DESIGN AND IMPLEMENTATION OF PARTICIPANT SELECTION FOR CROWDSOURCING DISASTER INFORMATION

E.T.-H. CHU[1], C.-Y. LIN[1], P.H. TSAI[2] & J.W.S. LIU[3]
[1]Department of Computer Science and Information Engineering, National Yunlin University of Science and Technology, Taiwan, R.O.C.
[2]Institute of Manufacturing Information and Systems, National Cheng Kung University, Taiwan, R.O.C.
[3]Institute of Information Science, Academia Sinica, Taiwan, R.O.C.

ABSTRACT
A typical disaster surveillance and early warning system often needs to make timely and critically important preparedness decisions before disasters strike. When crowdsourcing observational data from people to enhance its sensor coverage, the system must be able to make effective use of volunteers, guide them during their exploration of the threatened area, and process reports from them in real time to extract decision support information of sufficiently good quality. This article focuses on the participant selection problem (PSP) which the system must solve in order to select participants from available volunteers given the benefits and costs of deploying them. The PSP-Greedy (PSP-G) algorithm proposed is known to be a near-optimal solution with a small fraction of execution time when compared with well-known optimization methods. The article describes an implementation of the PSP-G algorithm and the integration of the resultant PSP-G module into the Ushahidi platform. Performance data from two case studies based on Haiti Earthquake, 2010, and Typhoon Morakot, 2009, also described here, clearly show that PSP-G is a general practical solution.
Keywords: Crowdsourcing, disaster management, maximum general assignment, social network.

1 INTRODUCTION

In recent years, smart mobile devices equipped with cameras; temperature and vibration sensors, etc.; and social networking services have become increasing more pervasive. Using them, increasingly more people are able to participate as *human sensors*. *Human sensor data* (i.e. observational data captured and contributed by human sensors) from them have enabled an increasingly broader spectrum of crowdsourced sensing systems and applications for purposes such as the generation of fine-grain maps of noise level, air quality, snow depth, radiation level, traffic and road conditions, etc. Crowdsourcing information from mass crowd during and after major disasters has also proved to be an effective crisis management tool and is supported by well-known crisis management systems such as Sahana [1] and Ushahidi [2].

In contrast, modern disaster surveillance and early warning systems typically do not crowd source human sensor data. Crowdsourcing is not a routinely used tool although observational data captured and sent from regions threatened by an imminent disaster by volunteers selected by the system to participate in its surveillance effort can be an effective way to mend fragmentations in physical sensor coverage and improve situation awareness of the system, especially in scenarios such as those described in refs. [3–5] when physical sensor coverage may be inadequate for one or more unavoidable reasons.

Unlike typical crowdsourced sensing systems and applications, a disaster surveillance system must be able to make critical decisions, sometimes within minutes or seconds. If the system uses human sensor data, it must be able to process them in real time and extract from them decision support information of good and quantifiable quality. Existing techniques and tools for processing social reports cannot meet this requirement. Often, the system should use

ISSN: 2041-9031 (paper format), ISSN: 2041-904X (online), http://www.witpress.com/journals
DOI: 10.2495/SAFE-V5-N1-48-62

as few participants as possible, direct them to locations where physical sensor coverage is poor, and help them to stay away from dangerous locations. The system needs tools for these purposes as well. CROwdsouring Support system for disaster Surveillance (CROSS) [6] was motivated by these needs. The proof-of-concept prototype contains a basic set of tools for managing *human sensor data collection* (HSDC) processes, including a fusion unit to help the system decide whether human sensor data is needed; a broadcast manager for soliciting volunteers; a participant selection module for selecting participants from volunteers; and a tour planning module, a map manager, and an emergency communication module for guiding participants in their exploration of the threatened area, tracking their locations, and supporting their communication with the command center and with each other.

This article focuses on the participant selection component of the system. The problem that the module must solve is to select from volunteers, participants of each HSDC process, and assign selected participants to collect data in different regions of the threatened area in order to optimize some specified objective, subject to constraints in the number, benefits and costs of available volunteers is called the participant selection problem (PSP). An earlier article [7] presented a natural formulation of the PSP and a heuristic algorithm, called PSP-Greedy (PSP-G), for solving the problem. Preliminary data show that PSP-G can find near-optimal solutions with a small fraction of execution times of Branch-And-Reduce Optimization Navigator (BARON) [8] and Basic Open-source Nonlinear Mixed Integer programming (BONMIN) [9], two well-known optimization solvers.

This article aims to demonstrate that PSP-G is a practical solution for selecting participants from volunteers in general and should be included in tool libraries of not only experimental systems such as CROSS but also in a commonly used platform such as Ushahidi [2]. For this purpose, the effectiveness of PSP-G was evaluated in two case studies based on real-life situations: Haiti, 2010, earthquake [10] and Typhoon Morakot, 2009 [5,11]. A PSP-G module that implements the algorithm has been integrated into the Ushahidi platform. The module, the extended platform Ushahidi$^+$, and the case studies are presented in Sections 4 and 5, respectively. To make the paper sufficiently self-contained, Sections 2 and 3 present an overview of the results in Chu *et al.* [7]: The formulation of the PSP and related problems are presented in Section 2. The PSP-G algorithm and the preliminary performance data on its performance are presented in Section 3. Section 6 summarizes the article and presents future work.

2 PSP AND RELATED PROBLEMS

Specifically, the solution of a PSP is a selection of participants of a HSDC process, or a crowdsourcing process in general, from available volunteers and an assignment of the selected participants to regions in order to optimize some objective function subject to constraints in terms of the number, qualities, and costs of the volunteers available for selection. The input parameters of the PSP are represented by the following notations:

- V is the objective function and is referred to as the (*total*) *value* of the selection.
- The disaster threatened area has ρ regions $R_1, R_2, \ldots R_\rho$, and their values are $v_1, v_2, \ldots v_\rho$, respectively.
- π volunteers $P_1, P_2, \ldots P_\pi$ are available for selection to be participants.
- For $i = 1, 2, \ldots \pi$ and $k = 1, 2, \ldots \rho$
    - $b_{ik}(0 \le b_{ik} \le v_k)$ is the *value* (*benefit*) achievable by P_i if he/she is selected and assigned to explore region R_k, and
    - $c_{ik}(0 \le c_{ik})$ is the cost of P_i when assigned to explore region R_k
- B (> 0) is the total budget available to be spent on all selected participants.

