CrowdSC: Building Smart Cities with Large Scale Citizen Participation

Karim Benouaret, Raman Valliyur-Ramalingam, François Charoy

CrowdSC: Building Smart Cities with Large Scale Citizen Participation

Karim Benouaret[1], Raman Valliyur-Ramalingam[2] and François Charoy[2]

[1]Inria Nancy – Grand Est, Villers-lès-Nancy, France
[2]LORIA/Inria/Université de Lorraine, Villers-lès-Nancy, France

Abstract – An elegant way to make cities smarter would be to design a platform where every citizen is given an opportunity to be effectively connected to the governing bodies in their location and to contribute to the general well being. In this paper, we present CrowdSC, an effective crowdsourcing framework designed for smarter cities. We show that it is possible to combine data collection, data selection and data assessment crowdsourcing activities in a crowdsourcing process to achieve sophisticated goals in a predefined context. We propose different strategies for managing this process. We also present an experimental study that evaluate outcomes of the process depending on these execution strategies.

1 Introduction

Nowadays, cities face complex challenges to meet objectives regarding urban development. As a consequence, pressure is growing for cities government bodies to leverage every available opportunity to become greener, smarter and to promote a better quality of life for their citizens. One of the opportunities is to foster citizen participation to allow them to contribute to the enhancement of their environment. This participation may occur as part of a crowdsourcing activity inside the delimited area of the city. What could be could be considered as infeasible in the past can be achieved at very affordable price today. More and more urban people are equipped with smartphones that provides advanced capabilities to connect to the internet, take picture and track their location[1]. It is now a well addressed idea to ask them to contribute in a given context the facilities of their device together with their intelligence to achieve large scale coordinated endeavour. It is realistic to try to benefit from citizens' knowledge, experience and collaboration to get an overview of the city infrastructure and utilities by crowdsourcing the result to questions such as : "What roads need a repair in Paris ?", or "What places are slovenly in Nancy ?".

Two main reasons motivate the need of citizens' participation in these and many other scenarios. First, in this kind of scenarios, information provided by humans (citizens) may be more accurate than that provided by sensors and computers. Second, it can be done on a lower budget since citizen participation is not directly rewarded.

Most of the work on crowdsourcing that can be found in the literature consider executing one single task at a time. Services, such as Amazon Mechanical Turk [2], are used to solve tasks that require human intelligence [3]. They are mostly used for atomic tasks; e.g., tagging images, evaluating products, etc. Here, we want to consider more complex queries that combine different kind of activities that can be decomposed in a series of small human tasks under the control of a process execution and that are context-aware.

As an example we imagine that the mayor of Nancy (A middle sized city from the North-East of France) wants to get a view on the state of the roads to schedule repairs. He may ask for citizens' contribution to help him to achieve that goal. For instance, he may ask the following query: "What roads need a repair in Nancy?". This query seems simple if you ask experts to do it with a precise specification. It is more tricky when you want ordinary citizen to help you. First the mayor must be specific. If hundreds of roads are reported as damaged, does all the roads really need a repair? How to find those that need a repair urgently ? Since smartphones are equipped with cameras the query can be changed to: "Find photos of roads that need a repair in Nancy, along with an assessment {not damaged, damaged, very damaged}". This query can be answered in a straightforward way, i.e., each participant send one or more photos of damaged roads along with their priority. With this procedure, we might receive a lot of photos with different priority assessment for the same locations (i.e., roads): some citizens make mistakes; some lie so that the roads in their vicinity will be repaired. We propose to overcome this problem with a more sophisticated decomposition of the process: *(i)* ask some citizens to provide photos of roads that need a repair. We refer to this step as *data collection (ii)* ask some other participants to select the most representative photo for each location. This step is referred to as *data selection*; and *(iii)* ask some citizens to assess the priority of each selected photo. We refer to this step as *data assessment*. Unfortunately, designing and managing this process execution would be very painful or costly for the mayor services and most probably not easy to repeat. He has to decompose his query into small tasks and post them to citizens. Then, he needs to decide how and when to move from one step to another, i.e., from data collection to data selection, or from data selection to data assessment. This is why we propose to design a crowdsourcing framework, that can be used in the context of smart cities to leverage citizen participation. It would *(i)* automatically decompose queries into simple tasks executable by humans; *(ii)* collect, aggregate and cleanse the data and answers provided by humans, and return the results to the user. This process could even be mixed with some automatic tasks depending on the expected outcome.

