中央研究院
資訊科學研究所
Institute of Information Science, Academia Sinica • Taipei, Taiwan, ROC

TR-IIS-16-003

Participant Selection Problem - Relative Performance of Five Optimization Solvers

C. Y. Lin, J. W. S. Liu and W. D. Yeh, E. T. H. Chu

Sep. 29 , 2016 || Technical Report No. TR-IIS-16-003
http://www.iis.sinica.edu.tw/page/library/TechReport/tr2016/tr16.html

Institute of Information Science, Academia Sinica

Technical Report TR-IIS-16-003

Participant Selection Problem - Relative Performance of Five Optimization Solvers

C. Y. Lin
Department of Computer Science, Tsing Hua University
Hsinchu, Taiwan
Email: gaogaolin0418@gmail.com

J. W. S. Liu and W. D. Yeh
Institute of Information Science, Academia Sinica
Taipei, Taiwan
Email: {janeliu, wdy}@iis.sinica.edu.tw

E. T. H. Chu
Dept. of Comp. Sci. and Info. Eng., Yunlin University of Science and Technology
Yunlin Taiwan
Email: edwardchu@yuntech.edu.tw

Participant Selection Problem - Relative Performance of Five Optimization Solvers

C. Y. Lin
Department of Computer Science, Tsing Hua University, Hsinchu, Taiwan
Email: gaogaolin0418@gmail.com

J. W. S. Liu and W. D. Yeh
Institute of Information Science, Academia Sinica, Taipei, Taiwan
Email: {janeliu, wdy}@iis.sinica.edu.tw

E. T. H. Chu
Dept. of Comp. Sci. and Info. Eng., Yunlin University of Science and Technology, Yunlin, Taiwan
Email: edwardchu@yuntech.edu.tw

***Abstract*—This paper presents integer linear programming formulations of four variants of the participant selection problem (PSP) that we may encounter in likely disaster scenarios. An evaluation study was carried out to quantify the relative performance of popular optimization solvers and a greedy heuristic algorithm when used to solve the PSP. The paper describes the study and presents the data.**

***Index Terms*—participant selection, generalized assignment, integer linear programming, optimization solvers, greedy algorithm, performance data**

I. INTRODUCTION

We are concerned here with the combinatorial optimization problem called the *participant selection problem* (*PSP*). A variant of PSP is a special case of the generalized assignment problem (GAP) [1,2]. We encountered the PSP when we want to select participants from available volunteers and assign the selected participants to disaster threatened regions in order to gather and report observational data from the regions.

A. Background

Typical disaster monitoring and surveillance systems (e.g., [3, 4]) use physical sensors at selected locations in disaster threatened area to capture observational data. Base on data from available sensors, the system makes appropriate response decisions. Many factors, including deployment cost and physical damages, may prevent a system from having a sufficient number of physical sensors to provide it with full coverage of the threatened area. When this happens, crowdsourcing is an effective way to get additional observational data [5, 6]. By *crowdsourcing*, we mean the use of volunteers with wireless devices and social services as mobile human sensors. The system sends human sensors to locations where physical sensors are missing or are inadequate. Their eyewitness reports can complement data from physical sensors to eliminate blind spots and improve resolution in sensor coverage.

Take the crowdsourcing support system CROSS [7] as an example. When the system finds the coverage of physical sensors inadequate, it starts a human sensor data collection process by broadcasting a call for participation to registered volunteers. The system then solves a PSP after receiving their responses: The input to the problem includes the number ρ (≥ 1) of regions $R_1, R_2, \ldots R_\rho$ in the disaster affected area. The regions have values $v_1, v_2, \ldots v_\rho$, respectively. Also given are the number π (>1) of volunteers available for selection and assignment and a benefit matrix and a cost matrix characterizing the volunteers: The element b_{ik} ($0 < b_{ik} < \infty$) of the benefit matrix and the element c_{ik} ($0 < c_{ik} < \infty$) of the cost matrix represent the benefit achievable by the selected participant P_i and the cost incurred by the participant, respectively, when he/she is assigned to the region R_k. The budget B constrains the total cost of all participants. A solution specifies whether each volunteer is selected and if selected to which region he/she is assigned. Among all solutions, the system tries to find one that maximizes the total benefit achievable by all selected participants subject to specified constraints, including that the total costs of all participants is no greater than the budget.

We formulated four variants of PSP, each of which takes into account of requirements and constraints in participant selection for some disaster scenarios. The variants are *PSP-Frugal, PSP-Reliable, PSP-Practical* and *PSP-Backup*. PSP-Frugal requires that the solution never assigns more participants to any region than needed to fully explore the region: A region is said to be *fully explored* when the total benefit achievable by all participants assigned to the region is at least equal to the value of the region; otherwise, it is said to be *partially explored.* Solutions of PSP-Frugal may leave some regions partially explored. When such solutions are not acceptable, PSP-Reliable and PSP-Practical offer good alternatives. The former admits only solutions according to which every region is fully explored. The latter admits solutions which are such that every region is either fully explored or not explored at all (i.e., is assigned no

participants). PSP-Backup reserves some volunteers as backups to handle future unexpected emergencies.

B. Objective and Contributions

Today, many optimization solvers (e.g., [8-14]) can be used to solve these variants of PSP. Given sufficient computation time, the solvers can find optimal solutions, but may be outperformed by simple heuristics when available time is limited. We present in this paper an evaluation study aiming to determine the relative performance of five popular optimization servers (i.e., Gurobi [8], MOSEK [9], XpressMP [10], CPLEX [11], and Cbc [12]) and the greedy heuristic PSP-G algorithm [15] when used to solve the PSP-Frugal and PSP-Practical variants. The reason for focusing on these variants is that they model many real-life situations more closely: By allowing solutions which assign no participants to some regions, PSP-Practical enables the system to work with limited budget. PSP-Backup is essentially the same as PSP-Frugal when participants are selected from specified available volunteers instead of selecting participants from all available volunteers.

A contribution of the paper is the formulations of the variants of PSP. Another contribution is the performance data from our extensive experiments. We use several figures of merits to compare the performance of the solvers and the heuristic algorithm along multiple dimensions. The data provide insight into characteristics of problem instances for which different methods (i.e., evaluated solvers and heuristic algorithm) can produce good solutions in available time. Based on such decision support data, a participant selection tool can better choose a method to use when presented with a PSP instance.

Following this introduction, Section II presents related work. In particular, the section describes the above mentioned optimization servers and presents reasons for choosing them over other solvers. Section III presents the four formulations of PSP. Section IV presents how parameters that define individual problem instances are generated and the rationales behind their generation. The section also defines figures of merits used to compare the performance of solvers. Section V and VI present the results produced by the solvers and PSP-G algorithm and discuss the choice(s) of solvers for different instance of the variants of PSP. Chapter VII presents our conclusions.

II. Related Work

We note that 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) [1, 2]. In turn, GAP is a generalization of the assignment problem [16].

