CARROT: Optimizing Task Recommendations in Context-Aware Mobile Crowdsourcing

Azalia Mirhoseini^{+}, Suman Nath*, Michel Goraczko*, and Jie Liu*
$^{+}$Rice University, Houston, TX
*Microsoft Research, Redmond, WA

***Abstract*—We propose CARROT[1], a suite of algorithms for optimizing task recommendation in a mobile crowdsourcing system. Unlike existing systems that list individual tasks for workers to select, CARROT automatically identifies and recommends to workers packages of tasks that are suitable for workers contexts such as their locations. This incentivizes workers to perform more tasks, helping tasks to complete faster even with smaller budgets. We consider packaging tasks for two optimization goals: to maximize the expected number of completed tasks and to dynamically price tasks to maximize the systems utility, which is a function of both task values and task completion rate. We show that the problems are NP-hard and exploit monotonicity and submodularity of the objective functions to devise computationally feasible algorithms with tight optimality bounds. Our experiments with real datasets demonstrate that, compared to a baseline algorithm, CARROT can improve task completion rate by up to $2\times$ with the same budget and $1.5\times$ with almost half the budget.**

I. Introduction

The advances in smart phone technologies along with their ever-increasing popularity have led to development of mobile crowdsourcing markets. Mobile crowdsourcing differs from traditional crowdsourcing in that workers often need to be at specific contexts (such as location, transportation mode, surroundings, etc.) to perform the tasks by using their smart phones. Many research prototypes have demonstrated the value of mobile crowdsourcing, e.g., detection of potholes [1], urban noise and pollution maps [2], [3], traffic monitoring [4], [5], speech data collection [6], etc. Several instances of paid mobile crowdsourcing markets have also emerged commercially including Gigwalk[2], FieldAgent[3], TaskRabbit[4], and Instant.ly[5]. Typical tasks in these markets pay workers a few dollars for capturing photos of buildings or places, checking price and placement of a product in stores, checking traffic, taking location-aware surveys, and so on.

In this paper, we consider the problem of recommending tasks to mobile workers, who are either getting paid or participating voluntarily. Our work is motivated by two major pain points that we have experienced with two existing commercial mobile crowdsourcing platforms (GigWalk and Instant.ly). First, tasks can take a long time to complete. In our experiments, even a simple task (e.g. taking a picture of a nearby specific business) with a reasonable payment of \$5 took up to several days to complete. This is because tasks require workers' mobility and the system sometimes cannot give the task to the "right" worker (who is willing to do it) at the "right" time. Second, mobile tasks are expensive, mainly because they may require workers to travel. In our experiments, a worker demanded \$13 on average (with a stdev. of \$16) to complete a simple task of taking a picture that requires a 5-minutes trip. Both slow task completion rate and high budget requirement are undesirable for task posters, especially when they have a large number of tasks to finish. They are undesirable to the crowdsourcing system as well since they indicate a poor utilization of the workers.

Motivated by the above limitations of existing systems, we aim to design algorithms to recommend the right tasks to the right workers at the right time, thus to increase the chances that tasks are performed quickly by mobile workers, without requiring the task posters to spend a lot of money. We believe that this is a key design goal of a sustaining mobile crowdsourcing platform: by achieving this goal, a platform can attract many task posters and can efficiently utilize its workers. Our solution is based on two key techniques.

Context-aware and Personalized Task Recommendation: We recommend tasks to workers based on their current and predicted *future* contexts, as well as their personal preferences towards different types of tasks. A worker's current context is inferred from his smart phone sensors, while his future contexts and preferences are dynamically learned from historical data such as frequently visited places, transportation modes, completed tasks, preferred times, etc. Exploiting a worker's context enables recommending tasks to suitable workers. For example, a traffic monitoring task can be recommended to workers driving at a target location, while a speech data collection task can be recommended to workers at quiet places (or noisy places, depending on the task requirements). By exploiting future contexts, we can take into account the information about the workers that are currently unavailable but will be available soon. That way we have a holistic view of worker pool and can do better optimizations and price setting. Finally, by exploiting workers' preferences towards task types, payments, distance to travel, etc., we can recommend tasks to workers who are likely to perform them. For example, knowing that a worker performs only near-by picture-taking tasks for a price greater than 2\$/task enables recommending him tasks that satisfy the requirements. CARROT's ability to exploit both current and

[1]Patent approved on November 5, 2015. Publication number: 20150317582
[2]http://gigwalk.com
[3]http://www.fieldagent.net/
[4]https://www.taskrabbit.com/
[5]http://www.instant.ly

future, location and non-location contexts is in contrast to prior efforts that consider only location contexts [7], [8] or only current contexts [9].

Packaging Tasks: Our second technique is motivated by several key discoveries from recent studies of a year-long dataset from a leading mobile crowdsourcing platform [10], [11]. The study points out an important factor behind success of efficient workers (called *super agents*), who constitute 10% of the crowd but perform 80% of the tasks. These efficient workers optimize their efficiency by carefully planning their trips, and bundling a sequence of tasks to perform in a single trip. Bundling tasks helps them to amortize their total trip cost (e.g., travel cost, time) over multiple tasks. In contrast, less efficient workers do not strategize as carefully and therefore have lower efficiency across the board. The study also points out that planning and searching for tasks consume a considerable amount of time, which also explains why many workers do not spend enough time on planning even though it can improve their efficiency.

We exploit the above observations as follows. Rather than just listing all available tasks for workers to choose, we aim to automatically identify *packages of tasks* and provide *recommendations* to workers about these *packages*. Recommending task packages can save significant time of workers for browsing through a large collection of available tasks to find the suitable ones. Packaging tasks have several key advantages as well. First, it can improve workers' efficiency in terms of earnings per unit efforts (e.g., miles traveled), especially for the ones who are less likely to identify the opportunity of packaging tasks by themselves. This also implies a faster task completion rate. Second, task posters may be able to get their tasks performed with smaller budgets as long as the aggregate revenue of a package remain more attractive than the incentive for a single task. Finally, this enables the platform to recommend otherwise unprofitable tasks in the same package as profitable ones, thereby avoiding the scenario where workers greedily choose profitable works leaving unprofitable ones uncompleted. Providing such action plans has been shown to be useful to workers in online crowdsourcing systems [12]; however, we are not aware of any such efforts for the mobile crowdsourcing platforms.

