathematics of computing** → *Stochastic processes;* •**Theory of computation** → Probabilistic computation;

KEYWORDS

Correlation, potential resources, social-crowdsourcing, utilization, workload allocation.

ACM Reference format:
Wei Chang, Wei-Shih Yang, and Jie Wu. 2017. Correlated Friends' Impacts in Social-crowdsensing. In *Proceedings of The 2nd International Workshop on Social Sensing, Pittsburgh, PA USA, April 21 2017 (SocialSens'17),* 6 pages.
DOI: http://dx.doi.org/10.1145/3055601.3055605

1 INTRODUCTION

Social sensing is an emerging technique that collects participants' surrounding information, such as sensory readings or photos, via a crowdsourcing paradigm, where a task owner uploads a time-consuming sensing task onto a server [5] and volunteers to undertake a part of the task. However, there are three defects with these systems. First, they lack a timely advertising mechanism to recruit workers. Unless the owner provides a tempting payment, it is hard for a newly created task to attract enough workers in a short time; many off-line workers, who are eager to do certain types of tasks, are not aware of the existence of the new tasks. Second, the current systems are centralized and platform-specified: malfunctions or the unavailability of a platform will make all of its tasks fail. Finally, someone has to pay a fee for using them. For instance, Mturk collects a 10% commission on top of a task's total payment [1].

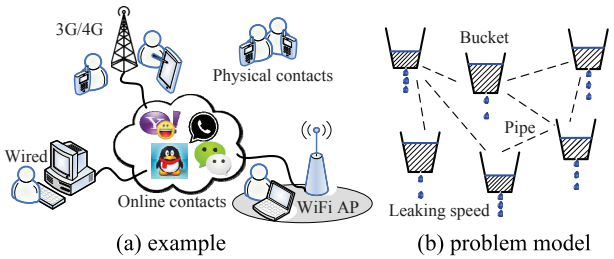


Figure 1: Workload allocation problem.

This paper first proposes a new self-organized distributed system, *Social-Crowdsensing* (SC). The main idea of SC is to create a multilayered outsourcing structure via any stochastic contacts, including physical encounters between friends and virtual contacts via any online-chatting system, such as Tencent QQ or WhatsApp. The general procedure for SC is as follows. A job owner first creates an SC task. Once it is done, the owner becomes the first worker and begins to locally conduct the task. Here, the job owner could be the original owner of the sensing task or a worker who undertakes a subtask from a conventional crowdsourcing platform, such as MTurk [2]. In SC, any worker can further recruit new workers via social contacts. For example, when a worker notices one of his friends coming online or when he physically encounters a friend, he will send a message to the friend and ask his willingness to join the task. If the answer is yes, a portion of the workload will be transferred from the worker to his friend, and then, both of them work in parallel. Since workers may not immediately check messages, always access the Internet, or stay together, their contact frequencies may be different. Essentially, subtasks are assigned to workers via *multi-hop* relays. Whenever a worker completes the assigned

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
SocialSens'17, Pittsburgh, PA USA
© 2017 ACM. 978-1-4503-4977-2/17/04...$15.00
DOI: http://dx.doi.org/10.1145/3055601.3055605

workload, he returns the sensing data to the job owner via the Internet or physical counters-based multihop relays.

Typical SC applicable scenarios involve the demand of timely collecting a set of reliable sensing data. For example, taking pictures is a special type of sensing task; after President Donald Trump signed the immigration executive order to restrict people from seven countries from entering the United States, journalists or body language experts may want to capture $1,000$ pictures of the immediate facial reactions of Muslim-Americans living in Hamtramck, Michigan when heard the news for the first time. Since news is propagated quickly and the task is about the first reactions of people in a certain area, the conventional crowdsourcing platform is not suitable for this type of task.

