Worker Selection for Reliably Crowdsourcing Location-Dependent Tasks

Kevin Emery(✉), Taylor Sallee, and Qi Han

Department of Electrical Engineering and Computer Science,
Colorado School of Mines, Golden, CO 80401, USA
{kemery,qhan}@mines.edu, taylorsallee@gmail.com

Abstract. Obtaining accurate information about specific locations is of great importance to today's many crowdsourced smartphone applications. To verify information about a location, smartphone users are selected to go to the location and answer a yes/no question about the location. Our research focuses on the location-aware worker selection problem, which is the problem of selecting a group of workers who, together, can give the most accurate answer to the location-based question. We define the location-aware worker selection problem, mathematically formulate it, and then show that an optimal solution is exponential in time complexity. We present our heuristic solutions that take into account both the reliability of the users and the level of convenience for each user to complete the task. We evaluate and compare our approaches to three other heuristic algorithms via simulation.

Keywords: Crowdsourcing · Mobile sensing · Worker selection

1 Introduction

As the number of smartphone users worldwide continues to grow, the use of crowdsourcing applications has become increasingly prevalent. One important problem for many crowdsourcing applications is data verification; that is, when smartphone users provide information to a crowdsourced application, the data is not guaranteed to be accurate. One solution to this problem is to select users that are both willing and reliable enough to provide accurate information.

Many of today's mobile applications rely heavily on accurate location data. Take, for example, a coffee shop locator application that allows users to locate the nearest/cheapest/best coffee shop. In an app such as this, the quality of the user experience depends on the accuracy of location information. Many other applications benefit from knowing exactly where certain restaurants, shops, offices, etc. are located. Our research applies to two classes of location-based problems: one is location verification, where we believe a specific business is at a specific address, but we are not certain about the veracity of the information; and the second is information verification, where we believe a certain event is happening at an address (or a specific person is there, etc.), but we need to confirm the information.

S. Sigg et al. (Eds.): MobiCASE 2015, LNICST 162, pp. 71–86, 2015.
DOI: 10.1007/978-3-319-29003-4_5

Both of these classes of applications involve simply verifying information (i.e. they can be phrased as a yes/no question). Both classes also require someone to physically go to the address to answer the question. Crowdsourcing can be of great help in these applications. We propose to ask individuals near the address to physically go to the address to answer a question. Our research attempts to solve the "location-aware worker selection problem", which is essentially the question of finding the best individuals to select in the interest of answering a location-dependent question.

Selecting a group of workers to answer a question about a physical location comes with a set of unique challenges. First, to motivate people to respond to location-dependent queries, incentives need to be used. However, in the real world there is always a budget that limits the amount of money available to pay users as an incentive. We must choose a group of workers to pay in order to get the best possible answer to the location-based question, while staying under the budget constraint. Second, users are unlikely to respond to a query unless they are near the location in question. For this reason, we must consider workers' current locations and commute patterns to help us select the best workers. Finally, as with any worker selection problem, we must consider the reliability of our users and their responses and find a way to judge the accuracy of responses.

Problem Definition. We assume each worker is associated with a set of attributes: reliability rating, incentives needed, home location, work location, and current location. The reliability rating is a simple way to determine a worker's past response record. We use the home and work locations to determine a commute route for the worker, and this information is useful in determining how convenient completing the task would be for each worker. Informally, the location-aware worker selection problem can be defined as follows: given a yes or no question about a specific location, a set of workers that are reasonably near the location, and a budget to spend on incentives, find a subset of workers to dispatch to the location such that the answer given by the group as a whole is as accurate as possible.

Existing works have looked at worker selection, but as of yet, none have considered the location-aware worker selection problem, which is distinct because it requires workers to physically go to the location in question to perform a task. In addition, most previous works have only considered worker reliability in their selection process. Because our work is specific to physical locations, we consider a convenience factor for each worker. The addition of convenience to the problem adds additional complexity. This work makes the following contributions:

- We mathematically define the objective function that serves as a metric for evaluating solutions to the location-aware worker selection problem.
- We theoretically analyze the time complexity of an optimal solution for the problem.
- We propose RECON and CORE, two heuristics that seek to choose only workers that are both highly reliable and can respond to queries with a high level of convenience.

– We carry out extensive experiments to evaluate our proposed algorithms, comparing them to two other algorithms and showing that RECON out-performs others.