Our packaging algorithm runs proactively in the background given a set of tasks and a set of potential workers. It can leverage push notifications to remind workers of available task packages that meet their contexts.

Contributions: We introduce CARROT (Context-AwaRe Recommendation Of Tasks), a suite of algorithms for personalized, context-aware task recommendation in a mobile crowdsourcing platform. We consider packaging tasks and recommending them to workers for two optimization goals: to maximize the expected number of completed tasks and to dynamically adjust prices to maximize the systems utility, which is a function of both task payments and task completion rate. This is different from most previous efforts assuming homogeneous incentive models of workers [13], [7]. We show that our problems are NP-hard. However, our objective functions are monotonic and sub-modular, which enables us to devise computationally feasible algorithms with tight optimality bounds, inspired by [14]. Our experiments with real datasets demonstrate that, compared to a baseline algorithm, CARROT can improve task completion rate by up to $2\times$ with the same budget and $1.5\times$ with almost half the budget.

II. System Model and Goals

Unless otherwise stated, we assume a paid mobile crowdsourcing platform similar to the commercial ones such as Gigwalk; later we discuss how CARROT can benefit voluntary crowdsourcing as well. Each task needs to be performed at a specific context such as at a location or while the worker is driving, and has a budget and a deadline. A worker can perform multiple tasks and a task (or multiple copies of it) can be performed by multiple workers. A task is called *successfully completed* if it is performed by at least one worker within the deadline of the task. For each successfully completed task, the worker and the platform get paid by the task poster.

CARROT aims to recommend packages of tasks to workers such that the recommendations are likely to increase the number of successfully completed tasks, with small payments. CARROT recommends a task to a worker if the worker has a high probability (say 0.8) to do it. Given a recommended task package, a worker can (1) accept it and perform it within the task deadline, (2) reject it right away, or (3) accept it but fail to do all or some of the tasks in the package within the deadline. CARROT aims to minimize the cases (2) and (3), by learning workers' tendencies and preferences towards different types of tasks. CARROT has a continuous sensing module that runs on worker's smartphone and monitors his physical contexts such as his locations and transportation modes and how he reacts to recommended tasks (e.g., whether he actually completes a task that offers a payment of 10$ and requires him to drive 5 miles). The model allows CARROT to recommend a task to the right person at the right time. For example, it can recommend a location-specific task to a worker who is willing to perform similar tasks at the offered price and is currently near the location or whose daily commuting route includes the location. We describe how CARROT dynamically learns such worker models in the next section.

CARROT takes as input the workers' real time and historical context information (as discussed in Introduction), task properties (such as location, payment, deadline). The output is preferred assignment of packages of tasks to active workers. We pursue two different objectives that will be discussed in details later. The first objective aims only at maximizing task completion rate where task payments are fixed. The second objective aims to optimize both task payments and task completion rate simultaneously by using adaptive pricing. Both the algorithms need models, characterizing historical behavior of mobile workers.

A mobile crowdsourcing platform can proactively run CARROT algorithms in the background, in periodic rounds. In each round, the platform runs CARROT with sets of available tasks and potential workers and pushes notifications to remind workers of recommended task packages that meet their contexts. As mentioned before, a worker may choose not to accept a task recommended to him or he may accept it but fail to complete it. Such uncompleted tasks go back to the pool of available tasks to be considered by CARROT for the next round. CARROT dynamically learns from workers' behavior and adapts to not recommend tasks to a worker who repeatedly rejected similar tasks or fail to finish tasks or does them at unacceptable qualities (to avoid cases (2) and (3) above).

III. Mobile Worker Characterization and Prediction Error

Workers may have different tendencies and strategies to accept and execute tasks on the platform. In general, the preference towards factors such as type of the task, its complexity, and expected payment per unit "effort" (e.g., time spent) varies across different workers. Such preferences can be learned and modeled by analyzing the history of workers; the model can then be used to evaluate the likelihood of a worker to successfully complete a future task. A crowdsourcing platform can use such a predictive model to recommend a task to a worker who will most likely perform the task at the offered payment.

In the following we explain our approach for modeling the worker context over time. Let us denote the likelihood of completing a new task by a worker by $P(y|x;\theta)$, where variable y indicates whether the worker completes a task successfully ($y = 1$) or not ($y = 0$), vector $x = (x_1, x_2, ...)$ includes the real time parameters (such as payment, distance, task complexity, etc.), and vector $\theta = (\theta_1, \theta_2, \theta_3, ...)$ is the learned coefficients/weights corresponding to those parameters.

A. *Regularized logistic regression for predictions*

We exploit regularized logistic regression to build the predictive model and find the parameter weights (i.e., vector θ). Let us denote the collected observations from a worker by $(x^{(i)}, y^{(i)})$ where $1 \le i \le T$. The regression model aims to find θ by minimizing the following cost:

$$arg\min_{\theta} \frac{1}{T} \sum_{1 \le i \le T} log(h_\theta(x^{(i)}))y^{(i)} + log(1 - h_\theta(x^{(i)}))(1 - y^{(i)})^2 + \lambda_r \sum_{j=1,2,...} \theta_j^2, \quad (1)$$

where $h_\theta(x) = P(y = 1|x, \theta) = \frac{1}{1+e^{-\theta^t x}}$ is the likelihood of completing a task by the worker and λ_r is a penalty coefficient, which is a small positive number that shrinks the norm of θ. The cost function minimizes the prediction error over the training dataset. The penalty term, also known as Ridge regularization, is used to avoid overfitting the model. In our evaluations, we perform cross-validation to find λ_r. To solve Equation 1, we use available gradient descent based algorithms that iteratively converge to the solution [15].

B. *Parameterizing worker's behavior*

For learning θ or worker's characterization model, we specify and quantize the parameters that affect a worker's decision to accept or reject a task. These parameters can be determined by assessing different types of context that can possibly contribute to worker's decisions and their subsequent performance. Example parameters include but are not limited to task payment, distance, complexity, deadline, the time it takes to complete a task, and the aggregated duration of all crowdsourcing tasks a worker has in his to-do list. By monitoring workers behavior with respect to these parameters, CARROT can learn personalized θ (and thus the characterization model) over time.