A. Related Problems and Classical Solutions

GAP arises in numerous real-life applications, including task scheduling, resource allocation, vehicle routing, computer network, and product planning. GAP is known to be NP-hard and APX-hard to approximate it. Over the years, numerous algorithms for solving or reducing GAP have been proposed and studied. They include relaxation algorithms, branch and bound algorithms, branch and cut algorithms, and heuristic algorithms. Some algorithms for solving the problem use solutions of the knapsack problem [17] as the basis. The heuristic algorithm PSP-G [15] and the enhanced version of PSP-G to be described in Section IV are examples. In addition to survey papers such as [2, 16-20], overviews of applications and solutions of GAP can be found in many graduate theses (e.g., [21, 22]) as well.

For finite B, the PSP-Frugal variant resembles the general distributed caching problem (DCP) treated in [18]. The problem can be stated in context of a server with a set of π known requests of t ($\leq \pi$) request types and ρ cache locations. Each location has a storage limit and a bandwidth limit, each request type has a known size, and each request has a bandwidth requirement and an associated profit. The server can respond to a request from a cache location if the request type is stored at the location and the location has sufficient bandwidth to meet the bandwidth requirement of the request. The server wants to find an assignment of request types to cache locations subject to the storage and bandwidth limits of the locations and selections of requests of the types stored at each location to respond with the objective of maximizing the total profit of all responded requests.

The DCP problem is one of the so called separable assignment problems (SAP), the optimization problem of assigning objects to bins with separate constraints for each bin. It is easy to see that PSP-Frugal is not a SAP but becomes one when the budget B constraining the total cost of exploring all regions is replaced by per region budgets B_k, for $k = 1, 2 \ldots \rho$ where B_k is the budget for exploring region R_k and $\sum_{1 \leq k \leq \rho} B_k = B$. We can solve the PSP-Frugal problem approximately by first allocating the given budget to regions. There are many ways to solve the separable assignment problem, including the LP-rounding based approximation algorithm and polynomial-time local search approximation algorithm [18].

B. Optimization Solvers

As it will be evident in the next section, the variants of PSP studied here are binary integer linear programs (ILP), i.e., ILP in which every variable is constrained to be 0 or 1. Recent advances in modeling languages and software tools have produced over 60 optimization solvers. Typical PSP instances presented by crowdsourcing support systems (e.g., [7]) are small to intermediate in size (e.g., numbers of regions and volunteers are in order of tens); available time for computing a solution is in order of minutes to tens of minutes, and near-optimal solutions are often acceptable. Many of the existing optimization solvers are well suited for them. For this reason, we choose to direct our attention to selecting and identifying optimization solvers that can produce good solutions for frequently encountered types of PSP instances, rather than modifying and evaluating existing approximate and heuristic algorithms developed to solve related problems in order to make them work sufficiently well for PSP.

Specifically, among the state-of-the-art solvers, we chose to evaluate in depth the performance of the following five solvers when used to find optimal

solutions of PSP-Frugal and PSP-Practical. Hereafter, they are referred to as evaluated solvers.

- *Gurobi Optimizer* [8]: Gurobi is one of the fastest, most powerful commercial solvers for linear programming (LP), quadratic programming and mixed integer programming (MIP) problems. Gurobi supports a variety of commonly used programming and model building languages.
- *MOSEK optimization software* [9]: MOSEK is a high-performance optimization solver. Besides linear and mixed integer programming problems, MOSEK can solve quadratic and convex nonlinear programming problems and has interfaces to C, Java, MATLAB, .NET, and Python.
- *FICO ® Xpress Optimization Suite* [10]: Xpress offers a powerful and versatile solver for solving highly complex, real world LP, MIP and NLP problems. It also provides a collection of optimization algorithms for solving large-scale linear problems and mixed integer problems.
- *CPLEX* [11]: CPLEX Optimization Studio is a commercial solver designed to solve large scale linear programming problems. It was named for the simplex method implemented in the C. CPLEX is accessible through modeling systems such as AMPL (a mathematical programming language) [23] and GAMS (General Algebraic Modeling System) [24].
- *Cbc* (Coin-or branch and cut) [12]: Cbc is an open source mixed integer programming solver. As its name indicates, Cbc uses branch and cut algorithm to solve optimization problem. It can be used as a callable library or a stand-alone executable.

C. Performance Comparisons

In practice, optimization solvers may not find and return optimal solutions when the problem size becomes large and time available to find the solutions are limited. This fact motivated several previous studies on relative performance of optimization solvers, as well as the evaluation study to be reported later in the paper.

As an example of efforts in this direction, A. Neumaieret, *et al*. [25] evaluated several existing optimization software, including BARON [14], for global optimization problems and constraint satisfaction. Their goal, like ours, is to provide data on the relative performance of optimization solvers that can help us to determine the choice(s) of solvers for specific problem instances. The results of the study showed that BARON was the fastest and most robust solver: BARON was able to complete the search in over two third of the optimization problems with less than 100 variables. For larger problems, other optimization solvers may have higher success rate in finding an optimal solution than BARON, but may have longer execution time.

The benchmarks published in [26] compared the performance of Gurobi with many commercial and open source optimization solvers. The test sets include internal type and public type. The internal-type test set has over 10,000 problems. They were used to assess performance improvements achieved by Gurobi from version to version. The public type uses 2010 MIPLIB (Mixed Integer Problem Library) [27] as test sets to compare the performance against other solvers. The performance data showed Gurobi is the fastest solver for linear programming, mixed integer programming or quadratic programming models. XPRESS is a slightly slower than Gurobi on linear programming problems, and so is Cbc.

Our choices of the above listed optimization solvers for in depth evaluation are based on the benchmark performance of state-of-the-art solvers published in [28]. The test cases used are from NETLIB [29] and MIPLIB library containing a collection of real-life problems. The evaluation set 25,000 seconds as time limit. The results on benchmark performance of commercial optimization solvers (including Gurobi, MOSEK, and XPRESS and CPLEX) show that they perform better than other solvers.

III. Formulations of PSP

Specifically, the participant selection problem (PSP) can be formulated as an integer linear programming problem. The parameters of the formulations are denoted by the following notations. They are positive integers:

- V is the *objective value* and is also referred to as the *total value* of the selection.
- The threatened area contains ρ regions $R_1, R_2, \ldots R_\rho$, and their values are $v_1, v_2, \ldots v_\rho$, respectively.
- A selection is from π volunteers; They are referred to as $P_1, P_2, \ldots P_\pi$.
- For $i = 1, 2, \ldots \pi$ and $k = 1, 2, \ldots \rho$,
  - b_{ik} is the benefit achieved by P_i if he/she is assigned to region R_k.
  - c_{ik} is the cost of P_i when the participant is assigned to region R_k.
- $\{x_{ik}\}$ for $i = 1, 2, \ldots \pi$ and $k = 1, 2, \ldots \rho$, is a boolean set. $x_{ik} = 1$ indicates that P_i is selected and is assigned to R_k; $x_{ik} = 0$ otherwise.
- B is the total budget available to be spent on all selected participants.
- A selection has η groups $G_1, G_2, \ldots G_\eta$ of participants: Each participant belongs to one group according to his/her skills and the organization he/she belongs.