2 Related Work

Much research has already been done on various versions of the worker selection problem. Some call this problem the task assignment problem. Both of these problems involve distributing a set of tasks among a set of workers, hopefully in a way that gives the tasks to the users best-suited to perform them. Different worker selection algorithms have different goals. Some seek to maximize the reliability of the workers (this concept is explored in-depth in [8]), while others seek to minimize budget as in [5]. Our work is focused on maximizing the accuracy of the answers given by our selected workers. We have no time constraint, but we do have a budget constraint.

Many previous works focused on online crowdsourcing markets, such as Amazon's Mechanical Turk [1]. These online markets allow requesters to post tasks (sometimes called Human Interaction Tasks, or HITs) on the market, and then workers select the tasks they wish to perform. Many previous works focus specifically on efficient task allocation in online markets. For example, [4] attempted to allocate workers to tasks in a way that maximizes the benefit to the requester, and tested their algorithm on Mechanical Turk. In [2], the authors performed an experiment on CrowdFlower to test their CRITICAI algorithm, which determines the best human workers to assign a set of tasks to ensure reliable results while meeting real-time constraints.

Selecting reliable workers to answer a question is one step towards solving the truth estimation problem. The truth estimation problem is simply the problem of estimating the ground truth from a crowd-provided answer. Papers such as [3] discuss the challenges involved with resolving the truth in a crowdsourced application. When estimating the truth, choosing reliable workers will increase the probability of the majority vote being the ground truth. [9] used a streaming approach to solve the truth estimation problem. [7] studies a slightly different version of the truth estimation problem, which they call the "fact-finding problem". Both of these papers assume that workers give answers of unknown reliability, and finding the truth from the many answers may be a little more complex than simply using the majority vote.

Both the worker selection/task assignment problem and the truth estimation/fact-finding problem assume that more reliable workers will provide better responses, and that their responses can be trusted with a high level of confidence. None of the works so far have considered the convenience for the worker to actually perform the task. This paper shows that we can achieve better results when the convenience to the worker is considered in the algorithm.

None of the previous works mentioned have studied the location-based worker selection problem. They assume an online pool of workers that can perform tasks from their computer, without the need to travel and actually observe a physical

location. Adding a physical location to the problem introduces additional complexity. Our work is unique in that it considers a worker's location, commute patterns, and historic reliability, and in that it requires workers to actually go to a physical location to perform the assigned task.

3 Our Approaches

To better formulate the location-aware worker selection problem, we describe the models we use to define and solve the problem.

3.1 Definition of Accuracy Metric

Since we rely only on the selected workers to determine the ground truth of our questions, we use a majority voting scheme to determine the "correct" answer to the location-based question. Other methodologies can be used to determine ground truth [3,7,9], but majority voting is the most common. In order to judge the correctness of any solution to this problem, a metric must be used. We propose to use the following scheme: given a set of workers, W, consisting of users who have provided an answer to the location-related question, the metric used to judge correctness is defined as the ratio of the number of users who agree with the majority vote over the total number of users in W. For example, if W contains 100 workers, and 78 of them agree that the answer to the given question is yes, then the metric will have the value .78, or 78 %. This implies that we seek to maximize the number of users who agree, because our goal is to be as sure as possible that the group has provided a correct answer. We call this metric the Majority Agreement Percentage (MAP).

We use pc_i to denote the probability of user i providing a correct answer for a given question. If each worker's pc_i value was the same (pc) the distribution of MAP values for different tasks would be a binomial distribution centered at pc. Because each user may potentially have a different value for their individual pc_i, a binomial distribution cannot be used. However, a binomial distribution is simply a special case of the more generic Poisson binomial distribution [10]. In a Poisson binomial distribution, each event in the model can be considered to have a unique probability, exactly like what we are dealing with when determining MAP(W). Furthermore, because a poisson distribution is a normal distribution, the piece of the distribution that shows the most likely response for a group W is the mean. For a Poisson distribution, the mean value of the distribution is simply the sum of all of the individual probabilities that make up the distribution divided by the total number of probabilities. We now formally define the probabilistic formulation of MAP: $\text{MAP}(W) = \sum_{i=1}^{\|W\|} (pc_i)/\|W\|$.

3.2 Formal Problem Statement

Given a location L, a yes or no question Q about the location, a budget B to spend on incentives, a global set of users $U = \{u_1, ..., u_n\}$ where user i needs

incentive I_i to go and find out the answer to the question and has a probability pc_i of responding correctly, find a set of workers $W = \{w_1, ..., w_m\}$ that contains at least m workers such that after all users in W have provided their answers to Q, MAP(W) is maximized. This is formulated mathematically as follows:

Maximize MAP(W)
subject to
1. $W \subseteq U$.
2. $\|W\| \geq m$
3. $B \geq \sum_{i=1}^{\|W\|} I_i$.