C. *Continuous updates*

The workers characterizations are updated over time as more data is collected from them. We apply the changes in workers behavior to our model periodically to have an accurate and up to date estimation of their behavior. To be more exact, every time a new observation pair (x, y) corresponding to a worker is collected, the parameter vector θ for that worker is updated as follows: update every θ_i in $(\theta_1, \theta_2, ...)$, with

$$\theta_i = \theta_i - \alpha(h_\theta(x) - y)x. \quad (2)$$

Where α is a weight parameter whose value determines the effect of the latest observation on the overall prediction modeling. A larger α accentuates the impact of the latest observation. The update equation naturally follows the gradient descent approach that has been employed to solve Equation 1.

Continuous revising of our prediction model allows the platform to keep track of the changes in workers behavior over time. Another advantage is its simplicity. Instead of processing a large number of samples at once which incurs undesirable computational complexity, we can update our model in real time upon receiving a new sample.

IV. Task Packaging Algorithms in CARROT

In this section we explain CARROT's task packaging optimization problem, objectives, and solutions. The problem input is workers real time and historical information and task properties (such as location, budget, deadline). The output is packages of tasks that can be recommended to workers. We pursue two different objectives that will be discussed in details in the next two subsections. The first objective aims only at maximizing task completion rate where task payments are fixed. The second objective aims to optimize both task payments and task completion rate simultaneously by using adaptive pricing. Package size is tunable in both algorithms.

A. *Maximizing the number of completed tasks*

We denote the set of available tasks by $\mathcal{T} = \{\tau_1, \ldots, \tau_m\}$ and the set of workers by $\Omega = \{\omega_1, \ldots, \omega_n\}$. Note that Ω consists of two kinds of workers. The first type are those who are currently available and are actively looking for tasks. The other type of workers are the ones that are currently unavailable but, based on prior information, are expected to be available before tasks deadlines. We define variable $X_{\tau\omega}$ as follows: $X_{\tau\omega} = 1$ if worker ω ($\in \Omega$) completes the recommended task τ ($\in \mathcal{T}$), otherwise $X_{\tau\omega} = 0$. Let $P_{\tau\omega}$ be the probability that worker ω completes task τ.

We denote by $g_\omega(\mathcal{T}_\omega)$, the expected number of tasks that are completed by worker ω if set ($\mathcal{T}_w \subset \mathcal{T}$) of tasks is recommended to them. Thus,

$$g_\omega(\mathcal{T}_\omega) = E[\sum_{\tau \in \mathcal{T}_\omega} X_{\tau\omega}] = \sum_{\tau \in \mathcal{T}_\omega} P_{\tau\omega} X_{\tau\omega}.$$

Our goal is to distribute tasks such that the overall number of tasks that are successfully completed are maximized. We now formally define **Problem 1** as follows:

$$\begin{aligned} \textbf{OF} &: \textit{max } \mathcal{G}(\mathcal{T}, \Omega) = \sum_{\omega \in \Omega} g_\omega(\mathcal{T}_\omega), \\ \textbf{Const}. &\ \forall \ \ \omega_i \neq \omega_j \in \Omega, \quad \mathcal{T}_{\omega_i} \cap \mathcal{T}_{\omega_j} = \emptyset. \end{aligned} \tag{3}$$

The objective maximizes the expected number of successfully assigned tasks after matching all tasks and workers. The constraints ensure that each task is recommended to at most one worker.[6]

Problem complexity. It is straightforward to observe the problem's combinatorial complexity. For each worker, there is an exponential number of different suitable task sets for which $g_\omega(.)$ should be evaluated. The probability that a worker completes a particular task is not only dependent on the task, but also is a function of the set of tasks that he has accepted to do. Even if the probabilities $P_{\tau\omega}$ could be independently computed for each worker ω and were not affected by the tasks in $\mathcal{T}_\omega$, Problem 1 would become an instance of a Knapsack Problem which is proved to be NP-hard [16]. Besides, workers have different characteristics that result in distinct $g_\omega(.)$ functions for each individual, rendering the problem even more complex. Thus, no optimal solution can be found in polynomial time (unless P=NP). In the following, we discuss how CARROT develops an efficient algorithm with tight optimality bound.

Computing $g_\omega(\mathcal{T}_\omega)$. To compute the objective function for a set of tasks, we should consider the effect of those tasks on the probability as a whole. Since the probability that worker ω completes a task τ ($y_\tau = 1$) is not only a function of task τ but also a function of the tasks in $\mathcal{T}_\omega$. For example, from parameters θ that we have listed in the previous section, task distance and the aggregated time for completing tasks are among the ones that are affected by the set $\mathcal{T}_\omega$. Finding the exact minimum traveling distance/time required for each task, given the set $\mathcal{T}_\omega$ can be very challenging. The problem becomes equivalent to solve a classical traveling salesman problem which is known to be NP-hard [16]. To compute the distance/time for each task, we use the nearest neighbor greedy algorithm that is an approximate solution with optimality guarantees. In this case we assume worker chooses the nearest unvisited task in $\mathcal{T}_\omega$ in his next move. Note the term "near" is not limited to distance and can be broadly defined on the metric space of context θ.

Properties of $g_\omega(\mathcal{T}_\omega)$. We claim that $g_\omega(\mathcal{T}_\omega)$ *is a monotonically increasing and submodular function*. The monotonicity property hold due to the definition of $g_\omega(\mathcal{T}_\omega)$. For $g_\omega(\mathcal{T}_\omega)$ to be submodular, it has to comply to the following two conditions: (1) it should always be positive, and (2) the marginal value of adding a new task should be monotonically decreasing as the number of tasks increase. In other words,

$$g_\omega(\mathcal{T}^1_\omega \cup \tau) - g_\omega(\mathcal{T}^1_\omega) \geq g_\omega(\mathcal{T}^2_\omega \cup \tau) - g_\omega(\mathcal{T}^2_\omega).$$

where $\mathcal{T}^1_\omega \subset \mathcal{T}^2_\omega$ are subsets of tasks. The first property follows directly from positiveness of probability function. The second property is intuitive; if workers already have a larger number of tasks in their to-do list they might not be as interested in a new available task compared to the case where they have a smaller subset of to-do jobs and the same new task is offered to them. In our characterization model, it is easy to observe that parameters such as total task completion durations/distances exhibit submodluar characteristics.The submodular (diminishing return) behavior of workers with respect to the increasing number of tasks has been shown in existing crowdsourcing user studies [11], [17].