Since work segments are disseminated via multi-hop relays to participants, the estimation of workers' workloads becomes a critical problem, which not only affects a task's completion time, but also the system resource utilization (i.e. the amount of wasted human/sensing resources). An immediate question in SC is as follows: given that SC tasks initiate at random nodes, by what workload allocation strategy can the system utilization be maximized? Unlike any model used in crowdsourcing, workers in SC are *correlated*. For example, colleagues have similar working hours, and therefore, their times of unavailability for participating in the sensing tasks are not independent. Experimental results (Fig. 2) also indicate the existence of correlativeness. Whether it is necessary to incorporate workers' correlations into the decision-making phase of workload allocation becomes an open question.

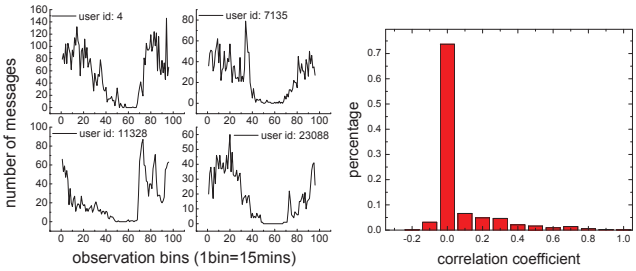
In this paper, we first propose a general model for the workload allocation problem in SC, and then, to investigate the impacts of the correlations, we provide an abstract model to describe the phenomena about workers' correlativeness. Two lightweight workload allocation algorithms are designed, with and without considering the correlations, respectively. By extensive simulations, we found a counterintuitive result showing that the awareness of users' correlation has a marginal contribution to the workload allocation problem in SC.

2 MODEL AND FORMULATION

2.1 System Model

We consider any type of social contact based on online-chatting systems or users' face-to-face contacts. Assume that an SC system consists of m users (workers), and some of them are friends, who may stochastically contact each other. Let v_i represent a user, and each user is associated with a constant working speed, s_i. For the ease of analysis, we assume that once a user participates in a task, he will not stop until the task has been finished. Note that, in reality, a user can quit at any time; before leaving, he should give the uncompleted works to the remaining workers.

SC involves multiple rounds of recruiting via social contacts: a user's workload will be outsourced to his friends, friends' friends, and so on. In SC, essentially, a time-consuming task is gradually partitioned into smaller pieces, and these pieces are propagated from friends to friends via



(a) user daily activity pattern (b) distribution of correlation

Figure 2: The correlation between social contacts

a multi-hop relay. A task consists of several work segments, and a segment is the smallest indivisible data unit. The workload of a worker is the total number of segments that have been assigned to him. Any user could be a task's owner, and multiple tasks may coexist.

In SC, workload exchanges happen during stochastic contacts between workers. When a worker physically comes across another one, the overloaded one can offload a certain amount of workload to the other one via a shortwave radio. As for the free WiFi users, they may not always stay in the same place, and some locations may not provide free networks. As a result, they may come on-line/off-line stochastically. Whenever they access the Internet, they can check and re-balance their workloads with their online friends (coworkers). Once in a while, cellular or wired network users may spontaneously query their friends' working progress and further adjust workloads. However, due to costs of network usage and workers' reaction times, the contacts among these workers are also intermittent, instead of ongoing. Clearly, the workers' contacting time is a random variable, and we use λ_{ij} to represent the average contacting frequency between v_i and v_j.

In practice, data transmission between workers can be implemented by using the auxiliary file sharing functions provided by any online chatting system or network storages, such as email or dropbox. Note that a worker's friends should not share one storage "box" due to the security reason. SC has a concept of commitment: at any time, a work segment only belongs to one worker. If we put work segments in one box and let the worker's friends fetch the data based on their dynamic needs, some workers may maliciously modify certain segments in order to prevent others from winning the bonus (the friends of a user may not be friends). In order to avoid using complex security mechanisms, SC does not allow multiple users to share network storage.