The first constraint basically requires that W must be a subset of U. This must always be true; we cannot have workers that are not part of the original user set. The second constraint is due to the fact that if the system was deployed in the real world, the person seeking the answer to the question would want to have a certain number of respondents in order to be confident in the response that they get. This constraint ensures that the set W containers at least as many workers as their minimum threshold specifies. The third constraint states that the sum of the incentives I_i paid to each worker in W must not exceed the budget B.

Time complexity analysis of an optimal solution In order to find the optimal solution to this problem, one must look at all possible subsets W where $\|W\| \geq m$. If we simplify that statement to only look at cases where $\|W\| = m$, then we need to look at $\binom{n}{m} = \frac{n!}{m!(n-m)!}$ different subsets. This can be expanded to $\frac{(n)(n-1)(n-2)...(n-m+1)(n-m)!}{m!(n-m)!}$ when $m \leq n/2$, and $\frac{(n)(n-1)(n-2)...(m+1)(m!)}{(n-m)!m!}$ when $m > n/2$, and from there they can simplified to $\frac{(n)(n-1)(n-2)...(n-m+1)}{m!}$ when $m \leq n/2$, and $\frac{(n)(n-1)(n-2)...(m+1)}{(n-m)!}$ when $m > n/2$. The reason for the split on $m = n/2$ is because when $m \leq n/2$ the $(n-m)!$ term is larger than the $m!$ term and therefore it is more efficient to cancel that term, while the opposite is true when $m > n/2$. The case where $m \leq n/2$ has a time complexity of $O(n^m)$ and the case where $m > n/2$ has a time complexity of $O(n^{n-m})$.

In order to look at the time complexity required to consider all possible subsets where $\|W\| \geq m$, it is just the summation of all of the discrete time complexities used for each value of $\|W\|$. The case where this is largest is where $m = n/2$, and this has a time complexity of $O(n^{n/2})$ which simplifies to $O(n^n)$. The optimal solution, therefore, has an exponential time complexity and cannot be solved in a reasonable amount of time. For this reason we propose our own heuristic algorithms.

3.3 Our Algorithms

When considering which users to choose to dispatch to a location, it is useful to know how reliable each user is. In other words, it is helpful to know their previous

answer history. For this reason, we assume each user has a reliability rating, r_i. In addition, when considering which users to choose to answer the location-based question, it is prudent to choose workers close to the location in question. We go a step further and consider the overall convenience for the worker to respond, based on their normal movement patterns and current location. This is useful because users are more likely to respond to a task if it is very convenient for them. For each location-dependent task, we calculate a convenience factor, c_i, for each user that captures the convenience of completing the task. Note that while a user's reliability rating is an attribute of the user and independent of any single task, a user's convenience factor must be calculated for each task.

Model of User Reliability. We use the following scheme to track user reliability, r_i. We first assume each user u_i begins with a reliability rating of 0.5. When a task is created, a group of workers W is selected to perform the task. After getting responses from each of the selected workers in W, majority voting determines which workers in W gave the correct answer. Each worker's reliability is then dynamically adjusted as follows:

- If a worker in W gave a correct answer, he is awarded a reliability of 1 for that task.
- If a worker in W gave an incorrect answer, he is awarded a reliability of 0 for that task.
- The worker's new reliability rating is the average of all the reliability ratings for all tasks he has performed.

For example, a new worker who starts with a rating of 0.5 and is chosen to perform one task will have a new rating of 0.75 if he answers the task correctly, and a rating of 0.25 if he answers incorrectly. We can also take into account the case when a user does not respond, which does not fundamentally change our approach. Our reliability model can also be replaced with another such as [9].

Model of User Convenience. We use the following scheme to calculate the convenience factor, c_i, for a given user to perform a given task. This factor will be calculated for each location-based question. We assume each user u_i has a current location, a work location, and a home location. We then calculate the diameter of the city in question (the distance from top left to bottom right corner). The convenience factor c_i is then calculated for each worker as follows:

- Calculate the distance from the user's current location, d_c.
- Calculate the distance from the user's work location, d_w.
- Calculate the distance from the user's home location, d_h.
- Calculate the shortest distance from any point along the user's commute path, d_p.
- Calculate min = MIN(d_c, d_w, d_h, d_p).
- Calculate c_i = 1 - min/diameter

The convenience factor is clearly based on both the user's current location and their normal commute patterns. Note that because the convenience

takes into account the diameter of the city in question, convenience values are always between 0 and 1, which normalizes convenience factors across different-sized cities. We acknowledge that this is a simplified model of user convenience. A more realistic model is to incorporate existing work studying people's commute or mobility patterns using either synthetic models or realistic traces, as well as map information.