An efficient solution. Given that $g_\omega(.)$ is monotonic and submodular, we can exploit theoretical solutions to optimize the objective function in Problem 1. A greedy approach is proven to give 1/2-approximation to the solution [18]. However, more sophisticated approaches provide improved optimization guarantees [14]. For a sub-modular and monotonic objective, the randomized continuous greedy approach achieves the following bounded w.r.t the optimal solution:

$$\mathcal{G}(\mathcal{T}, \Omega) \geq (1 - 1/e)OPT. \tag{4}$$

We propose Algorithm 1 which exploits a randomized continuous greedy approach suggested in [14] to repeatedly compute a marginal benefit achieved by assigning a new task to a worker. This marginal profit is computed in Step 2 of Algorithm 1. For computing $\Delta g_\omega(t)$ (the marginal profit) at each iteration, all the tasks that have already been assigned to ω by that iteration should be taken into account. This should be done by adjusting relative parameters including the task distance and total duration.

B. Maximizing utility with adaptive pricing

We exploit adaptive pricing to maximize the expected *utility* of the platform. We define a utility function that pursues a dual objective: maximizing task completion rate while minimizing the payments. The utility function for recommending set of tasks $\mathcal{T}_\omega \subset \mathcal{T}$ to worker ω is defined as:

$$g^{utl}_\omega(\mathcal{T}_\omega) = \Sigma_{\tau \in \mathcal{T}_\omega} P_{\tau\omega} X_{\tau\omega} \exp(-\lambda \frac{\pi_{\tau\omega}}{\pi_b}).$$

where π_b is the maximum budget for the task τ; $\pi_{\tau\omega}$ is the payment to be offered to worker ω for doing τ; λ is a parameter defined by the platform that can enforce different utility objectives. There is an apparent tradeoff between task payment π_τ and the probability of task completion $P_{\tau\omega}$; a higher payment increases the chances that a task is completed but decreases the utility. A larger λ yields to more savings in budget at the cost of lower task acceptance/completion rate.

[6]CARROT can also assign a single task to multiple workers by replicating the task in its algorithms.

Algorithm 1 : Maximizing the number of completed tasks with Fixed Task Payments.

```
Input: A set of workers $\Omega$ with characterization model $P_\omega$, and a set of tasks $\mathcal{T}$ with properties for each task $\tau \in \mathcal{T}$.
Output: Recommendation of a package of tasks $\mathcal{T}_\omega \subset \mathcal{T}$ to each worker $\omega \in \Omega$.
Objective: Maximizing expected task completion rate $\mathcal{G}(\mathcal{T}, \Omega)$.

1. Let $\delta = 1/(mn)^2$. Initialize $t = 0$ and $\mathcal{P}_{\tau\omega}(0) = 0$ for all $\tau$, $\omega$.
2. Create set $R_\omega(t)$ as a random set which contains each task $\tau$ with probability $\mathcal{P}_{\tau\omega}(t)$. Estimate the expected marginal profit of assigning task $\tau$ to worker $\omega$,
       $\Delta g_\omega(t) = E[g_\omega(R_\omega(t) + \tau) - g_\omega(R_\omega(t))]$
   by taking the average of independent samples.
3. For each $\omega$, let $\tau_\omega(t) = argmax\ \Delta g_\omega(t)$ be the preferred task for worker $\omega$.
4. Set $\mathcal{P}_{\tau\omega}(t+\delta) = \mathcal{P}_{\tau\omega}(t)$ for the preferred task $\tau = \tau_\omega(t)$ and $\mathcal{P}_{\tau\omega}(t+\delta) = \mathcal{P}_{\tau\omega}(t)$ for all other tasks.
5. Increment $t$: $t = t + \delta$; if $t < 1$, go back to Step 2.
6. Allocate each task $\tau$ independently with probability $\mathcal{P}_{\tau\omega}(1)$ to worker $\omega$.
```

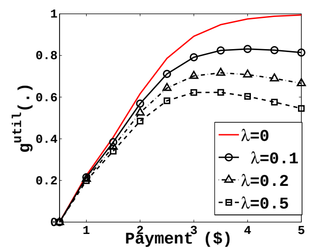


Fig. 1: Utility function for different λ's. By increasing λ, a higher reward will be given to reduced task payments rather than task completion. By setting $\lambda = 0$, the utilization objective becomes the same as task completion objective, i.e., $g_\omega^{util}(.) = g_\omega(.)$.

We define **Problem 2** as follows:

$$\textbf{OF} : max\ \mathcal{G}^{utl}(\mathcal{T}, \Omega) = \sum_{\omega \in \Omega} g_\omega^{utl}(\mathcal{T}_\omega),$$
$$\textbf{Const}.\ \forall_{\tau \in \mathcal{T}}\ 0 \le \pi_{\tau\omega} \le \pi_b,$$
$$\forall_{\omega_i \neq \omega_j \in \Omega}\ \mathcal{T}_{\omega_i} \cap \mathcal{T}_{\omega_j} = \emptyset. \tag{5}$$

The first constraint ensures that the payments are non-negative and less than tasks budgets. The second constraint enforces disjoint distribution of tasks.

Computing $g_\omega^{utl}(\mathcal{T}_\omega)$. To compute the utility function over a set of tasks, we first find optimized payments ($\pi_{\tau\omega}^\star$) by maximizing $P(\pi_{\tau\omega}|\mathcal{T}_\omega)exp(-\lambda\frac{\pi_{\tau\omega}}{\pi_b})$. Figure 6 shows how optimal payments vary with respect to different values of λ. Next, we follow the same steps as before to compute $g_\omega()$.