2.2 Problem Formulation and Challenges

SC is a self-organized distributed system. By following the recruiting procedures, several users can temporarily create an SC system by themselves. Although establishing SC is easy, determining the workload for each worker is not trivial: a workload allocation strategy directly influences tasks' completion times and system resource utilization. Here, we

define the resource utilization, U, as follows:

$$U = \frac{\sum_{i=1}^{m} s_i \int_0^{t^*} \delta_i(t)\, dt}{\sum_{i=1}^{m} s_i t^*} \quad (1)$$

where t^* is the time when a majority of the work has been completed, and $\delta_i(t)$ is a 0/1 function. $\delta_i(t) = 1$ if v_i is working on some SC tasks; otherwise, $\delta_i(t) = 0$. *Our objective is to determine a local workload allocation strategy to improve the overall utilization of a social-crowdsourcing system.*

We abstract an SC system as a stochastic bucket network, as shown in Fig. 1: each node is modeled as a bucket with unlimited volume. Initially, one (or a few) buckets have water (the original large task). Each bucket has a hole at the bottom, and its size is different, which reflects the workers' diverse working speed. Stochastically, we can pour some amount of water from one bucket to another, which models the workload allocation during contacts. We want to design a local algorithm to determine the amount of transmitted water for each contact. For the remainder of the paper, we will not differentiate how a social contact is conducted (physically or online), since all types of contacts can be abstracted as *stochastic* data exchanges between friends.

The optimal allocation [7] happens when (1) each worker participates in a task as early as possible and (2) all of them simultaneously complete their assigned work. However, due to the lack of global information and the accurate future encountering times, in practice, it is impossible to get an optimal result. We notice that the essential requirement for our objective is to quickly establish a balanced workload allocation within a stochastic system.

Inspired by the Pagerank algorithm, a percentage of the workload that a worker undertakes from a task is related to the worker's *overall* computing capacity, which is determined not only by his own working speed, but also the capacity from his friends, his friends' friends, and so on. In this paper, we allocate workloads according to it. However, locally estimating such an overall score is challenging: we should incorporate correlations, contacting uncertainties, and working speeds into the score. Underestimation of the score will inevitably result in a slow dissemination of the work segments, while overestimation will cause some workers to become overloaded.

3 CORRELATED SOCIAL CROWDSENSING WORKERS

In this section, we first provide a mathematic model to represent the correlation among SC workers, and then, we design a Markov Chain-based method to model the contacting behavior of each worker for a period of time. In Section V, we compare several workload allocation algorithms by using the synthetic data generated by this Markov Chain.

3.1 Correlation Model

Based on a user's real-time contacting frequency, we associate a virtual state to each user, called *active level*. For instance, if v_i contacts with other workers 20 times per hour and v_j contacts others 1 time per hour, then v_i is more active than v_j. So, v_i's active level is higher than that of v_j. Let σ_i be the active level of v_i, and there are $2k$ value types in total, $\sigma_i = \{-k, \cdots, -2, -1, 1, 2, \cdots, k\}$. In reality, we can define the value of a user's active level using thresholds. The larger the value of k is, the finer the granularity used: when $k = 1$, active levels indicate whether a user is online ($\sigma_i = 1$) or offline ($\sigma_i = -1$), while when $k = 2$, they stand for "frequently contacting", "moderate", "seldom contacting", and "offline". Active level is a random variable. For example, a user online contacted his friends for a while, then went out. His active level changes from high to low.

Our paper considers *the correlations between friends' active levels*. For example, a pair of positively correlated friends may often become online or offline at similar time. Let σ represent the active levels of *all* workers at any moment, $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_i, \ldots, \sigma_m)$. The probability distribution of workers' active levels $P(\sigma)$ can be represented as follows.

$$P(\sigma) = \frac{1}{Z} e^{\sum_i X_i \sigma_i + \sum_{i \neq j} X_{ij} \sigma_i \sigma_j + \sum_{i \neq j \neq k} X_{ijk} \sigma_i \sigma_j \sigma_k \ldots} \quad (2)$$

where $\{X\}$ are parameters. Note that, when $k = 1$, any m-dimensional probability distribution can be represented in the form of Eq. 2. Since we are focusing on the correlations between friends, we approximate $P(\sigma)$ as follows.