A. PSP-Frugal

PSP-Frugal is a variant that resembles GAP in general. Formally, the formulation of PSP-Frugal is stated below.

PSP-Frugal:

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

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

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

$$\sum_{1 \le i \le \pi} b_{ik} x_{ik} \le v_k \quad k = 1, 2, \ldots \rho \tag{4}$$

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

The total value V of the selection, as given by Eq. (1), is the sum of benefits achieved by all of the selected and assigned participants. Like all variants of PSP, the

objective of PSP-Frugal is to maximize the total value. The set $\{x_{ik}\}$ for all $i = 1, 2, \ldots \pi$ and $k = 1, 2, \ldots \rho$ gives the selection of a subset of volunteers to be participants and an assignment of the participants to regions. The inequality (3) ensures that each participant is assigned to at most one region. The constraint (4) ensures that the solution $\{x_{ik}\}$ never assigns more participants to any region than needed to achieve the full value of the region. This is why this variant is called *PSP-Frugal*. The inequality (5) says the total cost incurred by all selected participants must not be greater than the budget B.

Because of constraint (4), PSP-Frugal allows solutions in which the total benefit of all participants assigned to a region may be less than the value of the region. In other words, the numbers of participants assigned to some regions are insufficient to achieve the full values of the regions. We call such a region a *partially explored region* and the difference between the value of a region and the total benefit of all participants assigned to the region the *shortfall* of the region.

B. Other Variants

In scenarios where it is not acceptable to have some regions partially explored, PSP-Reliable is an alternative: The *PSP-Reliable* variant is the same as PSP-Frugal, except constraint (4) is replaced by

$$\textbf{PSP-Reliable}: \quad \sum_{1 \le i \le \pi} b_{ik} x_{ik} \ge v_k \quad k = 1, 2, \ldots \rho \qquad (4a)$$

PSP-Reliable guarantees that all regions in the threatened area are *fully explored* to achieve the full values of the regions. Because of this constraint, PSP-Reliable is unlike GAP and variants of GAP.

PSP-Reliable may not have a solution since solutions that satisfy constraints (2), (3) and (4a) may be too costly and hence exceeds the budget limit. A more practical alternative is called *PSP-Practical*: For this variant the constraint (4) is replaced by

$$\textbf{PSP-Practical}: \quad \sum_{1 \le i \le \pi} b_{ik} x_{ik} \ge v_k \text{, or } = 0 \quad k = 1, 2, \ldots \rho \qquad (4b)$$

The variant is said to be practical because it always has feasible solutions. A solution of PSP-Practical aims to maximize the total benefit of all the selected participants under the condition that each region is either to be fully explored or not explored at all. PSP-Practical resembles 0/1 assignment in general.

While PSP-Frugal may have the shortfall problem, PSP-Practical may have the waste problem: Constraint (4b) allows assignments according to which the total benefits of some regions exceed the values of the regions. In other word, there may be some region R_k, for which $\beta_k = \sum_{1 \le i \le \pi} b_{ik} x_{ik} > v_k$. We call the difference $\beta_k - v_k$ the *waste* of region R_k.

The variants defined above assume that all of volunteers are eligible for selection. Their solutions may leave no participants available to handle unexpected emergencies. This fact motivated the condition stating the need to reserve some volunteers as backups. In this case, we divide volunteers into η groups G_1, G_2, ... G_η according to some criteria (e.g., skills such as first aid, fire control, etc.) PSP-Backup is defined by the following constraint in addition to (1) - (5).

$$\textbf{PSP-Backup:} \quad \sum_{Pi \in Gj} (1 - \sum_{1 \le k \le \rho} x_{ik}) \ge \psi_j \,, \quad j = 1, 2, \ldots \eta \qquad (6)$$

ψ_j in the right hand side of the inequality denotes the required number of reserved volunteers in group G_j. The sum on the left hand side is the number of volunteers in group G_j who are not selected.

IV. Performance Evaluation Methods

We conducted simulation experiments to determine the relative performance of popular optimization solvers Gurobi, MOSEK, Xpress, CPLEX, and Cbc [8-12] when used to solve the PSP-Frugal and PSP-Practical variants of the participant selection problem. In particular, we aim to get data on their relative performance for different distributions of the parameters that define individual problem instances. We also aim to determine how the simple greedy heuristic algorithm PSP-G [15] and an enhanced version of PSP-G algorithm perform relative to the evaluated solvers.

A. Parameter Generation and Rationales

In each experiment, we invoked the solvers and PSP-G algorithm to find solutions of a synthetic instance of PSP. We divide the parameters that specify each instance into two subsets: a task assignment specification (TAS) and a participant selection specification (PSS). The values of the parameters in each TAS and PSS were generated randomly and independently.

Specifically, the TAS of a problem instance includes the number of regions and the values of the regions. PSS includes the number of volunteers, the benefits and costs of each volunteer, and the total budget. For each simulation experiment, we generated 3,000 sets of TAS and PSS samples, one for each simulation run. Performance data collected during the experiments indicated that the sample size is large enough to make the statistical error in each of the estimated performance measure less than 2% with 95 percent confidence.

Number of Regions: We chose the range of ρ (the number of regions) based on Taiwan geography: A threatened area is a county (e.g., Yunlin County) or city (e.g., Taipei City and Hsinchu City). Each region is a district in the county/city. We started from the minimum number of region being 5, since areas with fewer than 5 regions are too small for our purpose. The maximum number of districts is 38; that is the number of districts in Kaohsiung City. So, for each simulation experiment, we started from $\rho = 5$ and multiplied ρ by 2 or incremented ρ by 10 per step until $\rho = 40$.

Region Values: The total benefit achieved by the participants assigned to a region should be limited by the value of the region. Hence, region values affect the amounts of resources required to explore the threatened area. In our previous efforts to evaluate the PSP-G algorithm, we experimented with setting region values according to the population, productivity (e.g. tourism consumption, farming and fishery productions), transportation development, etc. of the districts [7] or

based on the number of injuries and values of property damages incurred in past disasters [15]. Region values thus chosen do not provide complete coverage of the range and variations of the parameter. So, we chose to generate region values randomly from distributions listed in Table 1: In some experiments, we selected region values randomly from uniform distributions with small coefficients of variation (CV). The regions with values thus selected are said to be *similar*. In other experiments, region values were selected from the exponential and multi-modal distributions with larger CV. In these cases, regions are said to be *dissimilar*.