Without loss of generality, the values of regions, costs of participants, and total budget are assumed to be positive integers.

The PSP can be stated as follows in terms of these notations:

Maximize
$$V = \sum_{k=1}^{\rho} \sum_{i=1}^{\pi} b_{ik} x_{ik} \tag{1}$$

Subject to
$$\sum_{i=1}^{\pi} b_{ik} x_{ik} \le v_k, \quad k = 1, 2, \ldots \pi \tag{2}$$

$$\sum_{k=1}^{\rho} x_{ik} \le 1, \; i = 1, 2, \ldots \rho \tag{3}$$

$$x_{ik} \in \{0, 1\}, \; i = 1, 2, \ldots \pi, \quad k = 1, 2, \ldots \rho \tag{4}$$

$$\sum_{k=1}^{\rho} \sum_{i=1}^{\pi} c_{ik} x_{ik} \le B \tag{5}$$

The variable $x_{ik} = 1$ means that volunteer P_i is selected to participate in the current HSDC process and is assigned to region R_k; it is equal to 0 if otherwise. The set $\{x_{ik}\}$ for all $i = 1, 2, \ldots \pi$ and $k = 1, 2, \ldots \rho$ gives an assignment of a subset of participants to regions. Equation (3) allows $\{x_{ik}\}$ to be a proper subset of the set of all volunteers. The term V in eqn (1) is equal to the sum of benefits contributed by all the participants; it is the total value achievable by all the selected participants when they explore their assigned regions. Equation (5) shows that the total cost incurred by them must not be greater than the budget B. Equation (2) ensures that the solution $\{x_{ik}\}$ never assigns more participants to any region than needed to achieve the full value of the region. The variant of the PSP is called *PSP-Frugal* because of this constraint.

In the case of infinite budget (i.e. $B = \infty$), PSP-Frugal is a special case of the well-known *maximum general assignment problem* (GAP) [12,13], which in turn is a generalization of the assignment problem [14]. The assignment problem is often stated as that of seeking an optimal placement of objects in bins. For each bin, each object in it has a profit and a weight that are dependent on both the object and the bin. The objective is to find placements of objects in bins so that the total profit is maximized subject to the constraints that the total weight of all objects placed in every bin is no greater than the given weight limit of the bin. The GAP is known to be NP-hard (Non-deterministic Polynomial-time hard) and APX-hard (Approximable hard) to approximate it. Some algorithms for solving the problem use algorithms for the 0-1 knapsack problem [15] as the basis. An example is the greedy ($\delta + 1$)-approximation algorithm in [12]. It finds a solution of the GAP iteratively using a δ-approximation algorithm for the knapsack problem to find a tentative solution of the single-bin sub-problem, and one bin at a time.

The special case of equal weight and profit knapsack problem is known as the subset sum problem [16]. The functional form of the subset sum problem can be stated as follows: Given a set of N non-negative integers, find a subset of integers with the maximum sum among all subsets with sums equal to or less than the given limit. This problem is known to be NP-hard in general, but can be solved exactly in a reasonable amount of time by exhaustive search when N is small (e.g. less than 20) or by dynamic programming when the precision of the problem is small. For the PSP-Frugal, N is the number π of available volunteers, which can be large. On the other hand, the number of distinct values of b_{ik} is usually small. In practice, it also makes sense to adjust the unit of b_{ik}'s to reduce the precision of the problem.

The PSP can also be thought of as a variant of human resource allocation problem, which has also been treated extensively for many other types of applications. As examples, Kwon and

Cho [17], Bartoli *et al.* [18], and Chen *et al.* [19] proposed human resource allocation algorithms for various scenarios. Compared to the formulation presented above, their models and problem formulations are *ad hoc*. Indeed, the work on PSP described here is among the first ones that apply GAP and knapsack algorithms to resource allocations by disaster surveillance and response applications.

3 PSP-G ALGORITHM AND ITS PERFORMANCE

Table 1 describes the heuristic PSP-Greedy (PSP-G) algorithm for solving the PSP in polynomial time [7]. It is presented here to make this article more self-contained.

After initializing related parameters (line 1), PSP-G calculates the benefit-to-cost (B2C) factors $Q_{i,k}$ (= b_{ik}/c_{ik}) of each volunteer P_i (line 2) and each region R_k and sorts all the B2C factors by their values in non-increasing order (line 3). It then selects from volunteer participants and dispatches (i.e. assigns) selected participants in turn to the regions according to volunteers' B2C (lines 4–13): Each time, the volunteer with the highest B2C is selected (line 5) and is dispatched to a region where he or she can increase the total value most if this assignment satisfies both the budget and the value constraints (line 6). The participant selection process stops when all volunteers are selected and assigned or when the total value cannot be further increased. The last line (line 14) says that a new call for participation in HSDC is issued to solicit more volunteers if the threatened area is not fully explored.

As one sees from Table 1, PSP-G takes $O(\rho\pi)$ to calculate the B2C factors (line 2). It takes $O(\rho\pi \log(\rho\pi))$ to sort them (line 3). The time complexity of the selection process (lines 4–13) is bounded by $O(\rho\pi)$. Therefore, the time complexity of PSP-G is $O(\rho\pi\ ln\ \rho\pi)$.

Table 1: Pseudocode description of PSP-G algorithm.