$$P(\sigma) \approx \frac{1}{Z} \times e^{\sum_{i \neq j} J_{ij} \sigma_i \sigma_j + \sum_i h_i \sigma_i} = \frac{1}{Z} e^{H(\sigma)} \quad (3)$$

where Z is a normalization factor. J_{ij} represents the correlation between v_i and v_j: $J_{ij} > 0$ indicates a positive correlation, while $J_{ij} < 0$ indicates a negative correlation; $J_{ij} = 0$ means independent. h_i indicates each worker's own tendency for being active or inactive. The larger a worker's h value is, the larger the chance that the worker is active. The relative value between h and J determines how strongly friends' active levels influence a worker. For general conditions where $k > 1$, we take Eq. 3 as an approximation of $P(\sigma)$. Eq. 3 shows that there are more chance for positively correlated friends to appear/disappear together. For example, when $k = 1$, v_i and v_j appearing/disappearing together means $\sigma_i = \sigma_j = \pm 1$; similarly, the complementary case indicates $\sigma_i = -\sigma_j$. Based on Eq. 3, we have $P(\sigma_i = \sigma_j) = \frac{1}{Z} e^{J_{ij} \sigma_i^2} > P(\sigma_i = -\sigma_j) = \frac{1}{Z} e^{-J_{ij} \sigma_i^2}$.

3.2 Simulate Correlations

In this subsection, we build a discrete-time Markov Chain, $(\sigma^t)_{t=0}^{\infty}$, to simulate the changes of users' active levels in a period of time. We assume that, at any moment, there is at most one worker changing its active level, and we take each possible combination of workers' active levels as one state of the Markov Chain. $\sigma^t = \{\sigma_1^t, \sigma_2^t, \ldots, \sigma_m^t\}$ gives the snapshot of all workers' active levels at time t. For the ease of description, here, we only show the case of $k = 1$: each worker is either active ($\sigma_i = 1$) or inactive ($\sigma_i = -1$). Note that we test both conditions where $k = 1$ and $k = 2$ in Section V. Simulation results show that $k = 1$ is sufficient for our problem. Let $\sigma' \sim_j \sigma$ represent that only worker v_j's active level is different between states σ and σ', and let

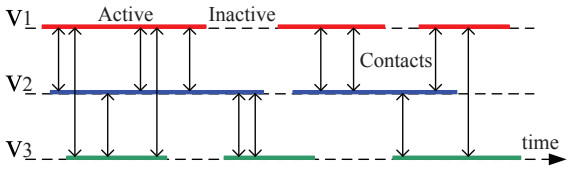


Figure 3: Contacts vs. active/inactive statuses.

$P(\sigma, \sigma') = P(\sigma^{t+1} = \sigma' | \sigma^t = \sigma)$ be the transition probability from state σ to σ'. We have:

$$P(\sigma,\sigma') = \begin{cases} \frac{1}{m} \cdot \frac{e^{\sum_{i\neq j} J_{ij}\sigma_i\sigma'_j + h_j\sigma'_j}}{\sum_{\sigma_j=\pm 1} e^{\sum_{i\neq j} J_{ij}\sigma_i\sigma_j + h_j\sigma_j}} & \text{if} \quad \sigma' \sim_j \sigma \\ \frac{1}{m} \cdot \frac{e^{\sum_{i\neq j} J_{ij}\sigma_i\sigma_j + h_j\sigma_j}}{\sum_{\sigma_j=\pm 1} e^{\sum_{i\neq j} J_{ij}\sigma_i\sigma_j + h_j\sigma_j}} & \text{if} \quad \sigma' = \sigma \\ 0 & \text{if} \quad \text{others} \end{cases} \tag{4}$$

where m is the total number of workers, and σ'_j is the active level of user v_j in state σ'.