TABLE I. RANGES AND DISTRIBUTIONS OF PARAMETERS

Parameters	CV	Ranges	Distributions
Region values	< 0.1	[7000, 8000]	Uniform
Region values	~0.3	[4000, 11000]	Uniform
Region values	1.0	[100, 15000]	Truncated exponential
Region values	> 1.0	[100,1500] [6750,8250] [13000, 15000]	Multi-modal uniform
Benefits	~ 0.5	$[1,\alpha]$	Uniform
Costs	~0.5	$[1, \gamma]$	Uniform

Benefit and cost values: We exploited the relationships among the parameters in TSS and PSS in the generation of benefits and costs of volunteers. In particular, for any meaningful problem instance, the benefit values of volunteers should be chosen relative to the values of regions. The total budget B should be set based on the costs of the volunteers. This is the rationales behind the generation of benefit and cost values of the volunteers.

Specifically, for each simulation experiment, we first generated the number ρ of regions and values v_k of the regions randomly as described above. We then computed $\alpha = \sum_{1 \le k \le \rho} v_k / \pi$ and $\gamma = B / \pi$. α can be thought of as the average benefit achievable by the volunteers if they all are selected. For a budget B, γ can be thought of as the average of available budget per volunteer. These values are used to set the ranges of the uniform distributions from which benefit and cost values in PSS were generated randomly. This fact is summarized by the last two lines in Table 1.

Budget: The budget B constrains the numbers and types of participants selected and assigned to regions. We divide the available budget into five levels, hereafter referred to as B = 20%, 50%, 80%, 100% and 200%. In the severely budget-poor case, the available budget can cover only 20% of the average total cost of all volunteers. In the budget-constrained and budget-rich cases, the available budget can cover 80% or 100% of the total cost of available volunteers, respectively. Setting B = 200% in essence removes the budget constraint.

B. Figures of Merits

We measure the merit of a solution $\{x_{ik}\}$, and hence the solver or algorithm that produces the solution, by the figures of merits defined below:

- *Objective value* of the solution $\{x_{ik}\}$ is given by (1). This is the criterion most commonly used to compare solutions.
- The *total number of selected participants* is given by $\sum_{1 \le i \le \pi} \sum_{1 \le k \le \rho} x_{ik}$, i.e., sum over all elements of $\{x_{ik}\}$. A good solution may use more participants with less total cost. However, in most situations, we consider solutions that select fewer participants to be better solutions.
- *Total cost of selected participants* is given by the sum $\sum_{1 \le i \le \pi} \sum_{1 \le k \le \rho} c_{ik} x_{ik}$. A solution incurring less total cost is clearly better than the ones incurring higher total costs.
- The *execution time* of a solver (or algorithm) refers to the amount of time required by the solver (or algorithm) to return a solution. In all the experiments, we set an upper limit of time that each optimization solver is allowed to take to return a solution. A solver may return a bad solution or no solution when it requires more than the allowed time to find an optimal solution.
- The *amount of shortfall of a region* R_k incurred by a solution of an instance of PSP-Frugal measures the degree the region is partially covered. It is given by $v_k - \sum_{1 \le i \le \pi} b_{ik} x_{ik}$. A good solution is one that keeps the total amount of shortfall small.
- The *waste of a region* R_k incurred by a solution of an instance of PSP-practical tells us the degree to which extra participants were assigned to the region beyond what is needed to fully explore the region. It is given by $\sum_{1 \le i \le \pi} b_{ik} x_{ik} - v_{k.}$. We use the total waste of all regions as a figure of merit.

For ease of comparison, we normalize the value of each figure of merit by the corresponding maximum possible value: They are, respectively, the total value of all regions, total number of volunteers, total cost of all volunteers, the maximum time limit set for the solvers and total value of all regions. Except for where it is stated otherwise, the maximum time limit for the solvers is 1000 seconds. The values of normalized figures of merit depicted by all figures presented hereafter are averages of results obtained from 3000 simulation runs. The 95% confidential intervals are within two percent of the corresponding average values.

C. PSP-G Algorithm

We also compare the performance of the above mentioned optimization solvers against the performance of two versions of greedy heuristic PSP-G algorithm. The original version of PSP-G algorithm is described in [15]. Designed to solve the PSP-Frugal variant, it is essentially a first-fit non-increasing bin-packing algorithm: The decision on participant selections and assignments are made in non-increasing order of the benefit-to-cost (B2C) ratios $Q_{i,k} = b_{ik} / c_{ik}$ of participant P_i and region R_k. The description of the original version is omitted due to space limitation and its similarity with the *enhanced PSP-G algorithm*, which is described in Figure 1. The enhancement is necessary to make the algorithm applicable to the PSP-Practical variant.

Pseudo code of enhanced PSP-G algorithm

Inputs: TAS and PSS parameters of the problem *instance*

Outputs: β_k, $k = 1, 2, \ldots, \rho$; objective value $\sum_k \beta_k$
$\{x_{ik}\}$: The selections and assignments of volunteers

Local variables:
β_k : Current total benefit of participants assigned to R_k ;
C_k : Current total cost of participants assigned to R_k ;
ϕ : The remaining budget;
Q_k: List of B2C ratios $Q_{i,k}$ of volunteer P_i to region R_k ;
L: List of region ratios $L_k = \sum_{1 \le i \le \pi} Q_{i,k}$ of all regions ;

Initialization

1. Initialize $x_{ik} = 0$ for $i = 1, 2, \ldots, \pi$ and $k = 1, 2, \ldots, \rho$;
2. Set $\phi = B$; set β_k *and* $C_k = 0$, $k = 1, 2, \ldots, \rho$;
3. For each region R_k, $k = 1, 2, \ldots \rho$, calculate *B2C* ratios $Q_{i,k} = b_{ik} / c_{ik}$, for $i = 1, 2, \ldots, \pi$, of all volunteers and put the ratios in the list Q_k in non-increasing order;
4. Compute region ratios $L_k = \sum_{1 \le i \le \pi} Q_{i,k}$, $k = 1, 2, \ldots, \rho$;
5. Sort list L *of* L_k in non-increasing order;

Selection and Assignment

6. **while** (L is not empty) **do**
7. Get k ; // The region ratio L_k is at the head of list L.
8. **while** (list Q_k is not empty) **do**
9. Get P_i **;** // Q_{ik} is at the head of Q_k
10. **if** (P_i is not selected and $\beta_k < v_k$ and $C_k + c_{ik} <= \phi$)
11. β_k += b_{ik}; C_k += c_{ik} ; $x_{ik} = 1$;
12. **end if**
13. remove Q_{ik} ;
14. **end while**
15. ϕ −= C_k;
16. **if** ($\beta_k < v_k$)
17. **for** $i = 1, 2, \ldots, \pi$, $x_{ik} = 0$;
18. $\beta_k = 0$; ϕ +=C_k;
19. **end if**
20. Remove L_k from the head of list L;
21. **end while**
22. return outputs;

Figure 1. Pseudo-code description of the enhanced PSP-G algorithm

Similar to the original version, the enhanced version also works with benefit to cost (B2C) ratios of volunteers to regions, computed in line 3. Both versions use the ratios to decide the order in which volunteers are considered for selection and assignment (lines 8-14). A major difference between the versions is that the enhanced version focuses on one region at a time in non-increasing order of region ratios: The *region ratio* L_k of each region R_k is the sum of the B2C ratios over all volunteers if they are assigned to R_k (line 4). When selecting participants for the region R_k with the largest region ratio in the list L (line 7), the enhanced version uses the B2C list Q_k for the region R_k (lines 8-14) , rather than the global B2C list for all regions used by the original version. Consequently, the enhanced version has a lower complex (i.e., $O(\rho\pi \log \pi)$) than the complexity of the original version (i.e., $O(\rho\pi \log \rho\pi)$).