Algorithm for PSP-G

β_k: The current benefit region *k* gets from the selected participants

ψ: The remaining budget

Q: An ordering set to record all benefit-to-cost (B2C) factors $Q_{i,k}$ of participant *i* to region *k*.

```
1:  Set $\psi = B$ and $\beta_k = 0, k = 1, 2, \ldots, \rho$;
2:  Calculate all B2C factors (i.e. $Q_{i,k} = b_{ik}/c_{ik}$);
3:  Sort $Q_{i,k}$ by their values in descending order;
4:  **while** ($Q$ is not empty)
5:  $H_{i,k}$ = the first element in $Q$;
6:  **if** ($P_i$ is not selected and ($\beta_k + b_{ik} \le v_k$) and ($\psi - c_{ik} \ge 0$))
7:  $\beta_k += b_{ik}$; $\psi -= c_{ik}$;
8:  Dispatch $P_i$ to $R_k$;
9:  Remove all $P_i$ 's B2C factors from $Q$;
10: Mark $P_i$ as a selected participant;
11: **end if**
12: Remove $H_{i,k}$ from $Q$;
13: **end while**
14: If participants are not enough to cover all regions, broadcast a call-for-volunteer message
    on social network again.
```

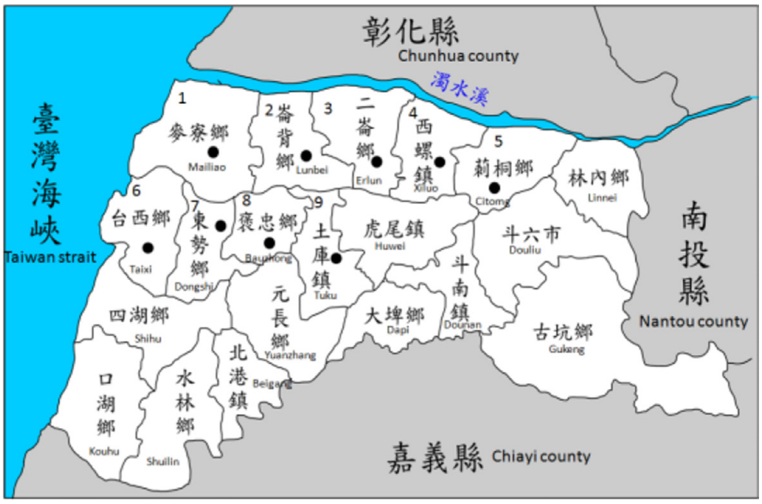


Figure 1: Map of Yunlin County, Taiwan.

Table 2: Model of volunteers.

Type	Property	Benefit	Cost
I	Professional responders	High	High
M	Registered volunteers	Medium/low	Medium/low
U	Unregistered volunteers	Low	Low

The PSP-G algorithm was evaluated via simulation experiments conducted on an Intel i7 processor with CPU speed 3.3 GHz and the total RAM is 6 GB [4]. The algorithm PSP-G was written in Java with Eclipse. The simulated scenario is a big earthquake that seriously damaged parts of Yunlin county, Taiwan. The affected area has the nine regions marked by dots and numbered by 1, 2, … 9 in the map shown in Fig. 1. The system started a HSDC process to collect data in the regions. It uses three types of volunteers with characteristics listed in Table 2. The total number of volunteers available for selection is 1,000. The number of each type of volunteers is one-third of the total volunteers. In other words, the first 333 volunteers are of type I, volunteers no. 334–666 are of type M, and remaining volunteers are of type U. Initially, volunteers of all types are uniformly distributed in the threatened area.

The simulation study in Chu *et al.* [7] assumes that the system sets the values of the regions at 100, 100, 80, 60, 80, 60, 60, 60, and 60, respectively. It uses three parameters to compute the benefit and cost of a participant assigned to a region: *Basic_Benefit*, *Basic_Cost*, and *Distance* from his/her initial location to the assigned region. The *Basic_Benefit* and *Basic_Cost* of a type-I participant are 10 and 3, of a type-M participant are 5 and 2, and of a type-U participant are 3 and 1, respectively. For sake of simplicity, it sets the distance between each pair of regions to the minimum number of region boundaries that a participant must cross to reach from one region to the other region. (e.g. the distance between regions 6 and 7 is 1, whereas the distance between regions 6 and 5 is 5.) The benefit b_{ik} and the cost c_{ik} for participant P_i and region R_k are computed according to

$$b_{ik} = Basic\_Benefit \times Distance$$
$$c_{ik} = Basic\_Cost/Distance$$

respectively, from the distance between the region where the participant is initially and the region R_k.

The performance of the PSP-G algorithm was compared with two commonly used optimization solvers: BARON [8] and BONMIN [9]. Both solvers were executed by NEOS servers online [20]. BARON is a global optimization solver for convex optimization problems. It solves both linear programming and nonlinear programming problems by using branch and bound strategies. BONMIN, also a global optimizer, adopts six different strategies (i.e. B-BB, B-OA, B-QG, B-Hyb, B-ECP, and B-iFP) to have better performance in optimization. These solvers represent the state of the art in optimization software.

Figure 2 shows the performance ratios of the total value achieved by PSP-G relative to the values achieved by the optimizers: V_g, V_{br}, and V_{bo} denote the total values achieved by PSP-G, BARON, and BONMIN, respectively. In the figure, the *x*-axis is the number of threatened regions and *y*-axis is the performance ratios V_g/V_{bo}, and V_g/V_{br}, i.e. the ratios of the total benefit achieved by PSP-G algorithm relative to the total benefits achieved by BONMIN and BARON, respectively. One can easily see from Fig. 2 that in all test cases, performance ratios V_g/V_{br} and V_g/V_{bo} are close to 1. This fact indicates that PSP-G is near optimal: It delivers almost the same total value as BARON and BONMIN.