THEOREM 3.1. *The stationary distribution of the constructed Markov Chain, $(\sigma^t)_{t=0}^{\infty}$, is equal to $P(\sigma)$ from Eq. 3.*

Due to page limits, we skip the proof. The basic idea is to show that $P(\sigma) = \sum_{\sigma'} P(\sigma')P(\sigma', \sigma)$. In practice, based on real data, one may learn J and h in Eq. 3 by the Levenberg-Marquardt Algorithm, and then, simulate and test any similar (but different) condition using Eq. 4. Or, one can directly give certain values to J and h, and generate data.

Based on the simulated active levels, stochastic contacts are further generated whenever a pair of friends' statuses are both active, as illustrated by Fig. 3. The contacting intervals follow an exponential distribution. In a general case, where $k > 1$, for each pair of friends v_i and v_j, the average contacting frequencies, $\lambda_{ij}(\sigma_i, \sigma_j)$, may be different for each type of active level combination. Admittedly, the real world is more complex than our model. Instead, our model provides an approach that simulates the correlation among workers: we can generate workers' contacting data with any desired correlation degree, which can be further used to check the impacts of correlation on the workload allocation problem.

4 WORKLOAD ALLOCATION

To make better use of system resources, a workload allocation strategy should be able to quickly establish a balanced workload distribution among workers. In this paper, we allocate workloads according to a worker's overall computing capacity. Due to the distributed feature of SC, we approximate the overall capacity based on workers' 2-hop information. In this section, we first propose a deterministic algorithm, which approximates the overall capacity without considering the correlation, and then, we provide another stochastic algorithm, which explicitly exploits the correlations. By using the Markov Chain method from Section III, we compare these two algorithms in Section 5. Experimental results show that the deterministic approach is a good

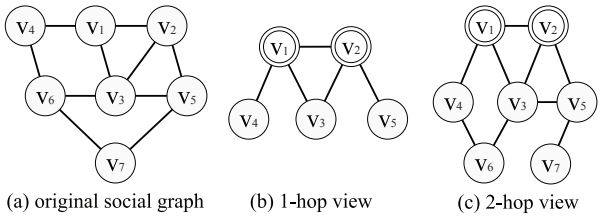


Figure 4: The local view of contacting graph.

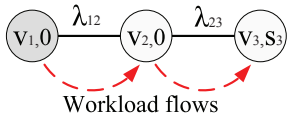


Figure 5: Potential capacities's estimation.

approximation algorithm, which implicitly incorporates the correlation into the estimation of the overall capacity.

In order to avoid making inconsistent workload allocation decisions, whenever workers contact each other, they are required to share their neighbors information within r-hop, including working speeds s_i, average contacting frequencies λ_{ij}, and social relations.

4.1 Basic Rule for Workload Allocations

Assume that, at a time, v_1 contacts v_2, and that their carrying workloads are w_1 and w_2, respectively. Let w_1^* and w_2^* be their updated workloads. Assigning workloads according to users' individual working speeds is the simplest way: $(w_1^*, w_2^*) = \left[(\frac{(w_1+w_2)s_1}{s_1+s_2}, \frac{(w_1+w_2)s_2}{s_1+s_2})\right]$, where s_1 and s_2 are the local computing speeds of v_1 and v_2. However, since SC allows workers to further forward the workload to others, in order to allocate workloads more properly, one should not only consider the computational capacity of the current contacting person, but also the capacity of his future potential contacts. Let q_i stand for worker v_i's overall capacity, which assembles both current and future contacts' computational capacities. The basic rule for workload allocation during a pairwise contact becomes the following:

$$(w_1^*, w_2^*) = \left[\frac{(w_1+w_2)q_1}{q_1+q_2}, \frac{(w_1+w_2)q_2}{q_1+q_2}\right] \tag{5}$$