Another difference between the versions is the selection rules (lines 10-12). The enhanced version decides to select a participant for a region if he/she is still available for selection and total benefit of all participants already assigned to the region is still less than the value of the region. In contrast, the original version is for PSP-Frugal; it selects and assigns a participant to a region only if the sum of his/her benefit and the current total benefit of the region does not exceed the value of the region, thus making sure that the total benefit of every region is at most equal to region value.

Lines 16-20 in Figure 1 ensures that the 0/1 constraint required by PSP-Practical is satisfied: If the total benefit β_k of the region R_k is not at least equal to its value, the participants assigned to the region are freed and their costs is returned to the remaining budget, thus leaving the region R_k without any participants.

D. GAMS and NEOS Server

To evaluate the performance of optimization solvers, the first step is to rewrite the formulations of PSP-Frugal and PSP-Practical in terms of a general algebraic modeling language so that they can be processed by the solvers. For this purpose, we use the modeling language provided by the General Algebraic Modeling System (GAMS) [24]. As stated earlier, GAMS is a high-level modeling system for mathematical programming and optimization. It provides not only a compiler that translates the formulations of linear, mixed integer linear, nonlinear and quadratic programs into inputs required by the solvers, but also accesses to the solvers.

Due to limitation on the number of constraints and variables and number of nonzero elements allowed by the free license, demo version of GAMS, we used it together with the NEOS Server [14] and implemented a tool for submitting GAMS model files to NEOS server. NEOS Server is a free internet-based service for solving numerical optimization problems. It provides access to more than 60 state-of-the-art solvers in more than a dozen optimization categories and offers a variety of interfaces for accessing the solvers. Sample code of software tools for creating and submitting jobs to the server and retrieving and parsing solutions returned by the solver can be found in [21]. For each submission, we can specify the optimization solver to be used and upper limit of time the solver can take to return a solution, while providing the solver with an external file containing data that specify the model and instance of PSP to be solved.

V. RELATIVE PERFORMANCE FOR PSP-FRUGAL

We now present the data on the relative performance of the evaluated methods (i.e., the optimization solvers [8-12] and the original version of PSP-G algorithm [15]) when they are used to solve instances of PSP-Frugal. The data were obtained via simulation experiments. Four data sets containing parameters that specify representative instances of PSP-Frugal were used for this purpose. The data sets differ in the underlying distributions from which values of parameters were selected randomly: Table 1 lists the distributions.

As stated earlier, the purpose of this evaluation is to obtain insight and data to support the choices of solvers and heuristics used to solve the problems instances that may arise in real-life scenarios. For this purpose, we measure the merit of a solution produced by a solver or algorithm, hence the merit of the solver or algorithm used to produce the solution, by the figures of merit defined in

Section IV. Each of the plots presented in this and the next section shows the dependency of a specified normalized figure of merit on the number ρ of regions, typically for values of ρ equal to 10, 30 and 40, corresponding to small to large threatened areas. The plots are parameterized by available budget, typically for B equals to 50%, 80% and 100%, corresponding to budget-poor, budget-constrained and budget-rich scenarios, respectively. Due to space limitation, only representative plots are presented here. Plots that are similar to the ones included here and plots for cases where all solvers and the PSP-G algorithm have similar performance are omitted. They can be found in [21].

A. Case of Similar Regions

Again, we say that regions with values selected from uniform distributions listed in the first two rows of Table 1 are similar. The results reported by Figures 2, 3 and 4 are for the case where the coefficient of variation (CV) of the underlying uniform distribution is 0.3; the results for the case of CV = 0.1 are similar and hence omitted.

Objective Values: According to objective values, all evaluated solvers have similar performance for different number of regions and available budget. The normalized objective values achieved by all solvers increases from 0.58 – 0.65 for B = 50% to 0.89 – 0.9 for B = 80%. With budget thus constrained, PSP-G algorithm achieved a smaller objective value, but the difference is within 5%. When B = 100%, all methods achieve the best possible normalized objective value of 1.0.

Number of selected participants: Figure 2 shows the normalized number of selected participants for budget equal to 80% and 100% . (We omit the plots for B equal to 50%. When the budget is so severely constrained, the solvers and PSP-G algorithms use essentially the same number of participants.) One can see that with B = 80%, the budget constraint the selection and assignment of participants. All evaluated solvers and PSP-G algorithm perform similarly along this dimension. The numbers of participants used by these methods differ noticeably when available budget is 100%: Part (b) of Figure 2 shows that the solution from MOSEK use approximately 10-20% fewer participants while achieving the same objective value as other solvers and PSP-G algorithm.

The total cost of select participants: Figure 3 shows the normalized total cost of selected participants incurred by solutions produced by the evaluated solvers and PSP-G algorithm for B = 80% and 100%. We note that in both cases, the total cost incurred by solutions produced by PSP-G algorithm decreases with number of regions. The algorithm also produces selections that incur less total costs than solutions produced by evaluated solvers, in some cases by over 30%. The reason is that PSP-G algorithm makes selection decisions in non-increasing order of benefit to cost (B2C) ratios of participants. In contrast, B2C ratios are not considered by the solvers.

Execution time: Figure 4 shows that the execution time of evaluated methods for B equals to 50% and 80%. (The plot for the case of B = 100% is similar to the plot for B = 80% and hence is omitted.) As expected, PSP-G algorithm has the smallest execution time. MOSEK and Cbc have longer execution time than other solvers when available budget is under 100%. MOSEK has the longest execution time when available budget is under 50%, and Cbc has longest execution time when available budget is over 80%. In general, Cbc has the longest execution time, and PSP-G algorithm has the smallest execution time in all simulation experiments.