Model of Incentive Scheme. As mentioned, we pay users an incentive to complete tasks. The sum of the incentives paid for a given task may not exceed the budget for that task. We use a monetary incentive, but this could easily be replaced by another type of incentive.

Choosing which users to pay and how much to pay them should be based on the probability that they will provide a correct response. Therefore, we pay users with a higher reliability rating more than we would a user with lower reliability. However, we must also consider the convenience for the user. If a user is very far away (has low convenience), we want to provide a larger incentive, so they do not feel like responding is a waste of time. We use the following incentive scheme that takes into account both user reliability and convenience: $I_i = k_1 * r_i + \frac{k_2}{c_i}$. This equation increases a user's incentive if they have a high reliability ranking and/or if they have a low convenience factor. This makes sense in a real-world situation because users who are further away (i.e., have low convenience) are less likely to give good answers, and users with higher reliability are more likely to give good answers.

Proposed Heuristics. Considering the NP-completeness of the problem, we propose two heuristics.

– RECON (REliable and CONvenient). For each location-based question, it first creates a group of high-convenience workers, with every worker in the group having a convenience factor no less than a certain threshold α. The value of α controls how selective RECON will be in choosing workers for W. A higher α means RECON chooses workers that are very close to the location in question, which means the incentives paid to those workers will be smaller, allowing the size of W to be larger. The algorithm then sorts the group in order of reliability rating. It then selects users for W in order of highest reliability to lowest, until the sum of the incentives paid to the users is as large as possible without exceeding the budget.

  RECON avoids picking users that have high reliability but low convenience, thereby improving the probability of the users in W providing a quality response. This algorithm chooses the users that are both highly reliable and conveniently located, so we can be confident they will produce an accurate response. Also, this algorithm is able to choose more users for each W, because users with a higher convenience are paid less than those with low convenience, assuming their reliability ratings are equal. Selecting more users for W is good, because it makes MAP(W) more precise. The larger $\|W\|$ is, the more confident we are that MAP(W) is a trustable number.

RECON has a complexity of $O(n + nlog(n))$, where n is the size of the user group U. The $nlog(n)$ piece comes from sorting the workers by reliability rating, and the additional n comes from walking through U and selecting only workers with a convenience greater than α.

– CORE (COnvenient and REliable). This heuristic is based on a similar model as RECON, but instead of focusing on selecting workers for W that have a high reliability rating, CORE instead focuses on using workers for which the location is most convenient. A subset of workers that have a reliability above a certain threshold β are selected, and then from those workers those with highest convenience are selected for W.

  Similar to RECON, CORE has a computational complexity of $O(n+nlog(n))$, where n is the size of the user group U, because the users still need to be sorted in $nlog(n)$ by convenience and then in the worst case we need to look at all n workers to fill W.

4 Performance Evaluation

To evaluate our heuristics to the location-aware worker selection problem, we built a simulator in Java. We decided to use a simulator instead of a real-world deployment mainly because of the logistics involved in getting actual workers. We could not use Mechanical Turk or CrowdFlower, because these services are just people sitting at their computer, and the workers are not expected to leave their computer to go visit a location. We envision our algorithms being used for commercial crowdsourcing applications, where the workers would be users with a mobile app installed, and the location-based questions are for verifying that a certain business exists at the location in question, or a certain event is happening there.

4.1 Simulation Setup

The simulator takes a city (consisting of many locations and workers) and an algorithm as input. The simulator runs the given algorithm on every location in the city, pulling workers to populate W from the many workers within the city. Over time, the reliability ratings of the users change, since some users give good answers and some give bad answers.

The constants k_1 and k_2 used in the incentive mechanism are to change the weight of each factor, and we imposed the following limit on k_1 and k_2: $k_1 + k_2 = 10$. After running a few preliminary tests, we settled on the values $k_1 = 7$ and $k_2 = 3$. This means that we gave more weight to reliability in deciding the incentive, but give the convenience term the ability to grow indefinitely as the c_i value gets smaller and smaller. Basically, the smallest incentive a user can get is 3 cents (when $c_i = 1$ and $r_i = 0$, and although the largest incentive is uncapped (because c_i is in the denominator), our algorithms basically never

choose a user with really low convenience, because the incentive for that user would be too high.