Properties of $g_\omega^{utl}(\mathcal{T}_\omega)$. We claim that $g_\omega^{utl}(.)$ *is also mono-*

Algorithm 2 : Maximizing Utility with Dynamic Payments.

```
Input: A set of workers $\Omega$ with characterization model $P_\omega$, and a set of tasks $\mathcal{T}$ with properties (including maximum value: $val_\tau$) for each task $\tau \in \mathcal{T}$.
Output: Recommendation of a package of tasks $\mathcal{T}_\omega \subset \mathcal{T}$ to each worker. $\omega \in \Omega$.
Objective: Maximizing expected platform utility $\mathcal{G}^{utl}(\mathcal{T}, \Omega)$ which has a dual objective: to maximize task completion rate and to reduce payments.

1. For each $\tau \in T$ and each $\omega \in \Omega$ solve the following optimization problem to find an optimal task payment $\pi_{\tau\omega}^\star$
       $\pi_{\tau\omega}^\star = argmax_\pi \ \exp(-\lambda\frac{\pi}{\pi_b})P_{\tau\omega}$.
2. For each task $\tau \in \omega$, update the task properties by setting its payments to personalized $\pi_{\tau\omega}^\star$ values.
3. Run algorithm 1, with updated task properties from step 2, and substituting the objective by $\mathcal{G}^{utl}(\mathcal{T}, \Omega)$ to determine the package assignments.
```

tonic and submodular. The monotonicity is apparent from the utility function's definition. To prove submodularity, let us assume that τ is a new task. We should prove that if $\mathcal{T}_\omega^1 \subset \mathcal{T}_\omega^2 \subset \mathcal{T}$ the following property holds:

$$P_{\tau\omega}(\pi_{\tau\omega}^1|\mathcal{T}_\omega^1)exp(-\lambda\frac{\pi_{\tau\omega}^1}{\pi_b}) > P_{\tau\omega}(\pi_{\tau\omega}^2|\mathcal{T}_\omega^2)exp(-\lambda\frac{\pi_{\tau\omega}^2}{\pi_b}), \tag{6}$$

where $\pi_{\tau\omega}^i$ is the optimal payment determined for a task τ that is assigned to a worker ω for $i = 1$ and 2. Since $\pi_{\tau\omega}^1$ is optimal, then

$$P_{\tau\omega}(\pi_{\tau\omega}^1|\mathcal{T}_\omega^1)exp(-\lambda\frac{\pi_{\tau\omega}^1}{\pi_b}) > P_{\tau\omega}(\pi_{\tau\omega}^2|\mathcal{T}_\omega^1)exp(-\lambda\frac{\pi_{\tau\omega}^2}{\pi_b}), \tag{7}$$

Since we already know that $g_\omega(.)$ is submodular, the following inequality holds:

$$P_{\tau\omega}(\pi_{\tau\omega}^2|\mathcal{T}_\omega^1)exp(-\lambda\frac{\pi_{\tau\omega}^2}{\pi_b}) > P_{\tau\omega}(\pi_{\tau\omega}^2|\mathcal{T}_\omega^2)exp(-\lambda\frac{\pi_{\tau\omega}^2}{\pi_b}). \tag{8}$$

From Equations 7 and 8, we can verify that Equation 6 holds. Thus, $g_\omega^{utl}(\mathcal{T}_\omega)$ is submodular.

Efficient solution. Now that we have proved the submodularity and monotonicity of $g^{utl}(.)$, we can use a similar randomized continuous greedy approach to solve Problem 2. Algorithm 2 summarizes the solution. We first compute optimal payments for maximizing the utility function for each task, worker pair. Then we use Algorithm 1 to find the task package assignments.

V. Discussion

A. Supporting general context

In describing CARROT algorithms, we have so far considered a few example context-based parameters and a relatively long deadline. CARROT algorithms can support arbitrary contexts and short deadlines as well, by appropriately adjusting the utility or probability $\mathcal{P}_{\tau\omega}$ (associated with assigning task τ to worker ω). For example, consider a traffic monitoring task that requires workers in "driving" context and on a specific freeway. If the task is real-time or has a short deadline such that the platform needs to recruit workers who are *currently* driving on the specific freeway, CARROT sets $\mathcal{P}_{\tau\omega} = 0$ for all workers ω who are not currently driving on that freeway. Similarly, if the deadline is long, CARROT predicts if a worker will be driving within the time window specified by the deadline, and if not, it sets his $\mathcal{P}_{\tau\omega} = 0$. In assigning a task τ, both Algorithm 1 and Algorithm 2 treat all workers ω with $\mathcal{P}_{\tau\omega} = 0$ as not suitable for the task.

B. Richer Prediction Model

We considered having one prediction model for each worker. With more training data, CARROT can build much richer models to consider factors such as time of the day/week, weather, activities in the recent past, etc. CARROT algorithms can naturally handle such richer models.

C. Feedback to task-posters

While posting a task, a task poster needs to decide its payment amount. This is particularly challenging for task posters who do not have much experience with the platform and its workers. Without enough statistics of the workers, it is almost impossible for the task poster to decide the minimum payment necessary to get his tasks completed (with high probability) within an acceptable deadline.

CARROT algorithms can help a corwdsourcing platform in order to provide such feedback to task posters. After a task poster specifies his tasks with their required contexts and deadlines, the platform can present him with a graph showing the completion rates for various payment amounts, generated as follows. To estimate the task completion rate for a given payment π, the platform runs Algorithm 1, with $\mathcal{P}_{\tau\omega}$ set to be the probability of the worker ω doing the task τ for the payment π. CARROT uses worker model learned from the historical data to determine the probability. Seeing the task completion rates for different payment values *before* actually posting a task, the task poster can make an informed decision on the minimum payment value for a completion rate acceptable to him.

D. Voluntary workers

A crowdsourcing system with voluntary participants can use Algorithm 1 with a fixed payment of zero. Algorithm 2 is not applicable to such systems.

VI. Evaluations

In this section we evaluate CARROT algorithms with real-world datasets. We first describe workers characterization results. Next, we present results to demonstrate how CARROT, by adopting a context-aware approach, efficiently handles the trade-off between budget allocation and task completion rate. Finally, we perform sensitivity analysis to evaluate our approach in presence of variations in workers behavior.

A. Datasets and Evaluation Set up

We conducted user studies in three phases. In the first phase, we built worker models of 109 workers: 12 graduate and undergraduate students and 97 workers from two leading crowdsourcing platforms (7 from Gigwalk and 90 from instant.ly). The survey included several task offers at various real business locations (e.g., restaurants or coffee shops) within a 15 mile radius and with different payments. Each task included going to the location and taking pictures of the front door of the business. The tasks were shown on a map and participants were asked whether they accepted or rejected the offer. One offer was shown at a time. We use the data to model the workers. We did not have daily location traces for Gigwalk and Instant.ly workers; instead we used an anonymized location tracking dataset from Twitter users (in the same area as where the survey was conducted) that contained location information (latitude and longitude) of the tweets. Each participant was matched to a random Twitter user. We used a density-based spatial clustering method called DBSCAN in order to process Twitter locations and assign daily routes to participants.