In this paper, we assume that the combination of data collection, selection and assessment can support a lot of scenarios. Based on this process, we propose an effective crowdsourcing framework called *CrowdSC*, which is designed for smart cities, and cope with the above mentioned challenges. The framework implements different strategies to manage the process. An experimental evaluation is conducted to evaluate the quality of the different strategies.

2 Related Work

Crowdsourcing tasks by citizens is a powerful means for making cities smarter. Hereafter, we provide an overview of some salient related work.

Location-based crowdsourcing: In [4], the authors propose a prototype for location-based mobile crowdsourcing consisting of a Web and a mobile client. Similarly, in [5], the authors design a framework for crowdsourcing location-based queries on top of Twitter. Kazemi and Shahabi propose in [6] new techniques for assigning the maximum number of tasks to a workers. Unfortunately, these frameworks cannot support complex queries. Given a query photo, taken from a location l, and a set of photos of l retrieved by a search engine, Yan et al. propose in [7] a system that ask humans to vote with "yes" or "no" for each retrieved photo. The authors then propose some techniques to find the most representative photo of l. This process may be useful from smart cities vision, but, can not support collecting and handling masses of data.

Optimising the quality of the results: Parameswaran et al. propose in [8] different strategies for filtering data with humans, using the "yes" or "no" filters. In [9], the authors propose different algorithms for finding the maximum, i.e., the best, item within a set of items. Given a quality threshold t, Liu et al. define in [10] the minimum number of humans to ask in order to achieve t. However, these works assume that the probability that each human provides the right answer is available, but in our case this information is not known.

3 Crowdsourcing Process Model

The first challenge is to transform our general crowdsourcing query into an executable process that aggregate different kinds of crowdsourcing activities. Here we consider a simplified form of query that can help to achieve contextualized goal that we have described. We will first define the input model of CrowdSC. Then we explain how to extract the tasks from the input and propose an output model.

3.1 Input Model

The input of CrowdSC is a query $Q = < O, C, L, A, T_s, T_e, S >$, where O describes the set of objects the user is looking for, C describes the context of O that citizens have to consider in answering Q, L stands for the location, i.e., the city, A comprises the domain of the assessments that can be attributed to O in the context of C, T_s and T_e are respectively the start and end time of the query execution, and S is a parameter to select a strategy (we present different strategies in Section 4, namely, Buffer, Deadline and FIFO). The query of our example would be represented as <roads, need repair, Nancy, {not damaged, damaged, very damaged}, 03/01/2013 – 8:00, 03/21/2013 – 20:00, deadline>.

3.2 Tasks Model

Given a query Q:$< O, C, L, A, T_s, T_e, S >$, we can define the data collection, data selection and data assessment tasks as follows:

- Data collection task: we use photos as a means to retrieve the set of objects O and define a data collection task DCT as a triple $< O, C, L >$. It asks citizens to take photos of O within context C, in location L. In our example, it asks citizens to take photos of road that need to be repaired in Nancy;
- Data selection task: as a lot of photos of the same object (e.g., the same road) are expected, we define a data selection task DST as a triple $< P, C, l >$, where P is a set of photos of objects in the same location l (we mean here by location the address, not the city like Nancy). A data selection task asks participants to vote "yes" for photos that represent location l within context C, and "no" for the others. The photo with the maximum number of "yes" votes is selected as the most representative photo of a location l within context C. For example, for $P = \{p_1, p_2, \ldots, p_{10}\}$, a data selection task can be: "vote for each photo in P that represents l within C";
- Data assessment task: once getting the most representative photos for a location, we need to assess this photo. To this end, we define a data assessment task DAT as a triple $< p, C, A >$, where p is a photo. An assessment task asks citizens to assess the photo p within context C, with assessment A. The result is the most attributed assessment. For example, if $p_3 \in P$ was selected, a data assessment task would be "assess p_3 with {not damaged, damaged, very damaged} within C.