Our goal is to maximize the average MAP value for each group of selected workers W. In a real-world situation, we would know which workers gave correct responses once they had all responded. In our simulator, we need to model pc_i. In a real-world situation, the value of pc_i for each user would be unknown; however, in our model we assume that both convenience and reliability affect the worker's ability to respond correctly. The probability that each user chooses correctly for a given task is be determined as this: $pc_i = (1 - k_3 * (1 - c_i)) * (r_i)$. This equation primarily uses user reliability to determine pc_i, but it also takes into consideration the possibility that the user's convenience factor is low, which could potentially cause them to go to the wrong location and provide the wrong information. The frequency with which this occurs is accounted for in the constant k_3, and the value of this constant can be adjusted to add more weight to the convenience factor. For our simulations, we settled on a k_3 value of 1/8. k_3 essentially represents how often the user gets lost, is unable to find the given location, or gets confused as to where they are and gives the wrong answer.

After a few preliminary tests of RECON, we set α to be 0.7, because we noticed that anything below 0.7 allowed too many low-convenience workers into the group. Similarly, after testing CORE the value for β was set to be 0.45. This allows new workers to be included in the chosen group, but if a worker provides more incorrect responses than correct responses then they are excluded from all future selections of W.

We test the performance of five different algorithms with the last three as baseline comparison. Each algorithm uses the aforementioned incentive scheme to determine how much each user will be paid for a specific task. The algorithms are listed below:

- RECON: An algorithm that sorts workers by reliability and then selects workers for W from highest reliability to lowest so long as the worker's convenience is greater than α
- CORE: An algorithm that sorts workers by convenience and then selects workers for W from highest reliability to lowest so long as the worker's convenience is greater than β
- RELIABLE: An algorithm that sorts the users in order of reliability rating, and selects workers for W from highest reliability to lowest, until the maximum number of workers have been selected while still under the budget constraint.
- CONVENIENT: An algorithm that sorts the users in order of convenience factor, and selects workers for W from highest convenience to lowest, until the maximum number of workers have been selected while still under the budget constraint.
- RANDOM: An algorithm that picks users randomly from the group U until the maximum number of workers have been selected while still under the budget constraint.

We assume each city contains a finite number of both locations and workers, with the number of workers being greater than the number of locations.

We wrote a city creator as part of our simulator. The city creator randomly distributes locations throughout a 2-dimensional space, and then randomly distributes workers throughout the same space. Each worker is assigned a permanent home and work location. Figure 1 shows what a small portion of a generated city would look like. Locations are red squares, workers are blue circles, home locations are green pentagons, and work locations are yellow triangles. Note that some workers are at home, some are at work, some are between their home and work locations, and some are just randomly distributed. The probability model we used to decide the workers' current locations is described below.

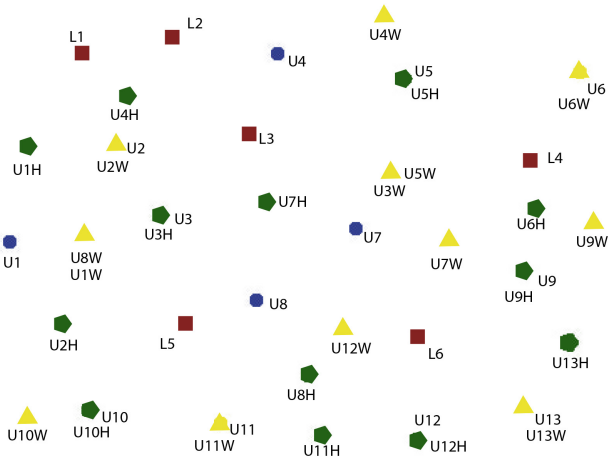

Fig. 1. A small portion of a city, showing 6 locations, 13 workers, and the home and work addresses of each worker (color figure online).

We tested each algorithm on three different cities, each with a different population density. The city width (1077 units), height (1077 units), and number of locations (1000) remained constant, with the worker population varying for each city. We decided to use this approach because the ratio of workers to shops for a given unit area is the most relevant factor for deciding the accuracy of a selection algorithm. For each city, we used three different budgets (100, 200, and 300 cents). We called these budgets low, medium, and high, respectively. The four city populations were 5K, 20K, 40K, and 60 K workers. We named these cities sparse density, low density, medium density, and high density, respectively. Therefore, every algorithm was run on 12 different city/budget combinations. For each of the 12 distinct city/budget combinations, we ran 20 trials for each algorithm. Once the four different-density cities were created, we did not change them for the rest of the testing. This ensured that we ran the different algorithms on the same cities every time.

The simulator walks through every location in the city (which are distributed randomly), and for each location, runs the given algorithm on that location. The algorithm selects a group of workers W, and the workers provide a

response, which is generated based on their pc_i value. The simulator then calculates MAP(W) for the group, records it, and moves on to the next location. As workers provide responses to the location-based questions, their reliability ratings are updated by the simulator (as previously discussed). Although the problem as defined only considers one task, any good algorithm will take into account the outcome of previous tasks, and seek to improve the MAP value of each task over time.