In Phase 2 and Phase 3 of the study, participants (i.e., workers) were required to select and actually complete tasks. In Phase 2, we used a first-come-first-serve (FCFS) model, where we list all tasks with their payments on a map and remove a task from the list as soon as a worker chooses to perform a task (so that subsequent workers cannot choose it). In Phase 3, we used CARROT's task packaging algorithms with price adjustments on the same tasks we used in Phase 2. For a fair comparison, Phase 2 and Phase 3 are run on the same set of workers, and hence we limit these phases only to 12 students whom we had a better control of. To collect physical context including the participants routine locations and daily trips, their GPS data were extracted for a period of two weeks.

Interpreting the results for larger datasets. Unless otherwise stated, all our experiments use all 109 workers. Note that key conclusions from our experiments are likely to hold for a much larger setup with more workers over a larger geographic area. This is due to the spatial locality of workers and their completed tasks. Since workers tend to perform tasks within their proximity, CARROT can partition the space into smaller regions containing a small number of workers (like what we consider in our experiments) and assign tasks independently in each region.

B. Workers Characterization

We employed regularized logistic regression with Ridge penalty on the parameters. We had a collection of 45 to 60 sample answers per worker from our survey. We divided the samples to form training ($\%80$ of samples) and cross validation sets ($\%20$ of samples). Figure 3 shows the lease square error of the logistic regression cost function in Equation 1 for one worker. As can be observed on the figure, as λ_r increases, the training and cross validation error begin to converge. After finding a suitable λ_r corresponding to each worker, we applied a gradient regression based algorithm to find the characterization parameter θ for each worker. We implemented the solver using MATLAB fminunc Optimization Toolbox.

The resulting models were evaluated against workers actual behavior in Phases 2 and 3 of the user study. These predictions were compared with the actual outcomes of the task selection/assignment in Phases 2 and 3. The prediction error which indicates the error in percentage of completed tasks were 9.0% and 6.7% in Phases 2 and 3 respectively. One reason contributing to a higher prediction error in Phase 2 is the highly imbalanced distribution of tasks in that experiment.

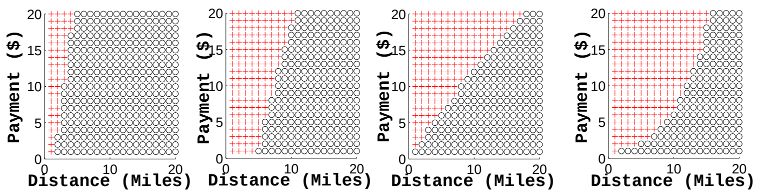


Fig. 2: Learned characterization model of 4 workers (ω_1 to ω_4). The plot shows decision outputs for given payment and distance of an offered task. The red (plus) signs indicate workers complete the offer with a probability of more than 0.5. The black circles indicate they complete it with a probability of less than 0.5.

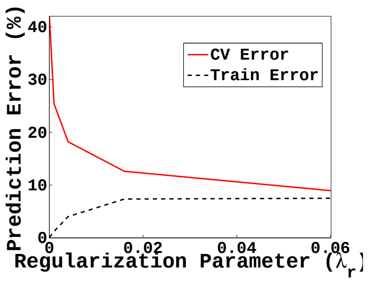


Fig. 3: Prediction error rate for different regularizing parameters λ_r (see equation 1).

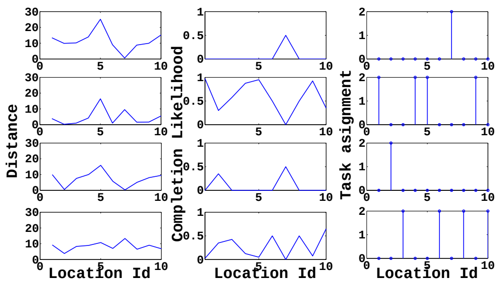


Fig. 4: Left: distances of workers ω_1 to ω_4 from task centers before assignment. Middle: likelihood of task assignment to each worker. Right: task assignment results. In this example, co-located tasks are automatically assigned to the same worker.

In phase 2, one single worker selected more than 50% of all tasks. However, he was not able to complete all of them before the assigned deadline. This observation is another indicator that workers performance is submodular with respect to the time they spend on the assigned tasks. In phase 3, tasks were distributed more evenly, as we limited the maximum number of assigned tasks to a person to 8.

Figure 2 shows the learned likelihood from four of the participants (shown by ω_1 to ω_4) in our user studies as a function of distance and payment parameters only. The red area (plus signs) is the region where the likelihood of completing an offer with a given distance and payment is more than 0.5, and the black area (circles) is the region where the likelihood is less than 0.5. It can be seen that workers characteristics can be considerably different. For example worker ω_2 is always preferable over ω_1 in terms of demanded payments. He is also preferable over ω_4 for near tasks, however, as the task distances increase, ω_4 becomes a better choice for task assignment. Our algorithms leverage this diversity for task assignments.

Example Scenario. To demonstrate how Algorithms 1 and 2 work, we consider a simple scenario in which we want to assign 20 identical tasks to the above 4 workers. The tasks are located at 10 different locations (thus there are 10 pairs of co-located tasks). The budget for each task is \$5. We set a constraint for assigning at most 8 tasks to each worker (this can be easily changed to other constraints such as the aggregated duration of all tasks for each worker). In Figure 4, the left column shows the initial distances of each worker from each location. The middle column shows the initial probability of task assignment for each worker and each task. The right column shows the results of task assignment. Workers ω_1 and ω_3 are each assigned to 2 tasks and workers ω_2 and ω_4 are assigned to 8 tasks. Since all the tasks are similar in this example, distance becomes the most important parameter and we see that co-located tasks are assigned to the same person. Note that in general, task Bundling is done automatically by our algorithms, given the context (payment, distance, complexity, duration, etc.) we train our model with.

Now we run Algorithm 2, for the same task budgets, but we vary λ (Algorithm 2) to observe the trade-off between task completion rate and budget spendings. The difference is that once a worker has been assigned to a task, his relative location with respect to other tasks change and the likelihood that he does more tasks in the same area (for possibly lower payments) increases. In Figure 5, we can see that as λ increases, both average task payments and likelihood of completion reduces. With $\lambda = 0.1$, the completion rate becomes 21% less than $\lambda = 0$, however there will be a 24% savings in budget. This trade-off enables a more efficient budget allocation with respect to the availability and characteristics of workers and the urgency of completion of tasks.