The execution times of PSP-G, BARON and BONMIN were also measured. In this experiment, the number of regions was set at nine and the number of participants was varied from 1,000 to 8,000. As Fig. 3 shows, compared with BARON and BONMIN, the execution time of PSP-G increases only slightly when the number of participants increases. Most of the experiment configurations can be finished by PSP-G in a few seconds. In contrast, the execution times of BARON and BONMIN increase significantly when the number of participants

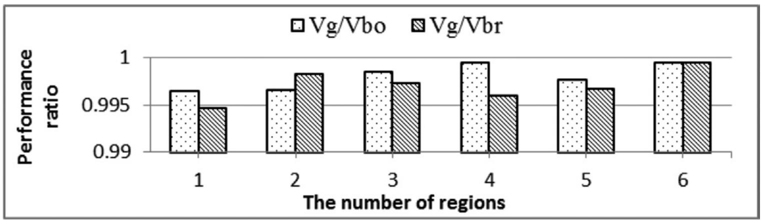


Figure 2: The performance ratios.

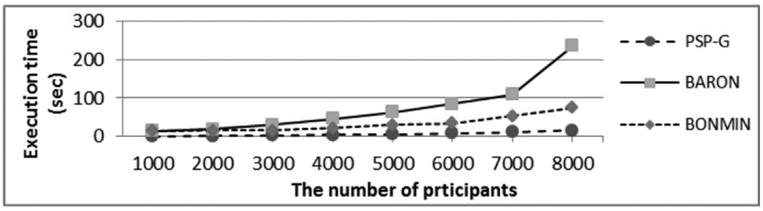


Figure 3: The execution time.

becomes large. When the number of participants comes to 8,000, the execution time BARON is almost 10 times longer than PSP-G.

4 IMPLEMENTATION OF PSP-G MODULE

To demonstrate the applicability of PSP-G in general for selecting participants of crowdsourcing processes from volunteers, Ushahidi platform [8] was enhanced by adding to it a PSP-G module that implements the algorithm and a mobile emergency APP that allows official responders and volunteers to easily communicate with Ushahidi server. The enhanced system is called Ushahidi$^+$. This section describes the architecture of Ushahidi$^+$ and the new emergency APP together with the flow of crowdsourcing disaster information.

4.1 The architecture of Ushahidi$^+$

Figure 4 shows the architecture of Ushahidi$^+$. The workflow diagram on the left illustrates a standard operating procedure (SOP) followed by a crowdsourcing-enhanced emergency management system. In particular, the figure shows how a PSP-G module and other participant selection tools may be used to support crowdsourcing information when a disaster is imminent or during emergencies. The procedure typically starts by official responders from the collection from physical sensors and eyewitnesses of data and information needed to acquire situation awareness. When the emergency manager or the response system finds available data and information insufficient for threat level evaluation and situation assessment (or when some response operations need to be performed), a volunteer recruitment message containing required and desired attributes (e.g. identifications, locations, skills) of volunteers is posted online. After a sufficient number of people have volunteered by reporting their attributes to the system, one or more participant selection tools are used to select participants of the current crowdsourcing process and assign each selected participant to a region in the threatened area. The system may send to each selected and assigned participant a route for the participant to follow, in addition to information on the tasks (e.g. reporting damaged buildings or emergency events encountered along the route) to be performed by the participant. During the current process, more data and information are collected by the system, allowing the system to better assess the situation and make a decision, including the decision

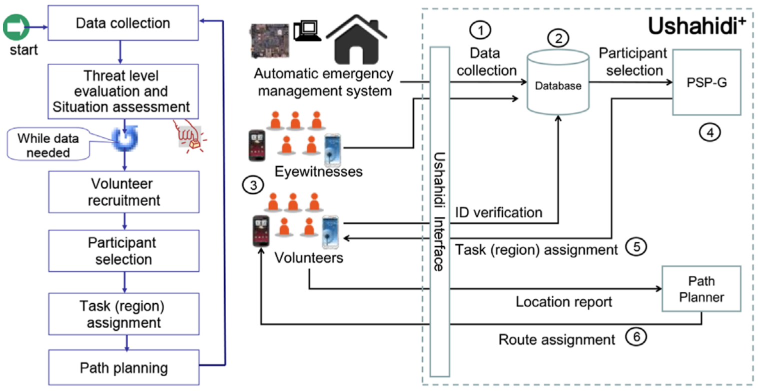


Figure 4: Architecture of Ushahidi$^+$.

on whether to recruit more volunteers and start new processes. Without loss of generality, the workflow diagram assumes that the current process is stopped manually when no more data needs to be collected.

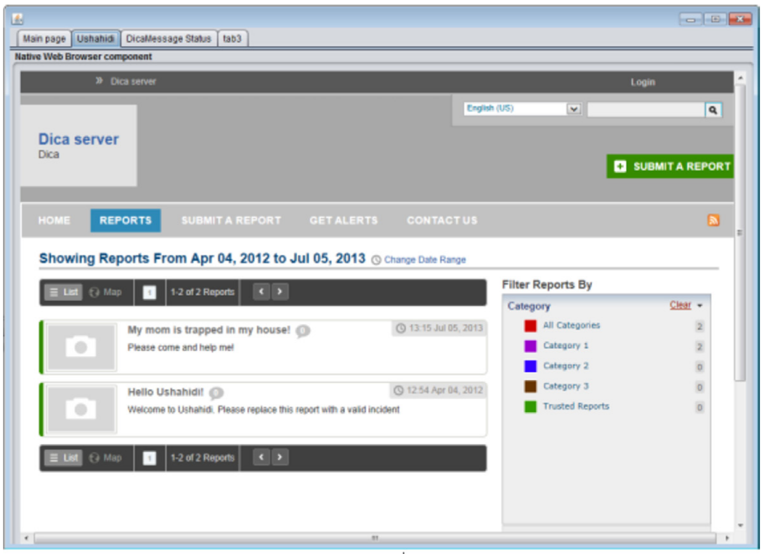
The diagram on the right-hand side of Fig. 4 shows the structure and components of the Ushahidi$^+$ that supports the SOP. Ushahidi$^+$ has three key modules: Database, PSP-G module, and Path Planner. The database stores information on all events reported by participants, as well as information on volunteers and selected participants. A default voting mechanism is used to verify the correctness of reported events. System administrator can delete events that cannot be identified and verified. PSP-G module is used for selecting participants, whereas Path Planner is used for determining a route for each participant. In other words, the PSP-G module supports the participant selection step and a part of task assignment step. A path planner implementing a tour planning algorithm such as the one described in Chu *et al.* [6] can be used to compute routes for all the selected participants.