Simulating Worker Location. Before each location-based question is asked, each user's current location is calculated based on the following probability model:

- 32 % of the time, the worker is at work.
- 9 % of the time, the worker is commuting.
- 22 % of the time, the worker is away from home, but not at work or commuting. This is for leisure/social activities, travel, etc.
- 37 % of the time, the worker is at home.

We obtained these percentages from estimates we made based on the information in [6]. We assumed that the average person spent 60 hours per week sleeping, and so 108 hours in the week were available for them to respond to queries. Of those hours, we estimated 35 hours would be spent at work, 10 hours in commute, 23 hours away from home, and 40 hours at home. To determine the user's current location in our simulator, we generated a random number between 1 and 108, and chose the worker's current location based on the interval into which the random number fell.

4.2 Experimental Results

The following figures highlight the results of our simulations. Figure 2 shows a scatter of MAP values for all five algorithms, as tested on the high density city (60,000 workers) with the low budget (100 cents). Each point represents the average MAP value over 20 trials of the algorithm for one location. As more locations were processed, worker reliabilities were updated. Figure 3 shows best-fit lines for each algorithm using the high density city (60,000 workers) with the low budget (100 cents). Both of these figures show the MAP values for RECON and RELIABLE increasing as more locations are processed. Both stabilize at a very high percentage near the end, but RECON outperforms RELIABLE at almost every location processed. This is because RELIABLE's biggest flaw is that it does not consider convenience. Convenience is important because it affects the worker's likelihood of providing the correct response.

Figures 4 and 5 show similar trends for the medium (200 cents) and high (300 cents) budgets. As we can see, the CONVENIENT and RANDOM heuristics did not perform very well, always selecting a W with a MAP value of around 0.5, and this holds true for every city/budget combination we tested. While CORE performed better than CONVENIENT and RANDOM, it was not significantly

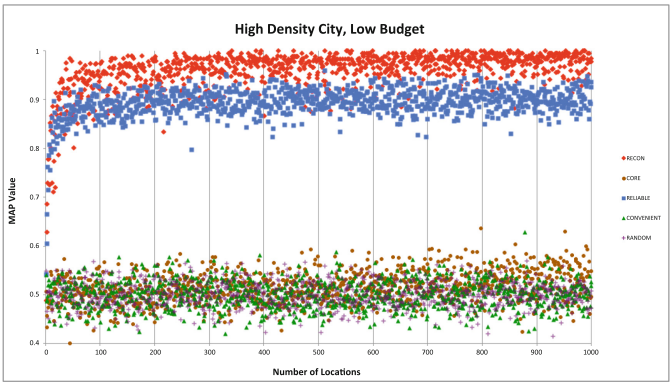

Fig. 2. Scatter plots of MAP values of all five algorithms over time, as the number of locations processed increases. The budget is fixed at $1, and the city population is 60,000.

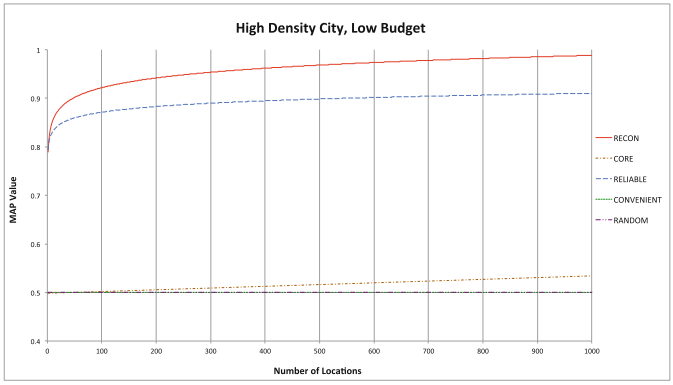

Fig. 3. Best-fit trend lines of MAP values of all five algorithms over time, as the number of locations processed increases. The budget is fixed at $1, and the city population is 60,000.

better than either algorithm. The biggest reason for this was because the threshold β cannot be set any higher than 0.5, otherwise a brand new worker would never be selected. For these reasons, we omit any further results for these three heuristics.