C. *Evaluating Task Packaging Optimizations in Algorithm 1*

To evaluate the effectiveness of the optimizations in Algorithm 1, we compared it to two other methods for task assignment. The first task assignment method that we considered was the first come first serve (FCFS) method. FCFS is a very common method that is practiced in the existing mobile crowdsourcing platforms. In this method, as it appears from its name, workers log into the system in order and lock the available tasks they are interested in doing. This method totally ignores the potential workers that are inactive at the time of task assignment. The second method which we refer

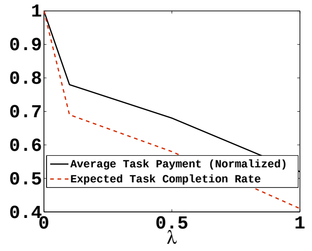


Fig. 5: Normalized task payments, and expected task completion rate as a function of λ (Algorithm 2). When $\lambda = 0$, all the payments are equal to task budgets.

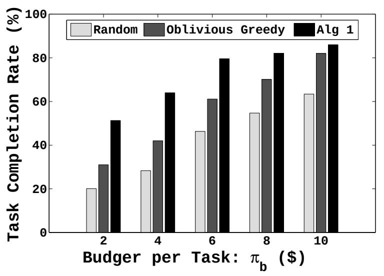


Fig. 6: Total payments for a test set of tasks: first come first serve (FCFS), oblivious greedy, and Algorithm 1.

to as Oblivious Greedy (OG), which unlike FCFS considers matching a group of workers and tasks together by exploiting context. However, in OG, each task is assigned greedily to the best available match (worker), without considering a global objective such as the one in Algorithm 1.

To compare the above methods, we simulated a scenario with 500 tasks (100 real business locations that were used in the user study, and 5 tasks per location). The workers profiles were generated as discussed earlier. It was assumed that each worker would select at most 10 tasks. Figure 6 shows the expected percentage of completed tasks for the three methods for different budgets (π_b) per task. We ran each method 10 times, and derived the expected value of task completion rate by averaging the outcomes. As shown, CARROT can achieve up to $2\times$ better completion rate than FCFS for a smaller budget (e.g., 2$). From the results we can see that for lower budgets, the performance gap between the three methods is wider. In that case, Algorithm 1's performance is much more effective due to the global optimizations in task assignment and utilizing more effective workers. As the budget increases, the performance gap becomes narrower. This is expected since more workers would do tasks for higher payments, and their diversity becomes less apparent.

D. Evaluating Budget allocation-Task Completion Trade-off

We now evaluate the efficiency of Algorithm 2. We assume the same simulated worker population and task assignment scenario as discussed above. Figures 7a and 7b show the average payments per task and task completion rates as a function of the optimization parameter λ respectively. As shown, for a reasonably large maximum budget (10$), CARROT can reduce the total spending by more than 50% at the cost of reduced completion rate of only $< 15\%$ ($\lambda = 0.2$). Compared to the baseline FCFS, this gives a $\approx 50\%$ faster completion rate, with only half the budget. When $\lambda = 0$, there would be no payment optimization and tasks payments would be the exact same as the budgets. As λ increases, both payments and completion rates decline as expected. However, it can be seen that for the same average task payment, the completion rate is slightly higher when the task budget is more. This can be explained due to the fact that Algorithm 2 becomes more flexible in allocating and distributing payments if more budget is available.

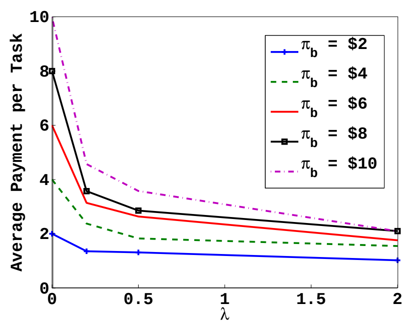
E. Sensitivity Analysis

CARROT takes into account workers' statistical behavior for assigning tasks. As we discussed earlier, we can keep track of variations in workers behavior by continuously updating the learned parameter θ for each worker as we collect new context and response from them. In this section, we evaluate how workers behavior variations affect the outcome of task assignment. To do so, we modeled the variations in the likelihoods P_{τ_ω} according to a general normal distribution $\mathcal{N}(\mu, \sigma^2)$, where $\mu = P_{\tau_\omega}$. We ran Algorithm 1 on the same simulated task and worker scenario as discussed above and measured the output for 2 different σ values. Figure 7c shows the Mean Square Error (MSE) of task completion rate where the normalized variation is 5% and 20%. We can see that variations in predicted likelihoods only moderately change the task completion likelihood results. For higher budgets, the results are effected less by worker behavior changes.

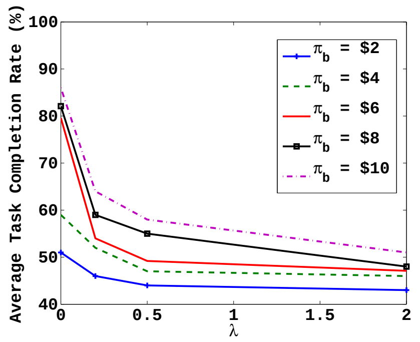
VII. Related Work

Several recent systems and applications have demonstrated values of mobile crowdsourcing. Systems such as mCrowd [19] and SensoRcivico [9] provide a generic platform to enable various crowdsourcing tasks. Several applications used crowdsourcing for, e.g., detecting potholes [1], mapping noise [2] and pollution [3] in urban areas, monitoring road traffic [4], [5], collecting speech data [6], measuring personal environmental impact [20], etc. Following the success of these research prototypes, several instances of paid mobile crowdsourcing markets have also emerged commercially including Gigwalk, FieldAgent, and TaskRabbit. Typical tasks in these markets pay workers a few dollars for capturing photos of buildings or sites, price checks, product placement checks in stores, traffic checks, location-aware surveys, and so on. CARROT focuses on core problem of all these systems: assigning or recommending tasks to workers. Both paid and voluntary crowdsourcing systems can readily incorporate CARROT algorithms.