In SC, a single and fixed-value overall computation capacity q is not sufficient for providing appropriate workload allocation. Let's take Fig. 5 as an example. Suppose that only v_1 is carrying work, while v_2 and v_3 are idle. Let q_i be the overall computation capacity of worker v_i, and s_i be its local working speed. One unique feature of SC is that there are zero-speed workers. Although they do not process tasks' data, the existence of them significantly accelerates the propagation of work segments. In Fig. 5, suppose $s_1 = s_2 = 0$, $s_3 \neq 0$, and that v_1 first contacts v_2 and then v_2 contacts with v_3. Here is the dilemma for the value of q_2. At the first time of contact, we need $q_2 > 0$ and $q_1 = 0$ such that v_1 will give all his workload to v_2, while for the second time, we

Algorithm 1: Pairwise Overall Speed

```
/*Suppose that v_u is contacting with v_u'*/
Eliminate node v_u' from social contact graph G
Initialization q_u ← s_u /*Local contribution*/
for i ∈ N(u) do
    q_u ← q_u + λ_ui s_i /*1-hop contribution*/
    for j ∈ N(i) do
        q_u ← q_u + λ_ui λ_ij s_j /*2-hop contribution*/
        if j ∈ N(u) then
            q_u ← q_u − λ_ui λ_ij λ_uj s_j /*Double counted*/
Return q^u_uu' ← q_u
```

need $q_2 = 0$ and $q_3 > 0$ so that v_2 will not keep any workload himself. Clearly, we cannot find out such a fixed number for q_2. Hence, a single and fixed-value q_i is not sufficient for estimating workers' overall computation capacities.

4.2 Overall Capacity Estimation without Correlation

The capacity q_i reflects the amount of workload that v_i and his friends can process within one unit of time. If the inter-contacting time T between a pair of workers follows exponential distribution with parameter λ, the probability that the workers contact each other within the unit time can be approximated as $P(T \leq 1) = 1 - e^{-\lambda} \approx \lambda$.

In SC, a workload exchange happens when a pair of friends contact each other. The values of q only need to discriminate the capacity's difference between the current contacting workers. Here, we propose a new concept, called Pairwise Overall Capacity (POC): the overall capacity of a worker is partially determined by whom the worker is compared with. Let $q^u_{uu'}$ and $q^{u'}_{uu'}$ be the respective overall capacities when v_u contacts $v_{u'}$. For avoiding the situation of cyclic helping relations, $v_{u'}$ is ignored during the computation of v_u's $q^u_{uu'}$, and vise versa.

A worker's POC is computed as the summation of his own working speed, s_i, and the expected amount of workload that his friends can help within a unit of time. In Fig. 4 (c), the expected amount of capacity that v_2 obtains from v_5 via direct contacts is $P(T_{25} \leq 1)s_5 = \lambda_{25}s_5$; the expected capacity via indirect contacts (2-hop) is $P(T_{23} \leq 1)P(T_{35} \leq 1)s_5 = \lambda_{23}\lambda_{35}s_5$. However, there is a double-counted condition where v_5 can be reached via both 1-hop and 2-hop contacts. Hence, for v_2, the capacity obtained from v_5 equals $(\lambda_{25} + \lambda_{23}\lambda_{35} - \lambda_{25}\lambda_{23}\lambda_{35})s_5$. Algorithm 1 gives the procedure for computing the POC. Each worker can locally compute the POC of himself and the other contacting worker. When computing $q^u_{uu'}$, Algorithm 1 only takes v_u's 2-hop information into account.

4.3 Overall Capacity Estimation with Correlation

In order to better estimate the overall capacity q, we incorporate the concept of the correlated active level into the POC.