3.3 Output Model

The output of CrowdSC, which comprises the result of the query Q, is a set of photos of O along with their locations and assessments. One can then invoke a map-based result visualisation service. Figure 1 shows a screenshot of the output model of CrowdSC.

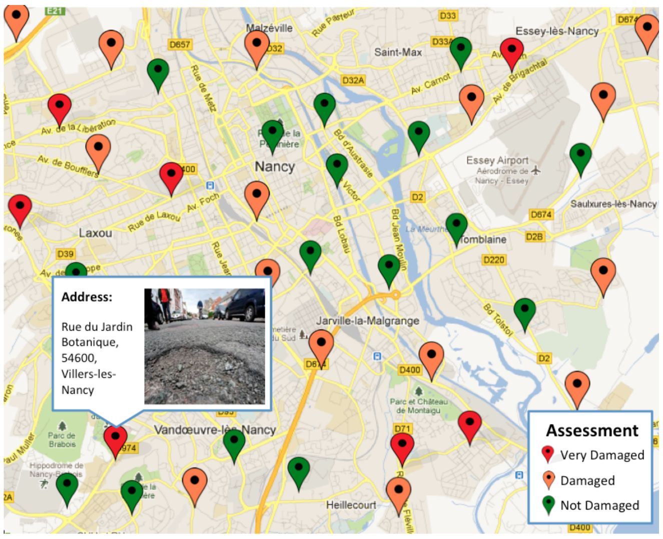

Figure 1: *Output Model of CrowdSC*

4 Query Processing Strategies

The second important challenge in building CrowdSC is managing and executing the different tasks, and detecting citizen errors or misbehaviour. The natural option to handle the errors is to distribute each task to k participants and to aggregate the answers [11, 8]. CrowdSC follows this direction for data selection and data assessment tasks since we assume that we will always find participants to provide answers. However, a data collection task is more critical since (1) it requires citizens to be on-site to take photos; and (2) it has an influence on the data selection and data assessment tasks. Of course, we can wait for k photos in each location l, but, this may take a very long time. In some scenarios, k photos may not be sufficient to get the right results. The number of photo that we need for eacu location may depends on the scenario as well as the duration of the query, i.e., T_s and T_e. Given a query Q:$< O, C, L, A, T_s, T_e, S >$, we present in the following three strategies to handle with these issues.

4.1 Buffer Strategy

The main idea of the buffer strategy is to wait for k photos of O from each location l ($l \in L$) to continue the processing. When k photos are collected for a given location l, we distribute the selection task regarding l and wait for k answers from citizens. The photo, say p, with the maximum number of "yes" votes is selected as the most representative photo of O for location l in the context C (if there is more than one photo with the highest score, we select randomly one among them). Then, we ask participants to assess the selected photo p (assessment are given in A), and wait for k answers. The more assigned assessment is considered as the appropriate assessment for the photo p. The process proceeds in the same manner for each location until T_e.

4.2 Deadline Strategy

The buffer strategy starts data selection only when it gets k photos for a given location l. Thus, if citizens do not provide k photos of l at T_e, we lose some results. This situation is expected in practice since some locations are more visited than others. Thus, we propose the deadline strategy. The idea is to collect photos of O starting from T_s until a deadline d, then to built buckets of photos for each location l ($l \in L$) – each bucket regroups photos of the same location. The next steps are similar to the buffer strategy. That is, we distribute the selection task for each bucket and wait for k answers. Once having the most representative photo, say p, for a given location in the contact C, we distribute the assessment task regarding p, and wait for k answers to get the assessment of p. The process proceeds in the same manner for each bucket until T_e.

4.3 FIFO Strategy

To move from the data collection phase to the data selection phase for a given location l, the buffer and deadline strategies wait for k photos of l or for the deadline d, respectively. It might be interesting to consider an execution that would generate results more instantly and where the result size would increase gradually. We propose the FIFO strategy, which proceeds as follows. When we receive a photo, say p, of O in a given location l, we immediately ask citizens to vote with “yes” or “no” to see if that photo really represents l in the context C, and wait for k answers. If the majority of votes are “yes”, we select p as a representative photo for location l; otherwise, we wait for another photo of O for l, and repeat the same procedure. Once having the representative photo of O for location l, the FIFO strategy follows the similar strategy as the buffer and deadline strategies, i.e., we distribute the assessment task regarding the selected photo, and wait for k answers to get its assessment. This strategy proceeds in the same manner for each location until T_e.