The diagram on the right-hand side of Fig. 4 also shows that the data collection (Step 1) is done by an active emergency disaster system (AEDS), in addition to official responders, eye witnesses, or victims. The AEDS [21] is an advanced disaster response system. Smart devices in it can parse standard-based warning messages sent by alert authorities (i.e. central weather bureau) and automatically perform emergency tasks (e.g. shutdown gas intake and return elevator to the ground floor) to avoid possible dangers. The AEDS can also collect information about victims through sensors deployed in buildings and report emergencies to Ushahidi$^+$. Once in the database, collected data and information are analyzed (Step 2). The Ushahidi classifies events into eight types: emergency, vital lines, public health, security threats, infrastructure damage, natural hazard, services available, and others. In Ushahidi$^+$, system administration should assign a score to each type of event to indicate its emergency level. If the additional participants are needed to deal with emergencies, the emergency manager posts volunteer recruitment messages on social network, such as Facebook or Twitter, in order to attract more volunteers (Step 3). Volunteers can report their personal information, including name, location, and professional speciality, to the server via the mobile emergency APP described below or Ushahidi$^+$ web interface shown in Fig. 5. After a sufficient number of volunteers have been recruited, the PSP-G module is triggered to perform participant selection (Step 4) followed by task assignment (Step 5) and tour planning (Step 6) as described earlier.

4.2 Mobile emergency APP

The new Ushahidi$^+$ mobile emergency APP was developed to assist volunteers, participants, and victims to communicate with Ushahidi$^+$ server. It is based on the original Ushahidi client APP. In addition to the default functions provided by the Ushahidi client APP, the Ushahidi$^+$ APP provides several functions for crowdsourcing volunteers to report their personal information (including their ids, skills, and locations) and data captured and events witnessed by them back to the server. Volunteers can also use the APP to receive information from the server on whether or not they have been selected to participate, the regions to which they are assigned, and to record and visualize the paths planned for them by the path planning module. The APP also supports online text and audio communication so that registered users can easily communicate with each other. Finally, victims and responders can also use the emergency APP to report events and needs instead of Ushahidi client APP.

Figure 6 shows user interfaces of the Ushahidi$^+$ emergency APP. When the user opens the mobile emergency APP, it first identifies the role of the user. If the user is a victim, the mobile


Main page | Ushahidi | DicaMessage Status | tab3

Native Web Browser component

Dica server Login

English (US)

Dica server
Dica

SUBMIT A REPORT

HOME REPORTS SUBMIT A REPORT GET ALERTS CONTACT US

Showing Reports From Apr 04, 2012 to Jul 05, 2013 Change Date Range

List Map 1 1-2 of 2 Reports

My mom is trapped in my house! 13:15 Jul 05, 2013
Please come and help me!

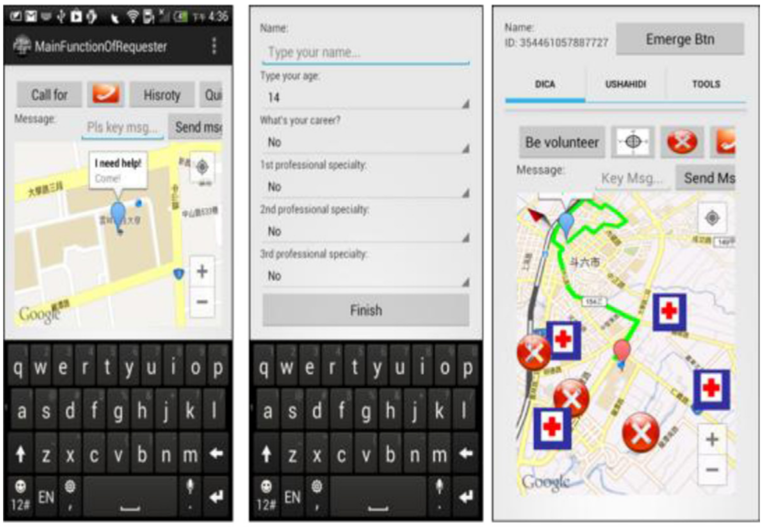
Hello Ushahidi! 12:54 Apr 04, 2012
Welcome to Ushahidi. Please replace this report with a valid incident

List Map 1 1-2 of 2 Reports

Filter Reports By

Category	Clear
🟥 All Categories	2
🟪 Category 1	2
🟦 Category 2	0
🟫 Category 3	0
🟩 Trusted Reports	0

Figure 5: Ushahidi$^+$ web interface.


MainFunctionOfRequester

Call for | Hisroty | Qui

Message: Pls key msg... Send msg

I need help!
Come!

Name: Type your name...

Type your age: 14

What's your career? No

1st professional specialty: No

2nd professional specialty: No

3rd professional specialty: No

Finish

Name:
ID: 354461057887727 Emerge Btn

DICA USHAHIDI TOOLS

Be volunteer

Message: Key Msg... Send Ms

(a) (b) (c)

Figure 6: User interfaces of mobile emergency APP.

emergency APP will enter the victim page where the user can report his/her emergency to the official agency (Fig. 6a). On the other hand, if the user is a volunteer, the mobile emergency APP will enter the participant pages, shown in Fig. 6b, where the user can fill in personal information, such as location, expert knowledge, skill, and what items he/she needs (e.g. food, water, etc.). After the registration process completes, the volunteer then logs into the Ushahidi$^+$ server to get an updated disaster map (Fig. 6c). If the volunteer is selected to be a

Table 3: Default threat score of each Ushahidi-defined event.