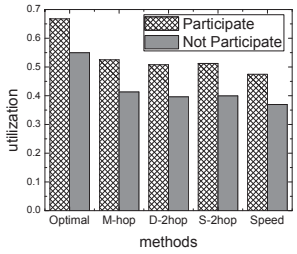


Figure 6: The impact of zero-speed workers

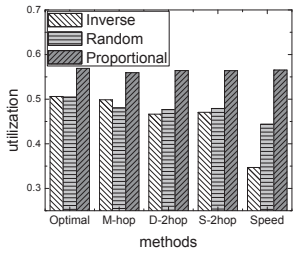


Figure 7: Speed distribution's impacts

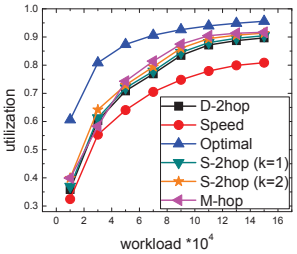


Figure 8: S-2hop's parameter: $k = 1$ VS. $k = 2$

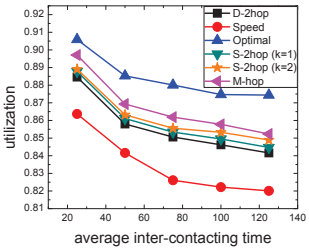


Figure 9: Ave. inter-contacting intervals

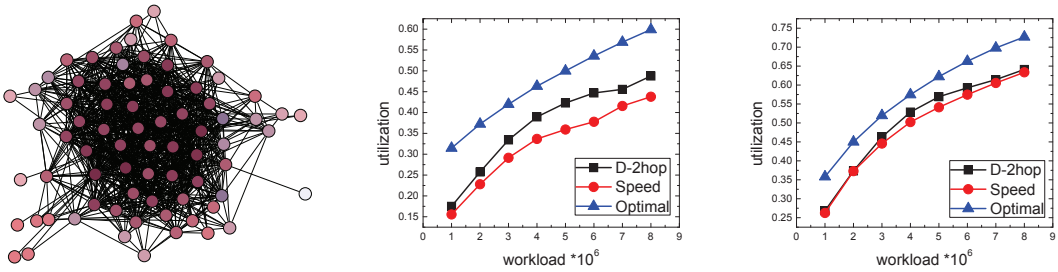
The idea is to add a predicator about the availability of future contacts based on the current information $P\{\sigma_i^{t+1} = \sigma_i', \sigma_j^{t+1} = \sigma_j' | \sigma_i^t = \sigma_i, \sigma_j^t = \sigma_j\}$, and to further apply a finer capacity estimation based on the predication, where the contacting frequency λ in Algorithm 1 is refined by the following equation:$\lambda_{ij}(\sigma_i, \sigma_j) = \beta_{ij} \times \frac{(\sigma_i+k)(\sigma_j+k)}{4k^2}$, where β_{ij} reflects the average number of contacts made by v_i and v_j in a unit of time. Note that the predicator can be created based on long-time observations, and that $\lambda_{ij}(\sigma_i, \sigma_j)$ essentially gives the average contacting frequency for each type of active level combination.

5 EVALUATION

5.1 Evaluation Setup and Metric

For ease of comparison, we call the scheme, which splits workloads according to workers' local speeds, *Speed*; the deterministic POC, which does not consider correlations, is called *D-2hop*; the stochastic POC that explores a worker's active level is called *S-2hop*. We also design an approach, *M-hop*. The basic idea is to use *global* information to create a transition matrix about the average percentage of workloads that a worker forwards to its contacts, and then, compute the stationary distribution. The details of *M-hop* can be found in our previous work [4]. Based on the posterior knowledge, one can compute the shortest path from a task owner to workers, and the optimal result can be found by letting all participants complete their workload at the same time. Although this algorithm is useless in reality, we use its results as a comparison criterion. We use *Optimal* to represent it.