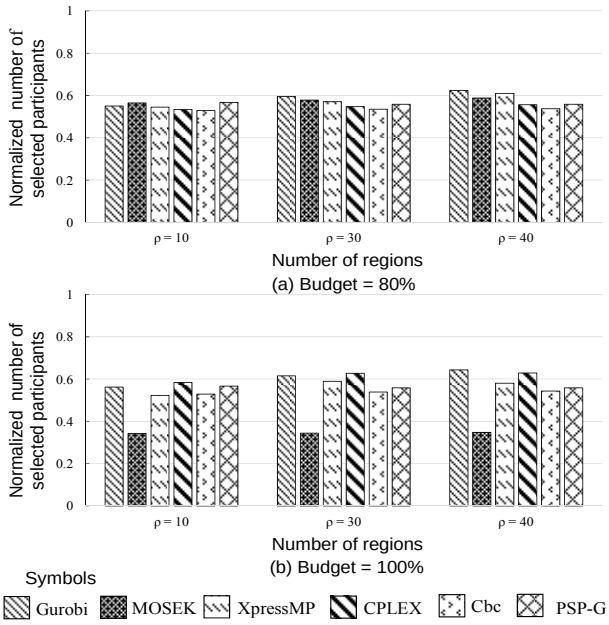

Figure 2. Normalize number of selected participants for the case of PSP-Frugal and similar regions

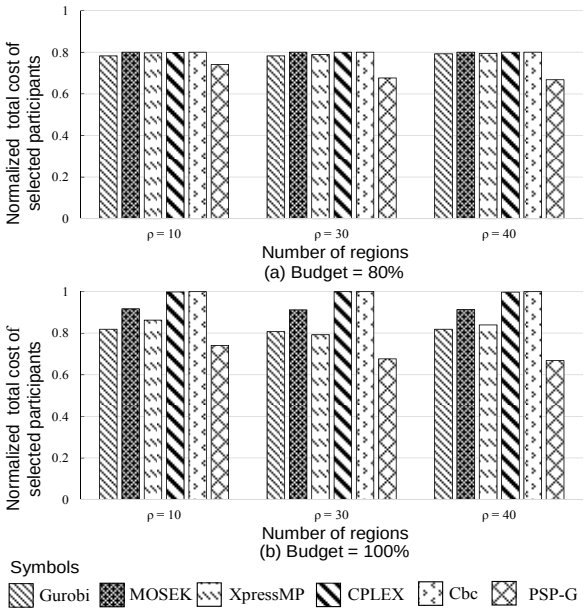

Figure 3. Normalized total cost of selected participants for the case of PSP-Frugal and similar regions

Amounts of shortfall: The amount of shortfall incurred by a solution for region R_k equals to $v_k - \beta_k$ where β_k equals to the total benefit of all participants assigned to the region. Hence, the normalized total shortfall is equal to one minus the normalized objective value. The solutions produced by all evaluated solvers

have comparable amounts of total shortfalls, while the solution produced by PSP-G algorithm has a slightly larger amount of shortfall.

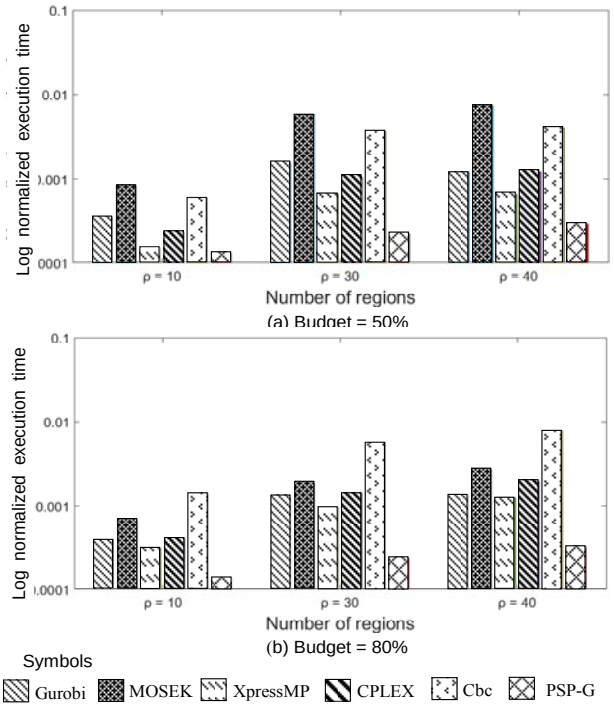

Figure 4. Normalized execution times for the case of PSP-Frugal and similar regions

B. Case of Dissimilar Regions

We also experimented with PSP-Frugal problem instances with dissimilar region values. The values were generated randomly from the exponential distribution (with CV = 1.0) and multi-modal distribution (with CV > 1) listed in rows 3 and 4 in Table 1. From the results of all experiments on instances of PSP-Frugal, we can conclude that for all practical purposes, the relative performance of evaluated solvers and PSP-G algorithm is insensitive the distribution of region values. So, the discussions presented above for the case of similar regions by and large apply to the case of dissimilar regions as well. For example, the normalized objective values and total shortfalls of solutions produced by all evaluated solvers are essentially the same. Again, the solution found by the PSP-G algorithm has a slightly smaller total value (and larger shortfall) and the value increases with number of regions.

Performance data on number of selected participants obtained from experiments on problem instances with dissimilar region values exhibit the same behavior as the data for the case of similar regions. Figure 5(a) depicts the normalized selected number of participants for the case of dissimilar regions and B equals 100%. Comparing this plot with the one shown in Figure 3(a), we see again that the solutions returned by MOSEK use the fewest number of participants when available budget is 100%, while the solutions from CPLEX and Cbc use larger number of participants. Figure 5(b) depict the plots for normalized total costs incurred by solutions produced by the evaluated solvers and PSP-G algorithm for B = 100%. As expected, the PSP-G algorithm has the best performance from the perspective of total cost; solutions from CPLEX and Cbc incur the highest total costs to select participants.

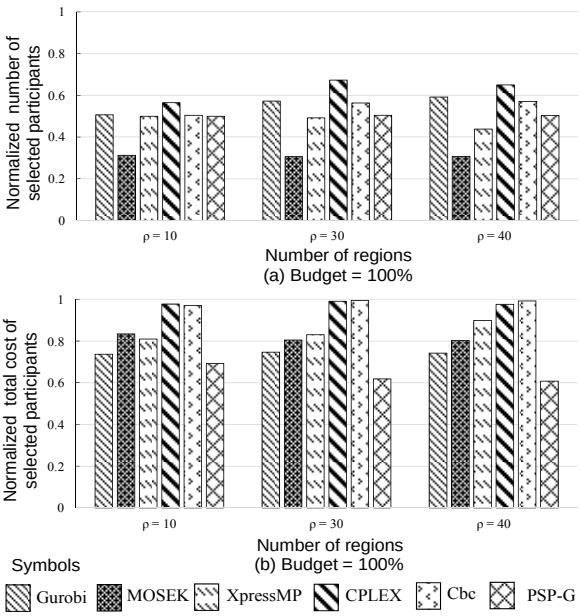

Figure 5. (a) Normalized number and (b) normalized total cost of selected participants for PSP-Frugal and dissimilar regions

VI. RELATIVE PERFORMANCE FOR PSP-PRACTICAL

We also assessed the relative performance of the evaluated solvers and the enhanced PSP-G algorithm when they were applied to solve instances of PSP-Practical: We used the four sets of problem instances described in Section IV in this series of simulation experiments as well. In addition to objective values, total number and cost of selected participants and execution time taken by the solvers and the algorithm, we also compared these evaluated methods according to the total waste incurred by the solutions produced by them.