Figure 6 shows the MAP values, with 95 % confidence intervals, for each different budget. We show only the results from the high density city (60,000 workers), and the MAP value reported is the trend line value after processing all 1000 locations in the city. This is the most relevant point to report, because it shows how each algorithm performs after having ample time to update each worker's reliability rating. This plot shows that the confidence intervals for RECON at each tested budget are above those of RELIABLE when the algorithms have

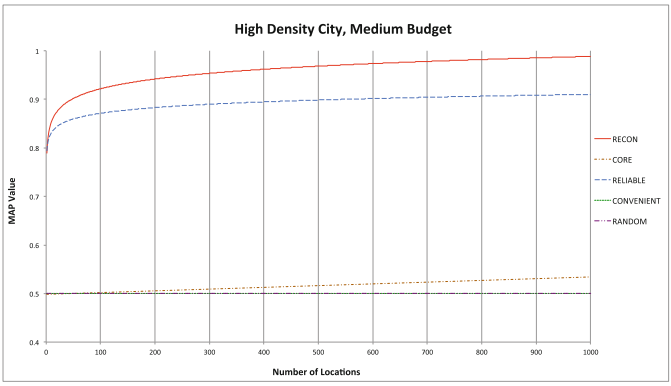

Fig. 4. Best-fit trend lines of MAP values of all five algorithms over time, as the number of locations processed increases. The budget is fixed at $2, and the city population is 60,000.

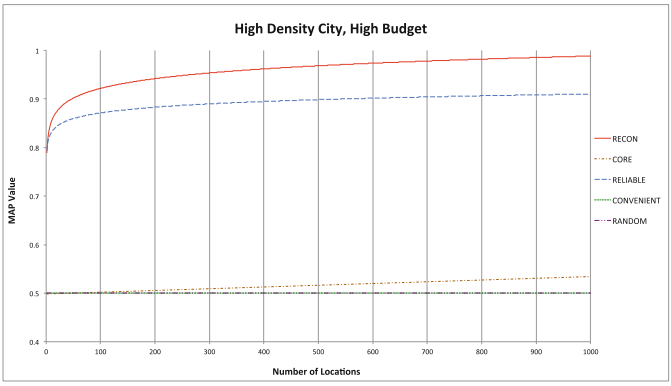

Fig. 5. Best-fit trend lines of MAP values of all five algorithms over time, as the number of locations processed increases. The budget is fixed at $3, and the city population is 60,000.

processed 1000 locations. This means that we can say, with 95 % confidence, that RECON outperforms RELIABLE at every budget we tested. The results from other cities are similar, with MAP consistently outperforming RELIABLE for every budget/city-density combination.

Figure 7 shows the MAP values, with 95 % confidence intervals, for each of the cities tested. We show only the results for the low budget (100 cents), and again we report the MAP value from the trend line, after processing all 1000 locations in each city. This plot shows that the confidence intervals for RECON at each tested population density are above those of RELIABLE when the algorithms have processed 1000 locations. This means that we can say, with 95 % confidence, that RECON outperforms RELIABLE at every population density

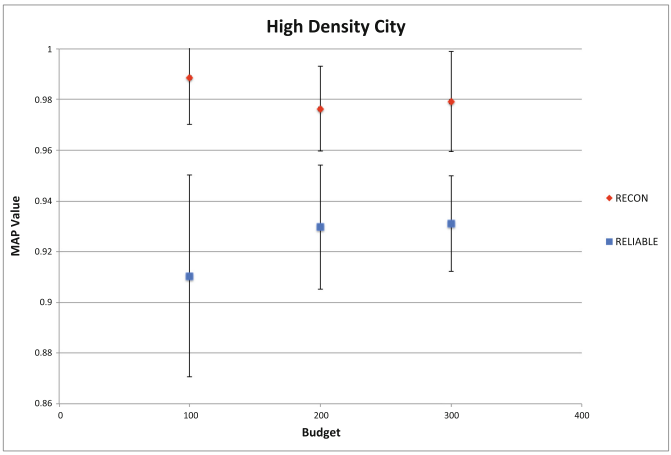

Fig. 6. Impact of budget on MAP value. City population is fixed at 60,000, and number of locations is 1000.

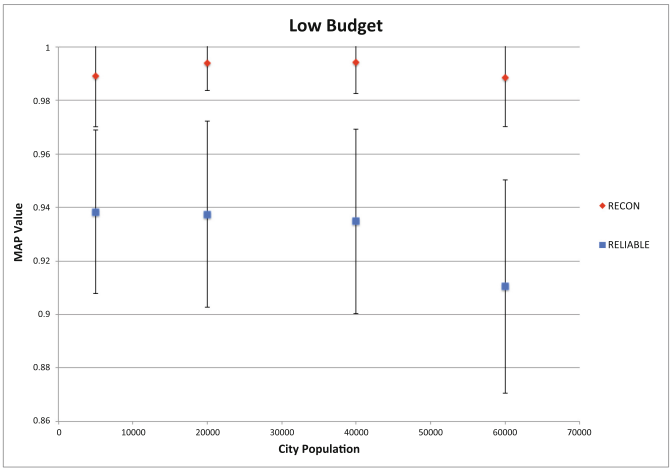

Fig. 7. Impact of population on MAP values. The budget is fixed at $1, and the number of locations is 1000.

we tested. The results from other budgets are similar, with RECON consistently outperforming RELIABLE for every budget/city-density combination.