We first use synthetic data to show that the result of D-2hop is very close to that of the S-2hop, and then, we test



(a) Social graph (b) Inversely (c) Random

Figure 10: Real data: Sigcomm09

the D-2hop on real data. The synthetic simulation is based a real social graph: karate club [3]: We use our Markov Chain model to create σ_i for each worker. Whenever two workers are active, they may have social contacts. The distribution of inter-contacting times follow an exponential distribution. Workers' contacting frequencies and tasks' owners are randomly selected. Workers' computing speeds follow a uniform distribution from 0 to 5 units of work per unit of time. The real data comes from the Sigcomm09 data set [6].

We use utilization as our evaluation metric. The utilization is computed as the ratio between the number of computing resources that have been used during the observation and the total number of available resources that the system is able to provide. Due to the propagation delays, the utilization can never reach 100%.

5.2 Simulation results

Fig. 6 shows the impacts of the zero-speed workers' participation. Since the zero-speed workers can accelerate the propagation of work segments, SC encourages people to participate, even if they will not work on the task by themselves. We can see that the utilization of all methods decreases when the zero-speed workers are excluded from the task, and the reduced amount is almost 10%. In Fig. 7, we test the impacts of the relation between a worker's local speed and the number of neighbors on a social graph. When the speed is proportional to the node degree, the utilization of all methods is significantly increased. When the speed is inversely proportional to node degree, *Speed* has the most impact. Since the frequently contacting users have lower speeds, work segments are trapped around the task owner.

S-2hop considers workers' active levels, and there are $2k$ levels. From Fig. 8, we can see that, by using a larger k, the results of *S-2hop* are improved, but the difference between $k = 1$ and $k = 2$ is very small, only about 0.5%. For the rest of the simulation, we will only consider the case where $k = 1$. In addition, the result of *S-2hop* is close to that of *M-hop*. Considering that *M-hop* explores the global information, *S-2hop* is a good approximation, even if $k = 1$. Fig. 9 shows the impacts of the average length of inter-contacting time. With the increasing length of intervals, it takes more time to propagate work segments, and therefore, the utilizations decrease.

Fig. 10 gives the experiment's results on real data. When the distribution of workers' speeds is inversely proportional to their number of friends, our approach can improve the system utilization by almost 10%. Moreover, the changing pattern of each method is consistent with our synthetic data-based simulation results.

6 CONCLUSION

This paper proposes a new system, social-crowdsensing, and further considers the workload allocation problem among workers. By analyzing real data, we find that friends' activities are not independent. Many questions arise once crowdsourcing workers are correlated. To our best knowledge, no one has systematically studied this problem before. In this paper, we first provide a model for describing and simulating the users' correlated activities. Then, we propose two schemes, with and without explicitly using the correlation information, to locally allocate participants' workloads during social contacts. Counterintuitively, extensive experiments show that, even if we explicitly imposed a correlation among users, the correlation information can only bring a marginal improvement to the system utilization and the work's makespan.

ACKNOWLEDGMENTS

This research was supported in part by NSF grants CNS 1629746, CNS 1564128, CNS 1449860, CNS 1461932, CNS 1460971, CNS 1439672, CNS 1301774, and ECCS 1231461.

REFERENCES

[1] aws.amazon.com/pricing/mturk/ visited at 2014.
[2] www.mturk.com/mturk/ visited at 2017.
[3] www-personal.umich.edu/ mejn/netdata/ visited at 2017.
[4] W. Chang and J. Wu. Progressive or Conservative: Rationally AllocateCooperative Work in Mobile Social Networks. In *IEEE TPDS*, 2014.
[5] L. B. Chilton, G. Little, D. Edge, D. S. Weld, and J. A. Landay. Cascade: Crowdsourcing taxonomy creation. In *ACM SIGCHI*, 2013.
[6] A.-K. Pietilainen. CRAWDAD data set thlab/sigcomm2009 (v. 2012-07-15). http://crawdad.org/thlab/sigcomm2009/, 2012.
[7] S. Zhang, J. Wu, and S. Lu. Minimum makespan workload dissemination in dtns: Making full utilization of computational surplus around. In *ACM MobiHoc*, 2013.