Ushahidi event type	Threat score	Ushahidi event type	Threat score
Emergency	20	Infrastructure damage	12
Security threats	18	Natural hazard	10
Public health	15	Vital lines	8
Services available	13	Other	5

participant, Ushahidi$^+$ server sends the map of the target region to him/her. The participant then follows the instruction to move to the target region. After the participant arrives at the target region, Ushahidi$^+$ server determines a route, shown in Fig. 6b, for him/her to follow and perform rescue actions, such as reporting damaged buildings or checking an emergency event.

The disaster event synchronization between Ushahidi$^+$ server and mobile emergency APP is crucial because it helps participants to avoid possible dangers. A monitoring server was implemented to periodically check a flag file stored in the Ushahidi$^+$ server. Whenever the monitoring server finds an update, it will notify mobile emergency APP to download the latest events. The major advantage of this synchronize mechanism is that it does not need to modify the kernel of Ushahidi and can be applied to different versions of Ushahidi.

4.3 The default settings of Ushahidi$^+$

Before starting the process of participant selection, the system administrator has to set up the values of several variables. They are the value of each region, the cost(s) of each participant and the achievable value(s) of each participant. Clearly, the right choices of these parameters depend on the emergency scenarios. As an example, the administrator may assign a score to each type of event. As a choice, the administrator may set the value for each region equal to the sum of emergency scores of all events happening in that region. Table 3 lists the default threat scores of Ushahidi-defined events, which can be used to compute the values of regions. System administrators can modify the setting according to their needs.

To determine the cost of each participant, a volunteer can specify his/her needs through the mobile emergency APP. Ushahidi$^+$ also includes a questionnaire for volunteers to fill in their skills, such as medical skills, language skills, multicultural sensitivity/awareness, mental health, and driver license. Administrators can assign a score to each type of skill in order to assess the achievable value of each participant.

5 CASE STUDIES

As stated in Section 1, the case studies described here were carried out in order to demonstrate the applicability of PSP-G module in general. In the simulation experiment described in Section 3, the values of input parameters of the greedy algorithm were chosen in a more or less *ad hoc* manner. This section presents more rationale choices of the parameter values based on two real-life disaster scenarios. Moreover, it aims to show that the relative merit of the PSP-G algorithm is not sensitive to values of the input parameters.

The scenarios of the case studies are the 2010 Haiti earthquake [10,22] and 2009 Morakot typhoon in Taiwan [5,11]. The former was a catastrophic magnitude 7.0 MW earthquake that occurred at 16:53 local time (21:53 UTC) on Tuesday, 12 January, 2010. At least 52 aftershocks measuring 4.5 or greater had been recorded. Approximately 222,500 people were dead and 196,000 were injured [10]. Typhoon Morakot was the most destructive storm of the deadly 2009 Pacific typhoon season. It was also the deadliest typhoon to affect Taiwan in recorded history. Typhoon Morakot wrought catastrophic damage in Taiwan, leaving 461 people dead and 192 others missing [11].

Because of their severity and devastation, extensive historical records on these disasters are available. This fact enabled the choices of input parameters values of PSP-G based on historical data, in case of Haiti earthquake based on data stored in Ushahidi research website [2] and wiki website [10], and in case of Morakot typhoon, based on historical data [11]. For both case studies, the basic cost of a participant was randomly generated in the range of 1–10 USD, and his/her cost is equal to the basic cost times the travelling cost, which is proportional to the traveling distance. The achievable value of each participant assigned to a region was randomly generated in the range of [5,100] percent of the value of the region.

5.1 Case study on Haiti earthquake 2010

According to the historical data [10,22], 10 counties (regions) were seriously damaged by the earthquake. They are Nord-Ouest, Nord, Nord-Est, Artibonite, Centre, Ouest, Sudest, Nippes, Grande-Anse, and Sud. Over 5,000 volunteers participated in rescue and relief operations. The total budget of the operations was 12.5 million USD. Most of the events reported by participants were classified as emergencies. For sake of simplicity, the total value of all regions is set at (22,25,000 + 1,96,000) × 20 = 8,370,000, where 20 is the threat score of emergency events. It seems reasonable to make the value of each region proportional to the number of dead and injured in the region. Table 4 lists the values of 10 counties set in this way.

In addition to BARON and BONMIN, the performance of PSP-G was compared with two more well-known online optimization solvers: AlphaECP [23] and DICOPT [24]. AlphaECP is based on the extended cutting plane (ECP) method, which is an extension of Kelley's cutting plane method. DICOPT is a program for solving mixed-integer nonlinear programming (MINLP) problems that involve linear binary or integer variables and linear and nonlinear

Table 4: Historical record and region values.

Region number	Region	Number of dead and injured	Percentage	Region value
1	Nord-Ouest	5,901	1.40	1,18,017
2	Nord	8,244	1.96	1,64,889
3	Nord-Est	293	0.07	5,859
4	Artibonite	20,883	4.98	4,17,663
5	Centre	4,394	1.05	87,885
6	Ouest	3,33,210	79.62	66,64,194
7	Sudest	32,643	7.80	6,52,860
8	Nippes	3,515	0.84	70,308
9	Grande-Anse	2,051	0.49	41,013
10	Sud	7,366	1.75	1,47,312
Total		4,18,500	100	83,70,000

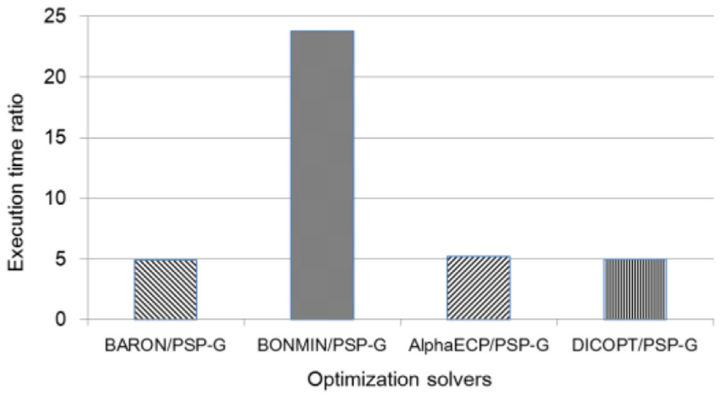


Figure 7: Execution time ratio in case study of Haiti earthquake.