5 System Architecture

CrowdSC is implemented using Bonita[12], an open-source BPM suite. Figure 2 describes the general architecture of CrowdSC. The main components of CrowdSC are the *Process Generator*, the *Process Engine*, the *Task Manager* and the *Result Visualizer*. The Storage Engine is external, and is accessed by our system at query time.

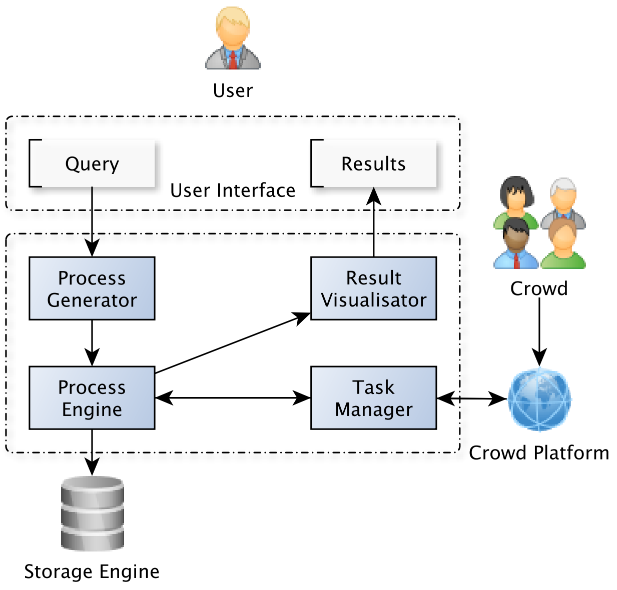

Figure 2: *CrowdSC Architecture*

The *Process Generator* receives the query from the user and transform it into a processing plan, i.e., it generates the data collection, data selection and data assessment process along with other service calls if needed.

The *Process Engine* takes the processing plan from the *Process Generator* and generates a sets of tasks for the citizens to perform and aggregates the answers; Roughly speaking, the role of the *Process Engine* is to execute the process according to the strategy (buffer, deadline or FIFO) chosen by the user.

The *Task Manager* receives progressively the tasks from the *Process Engine* and communicates with the crowd via a crowd platform to post tasks and retrieve the answers, then send them to the *Process Engine*. We assume that the citizen have installed an App on their phone that is able to communicate with the task manager. How we can get the citizens to install and use the app is out of our scope but we can imagine that Smart Cities may provide a dedicated app that provide access to the city services as it is common today.

The role of the *Result Visualizer* is to receive the results from the *Process Engine* and return them to the user as a map.

6 Experimental Evaluation

The last challenge with our system is to be able to evaluate the quality of the results. Since we have not yet used it in a large scale setting, we propose to evaluate the respective qualities of the proposed strategies, i.e., buffer, deadline and FIFO, focusing on: *(i)* the number of results returned; *(ii)* the quality of the results; for which, we use the F-measure, i.e., $2\frac{precision.recall}{precision+recall}$; and *(iii)* the progressivity, i.e., how the results accumulate over time.

Due to the limited availability of large real datasets, we use the Gowalla dataset [13]. It contains a set of users along with their check-in time and location on Gowalla, a location-based social networking website where users share their locations by checking-in. For our experiments, we assume that each user is a participant citizen. To generate the ground truth document, we make one random assignment of three possibilities for each location. We use a period of 5 days, and divide the data into three parts so that we have data collectors, data selectors and data assessors. We consider each check-in time in a given location as a response from that location for the current processing query. To be more realistic, for each collected data (photo), we introduce a parameter in $[0.5, 1]$ that controls whether the data selectors and data assessors provide a correct answer or not. That is, we assume that citizens have a high probability to provide correct answers.

For the deadline strategy, the deadline is set to the half of the duration of the process, i.e., $d = \frac{T_e - T_s}{2}$. For all strategies, the default values for the duration of the process, i.e., $T_e - T_s$ and the k value are 3 days and 7, respectively.