Objective values: The data obtained from our experiments on PSP-Practical also indicate that the relative performance of evaluated methods is not sensitive to the coefficient of variation of the distribution from which region values were selected. This statement is particularly true from the perspective of objective values. The normalized objective values achieved by solutions produced by all evaluated solvers and the enhanced PSP-G algorithm range from approximately 0.95 to 1.0 when the available budget is equal to 80% and 100%. The only exception is that the normalized objective value achieved by MOSEK for ρ = 40 similar regions and B = 100% budget is only 0.38. The reason is that MOSEK could not return a good solution within the available time of 1000 seconds. When we extended the time limit to 2,000 seconds, the normalized objective value achieved by MOSEK grew to 0.95.

Total number and cost of selected participants: Figure 6 and 7 show the normalized total number and normalized total cost of selected participants for B =

100%, respectively. (Data for B = 80% exhibit similar behavior.) In general, the solutions from almost all evaluated methods use slightly fewer participants in case of dissimilar regions. Relatively, the solutions produced by MOSEK and the enhanced PSP-G algorithm use more participants while solutions from Gurobi and XpressMP use fewer participants.

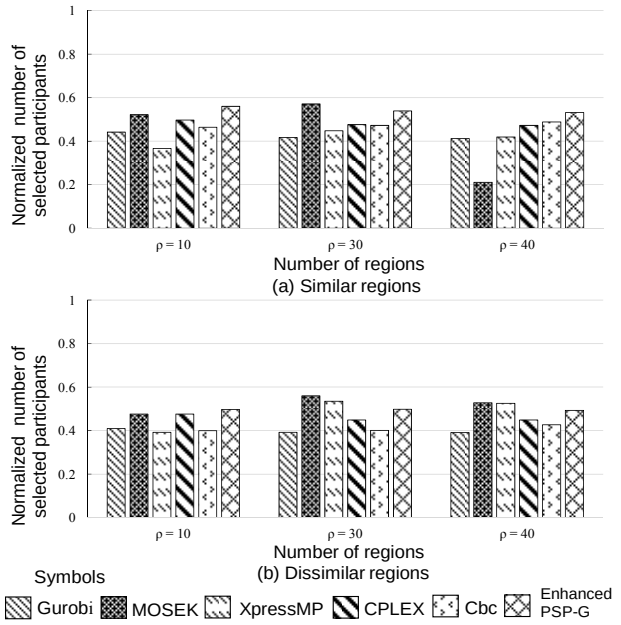

Figure 6. Normalized total number of selected participants for the case of PSP-Practical and B = 100%

Figure 7 shows the normalized total cost of selected participants incurred by solutions obtained by the evaluated solvers and the enhanced PSP-G algorithm. One can see that in terms of this figure of merit, all the methods perform slightly better for dissimilar regions than for similar regions. The enhanced PSP-G algorithm achieves the least total cost among all evaluated methods, as it is for PSP-Frugal problem instances.

For both figures of merit, an exception to the general observations occurs for MOSEK with B = 100% and ρ = 40 similar regions. Again, this happened when the solver was given only 1000 seconds to compute and was not able to return a good solution.

Execution time: Figure 8 shows the log of normalized execution time used by the evaluated solvers and the enhanced PSP-G algorithm to return their solutions of instances of PSP-Practical. Similar to the data on other figures of merit, the evaluated solvers and the enhanced PSP-G algorithm all perform slightly better for the case of dissimilar regions. The execution time taken by MOSEK is the longest in all simulation experiments, while the enhanced PSP-G algorithm takes significantly short time to return as good solutions as the solvers.

Waste of region: Every solution of PSP-Practical must either satisfies the constraint $\beta_k = \sum_{1 \leqslant i \leqslant \pi} b_{ik}\, x_{ik} > v_k$, or $\beta_k = 0$. The total waste of all regions equals to $\sum_{1 \leqslant k \leqslant \rho} \beta_k - v_k$. Performance data shows that the normalized total waste of regions incurred by solutions produced by the evaluated solvers and the enhanced PSP-G algorithm is at most 6% of total region value when B = 100%.

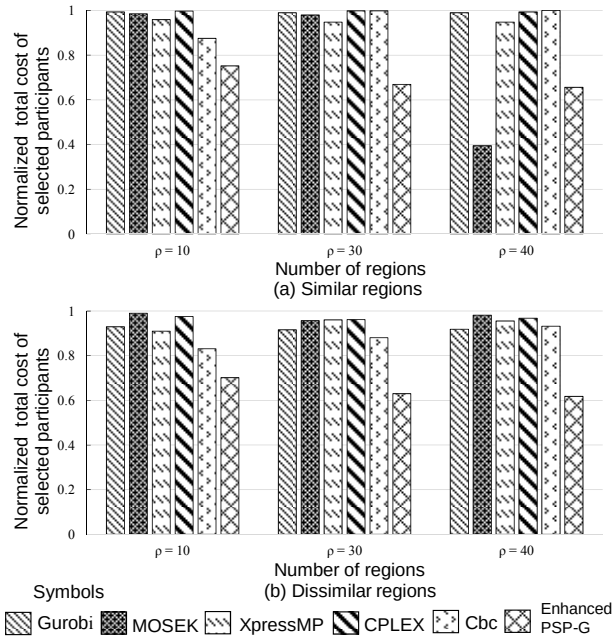

Figure 7. Normalized total cost of selected participants for the case of PSP-Practical and B = 100%

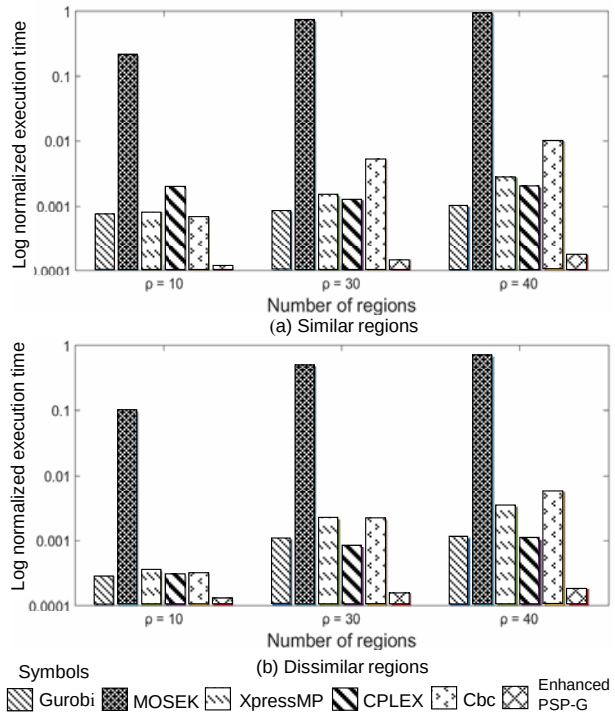

Figure 8. Log normalized execution times for the case of PSP-Practical and B = 100%

VII. Summary and Conclusion

The previous sections described four variants of the PSP (Participant Selection Problem): PSP-Frugal, PSP-Reliable, PSP-Practical and PSP-Backup. We counter the problem when we try to select participants from π of volunteers and assign the selected participants to ρ disaster threatened regions to collect data. Each volunteer is characteristics by ρ benefit values and ρ cost values he/she can achieve and incur if assigned to the regions.