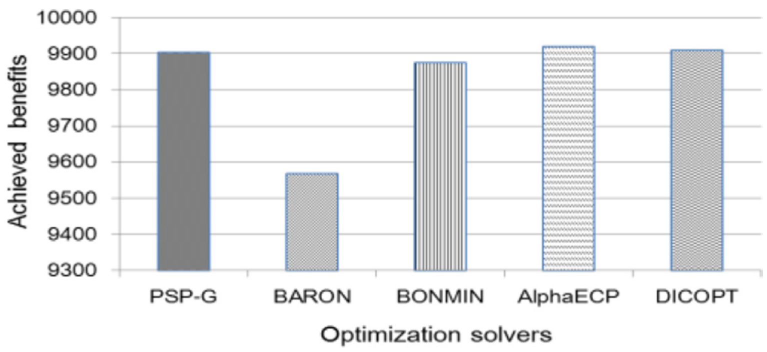


Figure 8: The achieved benefits in case study of Haiti earthquake.

continuous variables. As Fig. 7 shows, PSP-G algorithm is at least five times faster than any of the evaluated optimization solvers: The execution time of PSP-G is 0.026 seconds, making it particularly suitable at the early stage of a disaster response when multiple rounds of volunteer recruitment, and participant selection may be required to deal effectively with fast changes in threat level.

In these experiments, some solvers could not converge to the optimal solution when the number of participants was larger than 5,000. In contrast, PSP-G does not have this problem. Figure 8 shows the benefits achieved by different methods. The results show that PSP-G can deliver a near-optimal solution. In particular, PSP-G performs better than BARON and BONMIN in this case because BARON and BONMIN do not converge to the optimal solution.

5.2 Case study on Typhoon Morakot 2009

According to historical data [10], 11 counties (regions) in Taiwan were seriously damaged by Typhoon Morakot. They are Taipei, Yilan, Hualien, Taitung, Nantou, Chunghua, Yunlin, Chiayi, Tainan, Kaoshiung, and Pintung. Over 5,000 volunteers participated in rescue operations. These events are regarded as emergency events with a threat score of 20. Therefore, the total value of

Table 5: Historical record and region value.

Region number	Region	Number of the dead and injured	Percentage	Region value
1	Taipei	13	1.84	260
2	Yilan	4	0.57	80
3	Hualien	1	0.14	20
4	Taitung	18	2.55	360
5	Nantou	17	2.41	340
6	Chunghua	3	0.42	60
7	Yunlin	1	0.14	20
8	Chiayi	14	1.98	280
9	Tainan	29	4.11	580
10	Kaoshiung	559	79.18	11,180
11	Pintung	47	6.66	940
Total		706	100	14,120

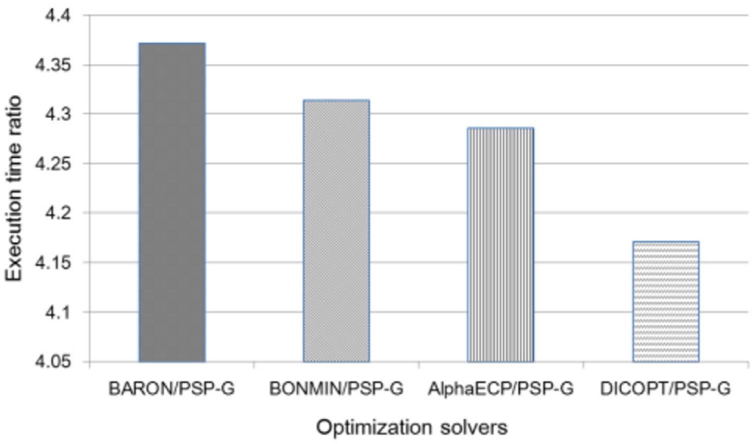


Figure 9: Execution time ratio in case study of Typhoon Morakot.

all regions is (660 + 46) × 20 = 14,120. Table 5 lists the value of each region when values are set to be proportional to the number of dead and injured in the region.

As Fig. 9 shows, PSP-G is at least four times faster than the selected optimization solvers. The execution time of PSP-G is 0.035 seconds. This fact again indicates that PSP-G is particularly useful at the early stage of a disaster when many rounds of participant selections may be required. Figure 10 shows the achieved benefits of different methods. The results show that PSP-G can also deliver a near-optimal solution. In particular, PSP-G performs better than BARON and BONMIN in this case study. The possible reason for their poor performance is that BARON and BONMIN cannot converge to the optimal solution in this case.

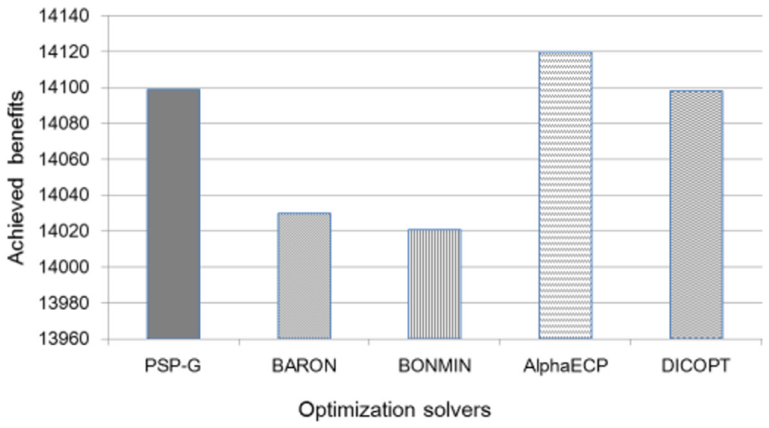


Figure 10: The achieved benefits in case study of Typhoon Morakot.