5 Conclusions

As the number of smartphone users continues to rise, the use of crowdsourcing has become a standard practice for many applications. Getting quality information from the crowd can be a challenge, especially when the ground truth is

unknown. Selecting workers that are both reliable and willing to perform tasks is hard enough, but verifying the validity of a crowdsourced answer can be even harder. For all applications that depend on crowdsourcing today, data quality is of paramount importance.

The location-based worker selection problem is a new frontier in crowdsourced worker selection problems. Requiring users to actually go to a physical location adds a new dimension to the classic task assignment problem. With location-based tasks, incentives vary with each user's distance from the location in question. Consequently, considering a user's reliability is no longer enough when choosing the best workers to respond. We have introduced the location-aware worker selection problem, presented our algorithm, RECON, which considers worker reliability and convenience, and shown that RECON outperforms three heuristic algorithms for all of our test cases. RECON's advantage comes from its consideration of user convenience in addition to reliability, because it only selects workers with a high convenience factor. This allows more workers to be chosen for a given task and budget, thus improving the accuracy of the group's response.

Future Work. The results we have shown here give us confidence that RECON is a strong algorithm. In simulation, it is more effective than the RELIABLE approach for solving the location-based worker selection problem. The natural next step would be to test RECON in a real-world environment. We would like to create a smartphone application that pays users for answers to location-based questions, and deploy the app to as many users as possible in at least one city. However, we would first like to see how RECON performs with some different modeling methods. For example, we want to test using other methods than majority voting to determine the ground truth of worker answers. We would also like to use a more realistic commute pattern model, instead of simply using a straight line from the worker's home location to work location. We would like to test our algorithm on a model of a real city, using maps of roads, homes, and businesses. We would also like to try out different worker reliability models. Our simple model is not bad, but using a more sophisticated model may provide different results. Eventually we would design experiments to test RECON and RELIABLE's performance on real users and locations. Empirically showing that our model and algorithm is applicable to real-world problems would be very useful for many companies, organizations, etc. The end goal is to provide a robust service that allows anyone to pay a small amount of money to get highly reliable answers to location-based questions, without having to go out and verify the answers in person.

In the future (and even now), selecting workers to answer location-based questions will become a standard practice. Smartphone users can make a little money on their way home from work with just a couple taps on their device, and organizations can save huge amounts of money by outsourcing their data gathering to the crowd. Confidence in the collected data will be key, and RECON can be a first step in verifying the correctness of collected data.

References

1. Amazon. https://www.mturk.com/mturk/welcome
2. Boutsis, I., Kalogeraki, V.: On task assignment for real-time reliable crowdsourcing. In: Proceedings of IEEE 34th International Conference on Distributed Computing Systems, June 2014
3. Cox, L.P.: Truth in crowdsourcing. IEEE Secur.Priv. **9**(5), 74–76 (2011)
4. Ho, C., Vaughan, J.W.: Online task assignment in crowdsourcing markets. In: Proceedings of 26th Conference on Artificial Intelligence (2012)
5. Karger, D., Oh, S., Shah, D.: Budget-optimal task allocation for reliable crowdsourcing systems. Oper. Res. **62**(1), 1–24 (2014)
6. Bureau of Labor Statistics. http://www.bls.gov/tus/
7. Tejchman, J., Kozicki, J.: Experimental and Theoretical Investigations of Steel-Fibrous Concrete. SSGG, vol. 3, pp. 3–26. Springer, Heidelberg (2010)
8. Peer, E., Vosgerau, J., Acquisti, A.: Reputation as a sufficient condition for data quality on amazon mechanical turk. Behav. Res. Methods. **46**(4), 1023–1031 (2014)
9. Wang, D., Abdelzaher, T., Kaplan, L., Aggarwal, C.: Recursive fact-finding: a streaming approach to truth estimation in crowdsourcing applications. In: Proceedings of IEEE 33rd International Conference on Distributed Computing Systems, July 2013
10. WolframMathWorld. Poisson distribution. http://mathworld.wolfram.com/PoissonDistribution.html