The variants take into account different requirements and constraints in participant selection in difference disaster scenarios. PSP-Frugal is a variant of the well-known generalized assignment problem (GAP) [7]. The other variants are new formulations.

The evaluation study presented in the paper aims to determine the relative merits of five popular optimization solvers when used to find solutions of PSP-Frugal and PSP-Practical. The solvers are Gurobi [8], MOSEK [9], XpressMP [10], CPLEX [11], and Cbc [12]. We also evaluated the performance of two versions of the PSP-G algorithm for PSP-Frugal and PSP-Practical. The performance data reported here were obtained via simulation using four sets of problem instances defined by parameters selected randomly from four different distributions. They intend to model similar and dissimilar regions and different disaster scenarios.

As expected, when compared according to the object values achieved by their solutions, the evaluated solvers have essentially the same performance. PSP-G algorithm achieves a slight smaller objective value, but the difference is within a few percents, insignificant for most practical purposes. Another commonly used figure of merit is execution time. According to this criterion, PSP-G algorithm is significantly more advantageous. MOSEK and Cbe have the longest execution time. In the case of 40 similar regions, MOSEK even took more than the allowed 1000 time limit to produce a good solution.

We also compared the solvers and PSP-G algorithm by the total number and cost of selected participants. The solutions provided by the PSP-G algorithm consistently incur the least total cost. The reason is that the algorithm uses the benefit to cost ratios of volunteers to guide its selection and assignment decisions. In contrast, the cost factors are not taken into account by the evaluated solvers. MOSEK uses fewer participants to achieve the same objective values.

We plan to use data obtained from the study to support the choices of solvers or algorithm by the participant selection tool in CROSS [7] when it is presented with PSP instances. All things considered, the tool can choose to use the PSP-G algorithm for most problem instances.

ACKNOWLEDGMENT

This work was supported by the Taiwan Academia Sinica, Sustainability Science Research Program, thematic project "Disaster Resilience through Big Open Data and Smart Things."

REFERENCES

[1] Generalized Assignment Problem, (last retrieved: June 19, 2016) http://en.wikipedia.org/wiki/Generalized_assignment_problem

[2] T. Oncan, "A survey of generalized assignment problem and its applications," *Information Systems and Operation Research*, Vol. 45, No. 3, pp. 123-141, August 2007.

[3] S. Ibrahim, "A comprehensive review on intelligent surveillance systems," Communications in Science and Technology, vol. 1, pp. 7-14, 2016.

[4] *Multisensor Surveillance Systems: The Fusion Perspective surveillance systems,* G. L. Foresti, C. S. Regazzoni, P. K. Varshney, ed. Springer, 2003.

[5] M. N. K, Boulos, *et al.*, "Crowdsourcing, citizen sensing and sensor web technologies for public and environment health surveillance and crisis management: trend, OGC standards and application examples," *International Journal on Health Geographics*, pp 1-67, December 2011.

[6] List of Crowdsourcing Projects, http://en.wikipedia.org/wiki/List_of_crowdsourcing_projects (Last retrieved June 2016)

[7] E. T.-H. Chu, Y.-L. Chen, J. W. S. Liu and J. K. Zao, "Strategies for crowdsourcing for disaster situation information," *WIT Trans. on the Built Environment*, Vol.119, pp. 257-269, 2011.

[8] Gurobi Optimizer, http://www.gurobi.com/products/gurobi-optimizer, (Last retrieved: June 2016).

[9] MOSEK, https://www.mosek.com/ (Last retrieved: 2016)

[10] FICO ® Xpress Optimization Suite, http://www.fico.com/en/products/fico-xpress-optimization-suite, (Last retrieved: June 2016).

[11] IBM ILOG CPLEX Optimization Studio: http://www-03.ibm.com/software/products/en/ibmilogcpleoptistud, (Last retrieved: June 2016).

[12] J. Forrest and R. Lougee-Heimer, "Cbc user guide," http://www.coin-or.org/Cbc/cbcuserguide.html, (Last retrieved: June 2016)

[13] List of optimization software, https://en.wikipedia.org/wiki/List_of_optimization_software, (Last retrieved: June 2016).

[14] Baron Software, http://archimedes.cheme.cmu.edu/?q=baron, (Last retrieved: June 2016)

[15] E. T.-H Chu, C.-Y. Lin, P. H. Tsai and J. W. S. Liu, "Design and implementation of participant selection for crowdsourcing disaster information," *International Journal on Safety and Security Engineering,* vol. 3, No. 1, pp. 48-62, March 2015

[16] D. W. Penticoa, "Assignment problems: a golden anniversary survey," European Journal of Operational Research, Vol. 176, No 2, pp. 774–793, January 2007.

[17] http://en.wikipedia.org/wiki/Knapsack_problem, Knapsack Problem, (Last retrieved, June 2016)

[18] L. Fleischer, *et. al*., "Tight approximation algorithms for maximum general assignment problems," in Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 611-620, 2006.

[19] D. G. Cattrysee and V. N. Van Wassenhove, "A survey of algorithms for the generalized assignment problem," *European Journal of Operations Research,* August 1992.

[20] B. Meindl and M.Templ, "Analysis of commercial and free and open source solvers for linear optimization problems," Report from Essnet Project on Common Tools and Harmonized Methodologies for SDC in the ESS, 13 pages, February 2012.

[21] C. Y. Lin, "Participant Selection Problems," MS thesis, Department of Computer Science, National Tsing-Hua University, Hsinchu, Taiwan, August 2016.

[22] I. Beniaminy, "Approximate algorithms for generalized assignment problems," MS thesis, Department of Math and CS, Open University of Israel, June 2005

[23] R. Fourer, D. M. Gay, and B. W. Kernighan, " AMPL: a mathematical programming language," *Management Science*, Vol. 36, pp. 519-554, 1990.

[24] GAMS: General Algebraic Modeling Systems, https://www.gams.com/, (Last retrieved: June 2016)

[25] A. Neumaier, O. Shcherbina, W. Huyer, and T. Vinko, "A comparison of complete global optimization solvers," Mathematical Programming, vol.103, no 2, pp. 335-356, 2005.

[26] GurobiOptimization [42] http://www.gurobi.com/pdfs/benchmarks.pdf

[27] Mixed Integer Problem Library, http://miplib.zib.de/ (Last retrieved, June 2016).

[28] H. Mittelmann, Benchmarks for optimization software, http://plato.asu.edu/guide.html, (Last retrieved: August 2016).

[29] Netlib, https://en.wikipedia.org/wiki/Netlib, (Last retrieved: June 2016