6 SUMMARY AND FUTURE WORK

This article first presented an overview of the authors' earlier work [7] that introduced a formal formulation of the PSP. The problem aims to maximize the total benefits contributed by all participants subject to a budget constraint. The PSP-G greedy algorithm as developed to solve the problem. The algorithm first calculates the B2C factor of each participant and then dispatches participants to regions in non-increasing order of their B2C. Each time the algorithm tries to maximize the total benefit of the partial assignment. According to experimental results presented in the previous section, PSP-G delivers a near-optimal solution in far less time than four well-known optimization solvers. In particular, its execution time can be reduced to only one tenth of that of the solvers.

A module implementing the PSP-G algorithm was integrated into Ushahidi, the web-based crowdsourcing support system that has been used in recent years to support emergency response operations world-wide. The enhanced system is called Ushahidi$^+$. Two case studies were conducted to demonstrate the applicability of PSP-G in real-life situations. The experiment results show that PSP-G is a practical solution and should be included in the tool kits of crowdsourcing support systems. The PSP-G module and the Ushahidi$^+$ mobile emergency APP will be released under GPL license and make it available through Ushahidi. Future work also includes enhancements of the participant selection module to provide solutions to other formulations of the PSP and implementations of the solutions will also be released and made available through Ushahidi.

ACKNOWLEDGMENTS

This work was partially supported by The Academic Sinica Project Open ISDN (Open Information System for Disaster Management) AS-102-SS-A04 and by the Taiwan National Science Council Grants NSC 100-2219-E-224-001.

REFERENCES

[1] Sahana FOSS Disaster Management System, available at http://en.wikipedia.org/wiki/Sahana_FOSS_Disaster_Management_System

[2] Ushahidi Research, available at http://en.wikipedia.org/wiki/Ushahidi

[3] Deep Horizon Oil Spill, available at http://en.wikipedia.org/wiki/Deepwater_Horizon_oil_spill and http://www.google.com/crisisresponse/oilspill/

[4] 2009 California Wildfires, available at http://en.wikipedia.org/wiki/2009_California_wildfires

[5] Typhoon Morakot Aftermath, available at http://en.wikipedia.org/wiki/Typhoon_Morakot#Taiwan_3

[6] Chu, E.T.-H., Chen, Y.L., Lin, J.-Y. & Lin, J.W.S., Crowdsourcing support system for disaster surveillance and response. *Proc. of 15th International Symposium on Wireless Personal Multimedia Communications (WPMC)*, 2012.

[7] Chu, E.T.-H, Lin, C.Y., Tsai, P.-H. & Liu, J.W.S., Participant selection for crowdsourcing disaster information. *WIT Transactions on the Built Environment*, **133**, pp. 205–215, 2013. doi: http://dx.doi.org/10.2495/dman130211

[8] BARON, available at http://neos.mcs.anl.gov/neos/solvers/go:BARON/GAMS.html

[9] Bonmin, available at http://neos.mcs.anl.gov/neos/solvers/minco:Bonmin/GAMS.html

[10] 2010 Haiti Earthquake, available at http://en.wikipedia.org/wiki/2010_Haiti_earthquake

[11] Typhoon Morakot, available at http://en.wikipedia.org/wiki/Typhoon_Morakot

[12] Generalized Assignment Problem, available at http://en.wikipedia.org/wiki/Generalized_assignment_problem

[13] Oncan, T., A survey of generalized assignment problem and its applications. *Information Systems and Operation Research*, **45(3)**, pp. 123–141, 2007. doi: http://dx.doi.org/10.3138/infor.45.3.002

[14] Penticoa, D.W., Assignment problems: a Golden anniversary survey. *European Journal of Operational Research*, **176(2)**, pp. 774–793, 2007. doi: http://dx.doi.org/10.1016/j.ejor.2005.09.014

[15] Knapsack Problem, available at http://en.wikipedia.org/wiki/Knapsack_problem

[16] Subset Sum Problem, available at http://en.wikipedia.org/wiki/Subset_sum_problem

[17] Kwon, T. & Cho, D.-H., Adaptive-modulation-and-coding-based transmission of control messages for resource allocation in mobile communication systems. *IEEE Transactions on Vehicular Technology*, **58(6)**, pp. 2769–2780, 2009. doi: http://dx.doi.org/10.1109/tvt.2008.2008651

[18] Bartoli, G., Tassi, A., Marabissi, D., Tarchi, D. & Fantacci, R., An optimized resource allocation scheme based on a multidimensional multiple-choice approach with reduced complexity. *Proc. of ICC*, pp. 1–6, 2011. doi: http://dx.doi.org/10.1109/icc.2011.5962916

[19] Chen, J., Wang, S. & Chen, C., Method research for dynamic multi-project human resource allocation based on Multidimension model. *Proc. of ICCIII*, pp. 78–81, 2011. doi: http://dx.doi.org/10.1109/iciii.2011.304

[20] NEOS, available at http://neos.mcs.anl.gov/neos/solvers/index.html

[21] Lin, C.-Y., Chu, E.T.H., Ku, L.-W. & Liu, J.W.S., Active disaster response system for smart buildings. *Sensors*, **14(9)**, pp. 17451S–17470S, 2014. doi: http://dx.doi.org/10.1109/is3c.2014.134

[22] Timeline of relief efforts after the 2010 Haiti earthquake, available at http://en.wikipedia.org/wiki/Timeline_of_relief_efforts_after_the_2010_Haiti_earthquake

[23] AlphaECP, available at http://www.neos-server.org/neos/solvers/minco:AlphaECP/GAMS.html

[24] DICOPT, available at http://www.neos-server.org/neos/solvers/minco:DICOPT/GAMS.html