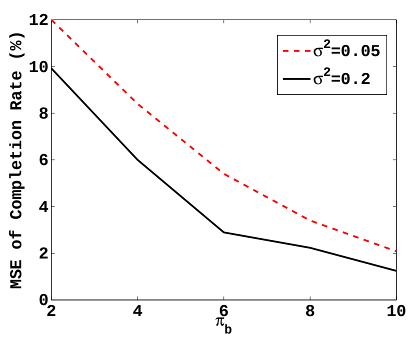
Several recent works have focused on designing algorithms to optimize various aspects of crowdsourcing operation. One line of research designs incentive mechanisms to better motivate workers to complete their tasks. In [13], authors develop incentive mechanisms for maximizing utility of the platform and the workers. Lee et al. [21] propose a reverse auction-based dynamic pricing algorithm and show that using dynamic price can reduce the incentive cost compared with fixed price. [22] proposes a greedy algorithm based on the recurrent reverse auction incentive mechanism, which aims to maximize the covered area under a budget constraint. In contrast to these algorithms, CARROT learns workers' desired incentive for



(a)



(b)



(c)

Fig. 7: Average task payment (a.), and completion rate (b.) as a function of λ and task budget π_b in Algorithm 2. (c.) Sensitivity analysis as a function of variations in workers behavior. The variation is modeled by a normal distribution with different σ's.

different tasks from their history and exploits the information during optimization of task recommendation.

Recent works have also considered using optimization algorithms with various goals. For example, SensoRcivico [9] aims to maximize conditions for worker participation, while minimizing the usage of energy. GeoTruCrowd [7] maximizes the number of spatial tasks that are assigned to a set of workers while satisfying the confidence levels of those tasks. WST [8] maximizes the number of worker's self-selected tasks in spatial crowdsourcing. CARROT's optimization algorithms differ from existing algorithms in three key aspects: (1) their optimization goals, (2) using the idea of packaging tasks, which is extremely valuable as shown by real-world studies [11], and (3) the ability to consider context in general, rather than being specific to location only (SensoRcivico can handle general context as well).

VIII. Conclusion

We proposed CARROT, a suite of algorithms for optimizing task recommendation in a mobile crowdsourcing system. Unlike existing systems that list individual tasks for workers to select, CARROT automatically identifies and recommends to workers packages of tasks. Our experiments with real datasets demonstrate that, compared to a baseline algorithm, CARROT can improve task completion rate by up to $2\times$ with the same budget and $1.5\times$ with almost half the budget.

References

[1] A. Mednis, G. Strazdins, R. Zviedris, G. Kanonirs, and L. Selavo, "Real time pothole detection using android smartphones with accelerometers," in *DCOSS*, 2011.

[2] R. K. Rana, C. T. Chou, S. S. Kanhere, N. Bulusu, and W. Hu, "Ear-phone: an end-to-end participatory urban noise mapping system," in *IPSN*, 2010.

[3] M. Stevens and E. DHondt, "Crowdsourcing of pollution data using smartphones," in *Workshop on Ubiquitous Crowdsourcing*, 2010.

[4] A. Thiagarajan, L. Ravindranath, K. LaCurts, S. Madden, H. Balakrishnan, S. Toledo, and J. Eriksson, "Vtrack: accurate, energy-aware road traffic delay estimation using mobile phones," in *SenSys*, 2009.

[5] P. Mohan, V. N. Padmanabhan, and R. Ramjee, "Nericell: rich monitoring of road and traffic conditions using mobile smartphones," in *SenSys*, 2008.

[6] J. Freitas, A. Calado, D. Braga, P. Silva, and M. Dias, "Crowdsourcing platform for large-scale speech data collection," *Proc. FALA, Vigo*, 2010.

[7] L. Kazemi, C. Shahabi, and L. Chen, "Geotrucrowd: Trustworthy query answering with spatial crowdsourcing," in *ACM GIS*, 2013.

[8] D. Deng, C. Shahabi, and U. Demiryurek, "Maximizing the number of worker's self-selected tasks in spatial crowdsourcing," in *ACM GIS*, 2013.

[9] A. Tamilin, I. Carreras, E. Ssebaggala, A. Opira, and N. Conci, "Context-aware mobile crowdsourcing," in *UbiComp*, 2012.

[10] M. Musthag and D. Ganesan, "The role of super agents in mobile crowdsourcing," in *Workshops at AAAI*, 2012.

[11] ——, "Labor dynamics in a mobile micro-task market," in *CHI*, 2013.

[12] N. Kokkalis, J. Huebner, S. Diamond, D. Becker, M. Chang, M. Lee, F. Schulze, T. Koehn, and S. R. Klemmer, "Automatically providing action plans helps people complete tasks," in *AAAI Workshops*, 2012.

[13] D. Yang, G. Xue, X. Fang, and J. Tang, "Crowdsourcing to smartphones: incentive mechanism design for mobile phone sensing," in *Mobicom*, 2012.

[14] J. Vondrák, "Optimal approximation for the submodular welfare problem in the value oracle model," in *STOC*, 2008.

[15] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[16] M. R. Garey and D. S. Johnson, *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., 1990.

[17] E. Kamar, E. Horvitz, and C. Meek, "Mobile opportunistic commerce: mechanisms, architecture, and application," in *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 2*. IFAAMAS, 2008.

[18] B. Lehmann, D. Lehmann, and N. Nisan, "Combinatorial auctions with decreasing marginal utilities," in *Proceedings of the 3rd ACM Conference on Electronic Commerce*, ser. EC, 2001, pp. 18–28.

[19] T. Yan, M. Marzilli, R. Holmes, D. Ganesan, and M. Corner, "mcrowd: a platform for mobile crowdsourcing," in *SenSys*, 2009.

[20] M. Mun, S. Reddy, K. Shilton, N. Yau, J. Burke, D. Estrin, M. Hansen, E. Howard, R. West, and P. Boda, "PEIR: the personal environmental impact report, as a platform for participatory sensing systems research," in *MobiSys*, 2009.

[21] J.-S. Lee and B. Hoh, "Dynamic pricing incentive for participatory sensing," *Pervasive and Mobile Computing*, vol. 6, no. 6, pp. 693–708, 2010.

[22] L. G. Jaimes, I. Vergara-Laurens, and M. A. Labrador, "A location-based incentive mechanism for participatory sensing systems with budget constraints," in *PerCom*, 2012.