Number of results: we measured the number of results, varying the duration of the process from 1 day to 5 day. The results of this experiment are presented in Figure 3. As expected, when the duration increases, the number of results increases since more citizens can participate. The number of results returned by Buffer is much less than that returned by Deadline and FIFO. Buffer needs to wait for k photos for each location, and some location are not popular. FIFO returns more results than Deadline since this later starts the data selection and assessment after the deadline, while the former, can do it whenever there is a photo received.

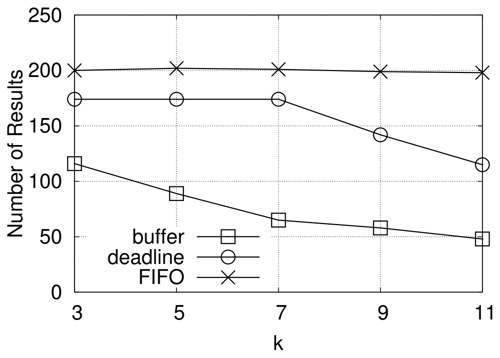

Figure 4: *Number of Results vs k (Duration = 3 days)*

We also measured the number of results, varying k from 3 up to 11. Figure 4 shows the results of this experiments. FIFO returns more result than the other strategies and remains unaffected since it does not wait for k photos or the deadline to starts the data selection step. Also, deadline remains unaffected for $k \leq 7$, and for $k > 7$, the number of results returned decreases as the remaining time after the deadline d is insufficient to select and assess the data. Moreover, the number of results returned by the buffer decreases significantly with the increase of k since it require k photos for each location since they have more time for collecting data.

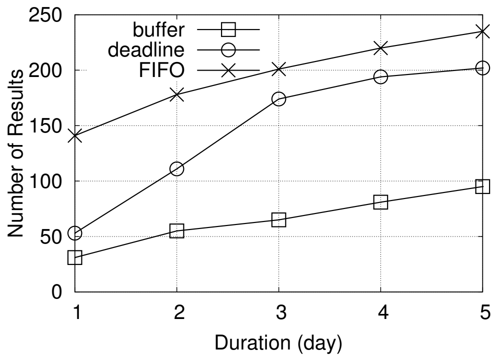

Figure 3: *Number of Results vs Duration (k = 7)*

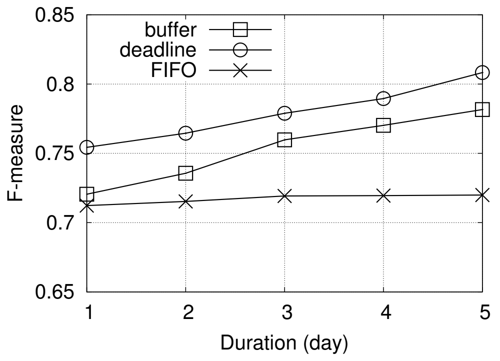

Figure 5: *Quality vs Duration (k = 7)*

Quality: Figure 5 shows the quality of the results varying the duration. As can be seen, FIFO has the lowest quality, and it is not affected by the duration.In almost all cases, the first photo of each location is selected; Since it is probably not the best photo of that location it affects the assessment step. The quality of Deadline and Buffer increases with the duration. Note that deadline is better since it collects the maximum number of photos until the deadline d, and can move to data selection and assessment without having k photos for a location, while buffer, misses some location because some photos are missing to reach the threshold.

Figure 6 depicts the quality of the results varying k. From this experiment, we can see that the quality of all strategies increases with the increase of k. This is because, when using more citizens, the probability to get a correct answer increases. Similarly to the last experiment, Deadline is better than Buffer, which in turn is better than FIFO for the same reasons.

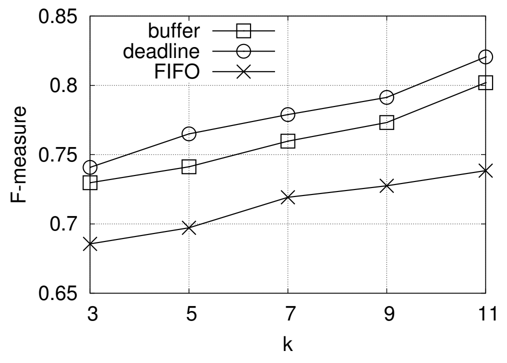

Figure 6: *Quality vs k (Duration = 3 days)*

Progressivity: As shown in Figure 7, FIFO and Buffer return some results the first day and progressively return the remaining ones. However, FIFO is around 3 times better since buffer have to wait for k photos for each location. Moreover, the first results of Deadline can only appear after the deadline d, i.e., after 1.5 days, then increases significantly.

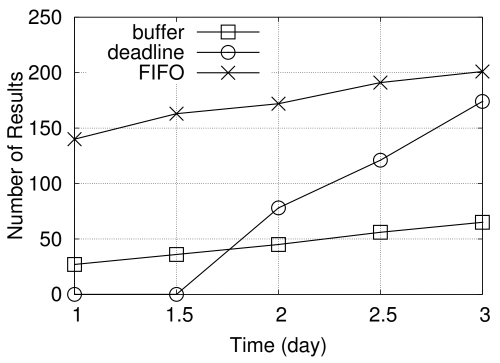

Figure 7: *Progressivity (Duration = 3 days, $k = 7$)*

7 Conclusion

We have presented a crowdsourcing framework for smart cities called CrowdSC, which aims at making cities smarter by leveraging citizen participation. We have provided different strategies based on the process of data collection, data selection and data assessment to answer users' complex queries. Through our experimental evaluation, we can see that each strategy has its own merits, therefore, the choice of the strategies depends on the user needs. Hence, the proposed strategies are complementary. For instance, if high accuracy is required, the Deadline strategy appears to be better but in a crisis situation where results are required urgently to take actions, the FIFO strategy is very well suited. Still, we have shown using a simulation that it is possible to combine different kind of crowdsourcing activities in a process to achieve potentially complex result within a specified context. More requires to be done on the side of the actual execution with our process engine and with actual citizens.

References

[1] Hans Schaffers, Nicos Komninos, Marc Pallot, Brigitte Trousse, Michael Nilsson, and Alvaro Oliveira. Smart cities and the future internet: Towards cooperation frameworks for open innovation. In *Future Internet Assembly*, pages 431–446, 2011.

[2] `https://www.mturk.com/mturk/welcome`.

[3] Jeff Howe. The Rise of Crowdsourcing. (accessed July 20, 2010), June 2006.

[4] Florian Alt, Alireza Sahami Shirazi, Albrecht Schmidt, Urs Kramer, and Zahid Nawaz. Location-based crowdsourcing: extending crowdsourcing to the real world. In *NordiCHI*, pages 13–22, 2010.

[5] Muhammed Fatih Bulut, Yavuz Selim Yilmaz, and Murat Demirbas. Crowdsourcing location-based queries. In *PerCom Workshops*, pages 513–518, 2011.

[6] Leyla Kazemi and Cyrus Shahabi. Geocrowd: enabling query answering with spatial crowdsourcing. In *SIGSPATIAL/GIS*, pages 189–198, 2012.

[7] Tingxin Yan, Vikas Kumar, and Deepak Ganesan. Crowdsearch: exploiting crowds for accurate real-time image search on mobile phones. In *MobiSys*, pages 77–90, 2010.

[8] Aditya G. Parameswaran, Hector Garcia-Molina, Hyunjung Park, Neoklis Polyzotis, Aditya Ramesh, and Jennifer Widom. Crowdscreen: algorithms for filtering data with humans. In *SIGMOD Conference*, pages 361–372, 2012.

[9] Petros Venetis, Hector Garcia-Molina, Kerui Huang, and Neoklis Polyzotis. Max algorithms in crowdsourcing environments. In *WWW*, pages 989–998, 2012.

[10] Xuan Liu, Meiyu Lu, Beng Chin Ooi, Yanyan Shen, Sai Wu, and Meihui Zhang. Cdas: A crowdsourcing data analytics system. *PVLDB*, 5(10):1040–1051, 2012.

[11] Michael J. Franklin, Donald Kossmann, Tim Kraska, Sukriti Ramesh, and Reynold Xin. Crowddb: answering queries with crowdsourcing. In *SIGMOD Conference*, pages 61–72, 2011.

[12] `http://fr.bonitasoft.com/`.

[13] `http://snap.stanford.edu/data/loc-gowalla.html`.