

# **Empowering Volunteer Crowdsourcing Services: A Serverless-assisted, Skill and Willingness Aware Task Assignment Approach for Amicable Volunteer Involvement**

RIYA SAMANTA, BISWAJEET SETHI, SOUMYA K. GHOSH, Indian Institute of Technology Kharagpur , India

Volunteer crowdsourcing (VCS) leverages citizen interaction to address challenges by utilizing individuals' knowledge and skills. Complex social tasks often require collaboration among volunteers with diverse skill sets, and their willingness to engage is crucial. Hence, matching tasks with the most suitable volunteers remains a significant challenge. VCS platforms face unpredictable platform demands in terms of tasks and volunteers requests, complicating resource requirement prediction for volunteer to task assignment process. To address these challenges, we introduce the Skill and Willingness-Aware Volunteer Matching (SWAM) algorithm, which allocates volunteers to tasks based on skills, willingness, and task requirements and developed a serverless framework to deploy SWAM. Our method outperforms conventional solutions, achieving a 71% improvement in end-to-end latency efficiency. We achieved a 92% task completion ratio and reduced task waiting time by 56%, with an overall utility gain 30% higher than state-of-the-art baseline methods. This framework will contribute to generating amicable volunteer and task matches, supporting grassroots community coordination and fostering citizen involvement, ultimately contributing to social good.

**CCS Concepts:** • Human-centered computing → Collaborative and social computing theory, concepts and paradigms; • Information systems → Crowdsourcing; • Cloud computing;

**Additional Key Words and Phrases:** Volunteer Crowdsourcing, Skill-oriented, Willingness, Task Assignment, Serverless Computing

## **ACM Reference Format:**

Riya Samanta, Biswajeet Sethi, Soumya K. Ghosh . 2024. Empowering Volunteer Crowdsourcing Services: A Serverless-assisted, Skill and Willingness Aware Task Assignment Approach for Amicable Volunteer Involvement. 1, 1 (August 2024), 12 pages. <https://doi.org/10.1145/nnnnnnn.nnnnnnn>

## **1 Introduction**

Crowdsourcing has become a powerful catalyst for social change and tackling complex societal issues. By leveraging the collective intelligence and resources of a diverse group of people, crowdsourcing initiatives can create innovative solutions to challenges such as situational awareness, environmental sustainability, and public health. Crowdsourcing enables global collaboration in many ways, such as using collective knowledge to set sustainable goals, involving citizens in monitoring biodiversity, or mobilizing volunteers to map crisis areas for humanitarian aid. Moreover, it empowers communities to actively participate in decision-making processes as in democrat [30].

Serverless computing, a cloud computing execution model, with its on-demand scalability and pay-per-use protocol, has immense potential to contribute to social good in various ways. By eliminating the need for provisioning and

---

Author's Contact Information: Riya Samanta, Biswajeet Sethi, Soumya K. Ghosh, Indian Institute of Technology Kharagpur, India, [riya.samanta@iitkgp.ac.in](mailto:riya.samanta@iitkgp.ac.in), [biswajeet.sethi@iitkgp.ac.in](mailto:biswajeet.sethi@iitkgp.ac.in), [skg@cse.iitkgp.ac.in](mailto:skg@cse.iitkgp.ac.in).

---

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.

Manuscript submitted to ACM

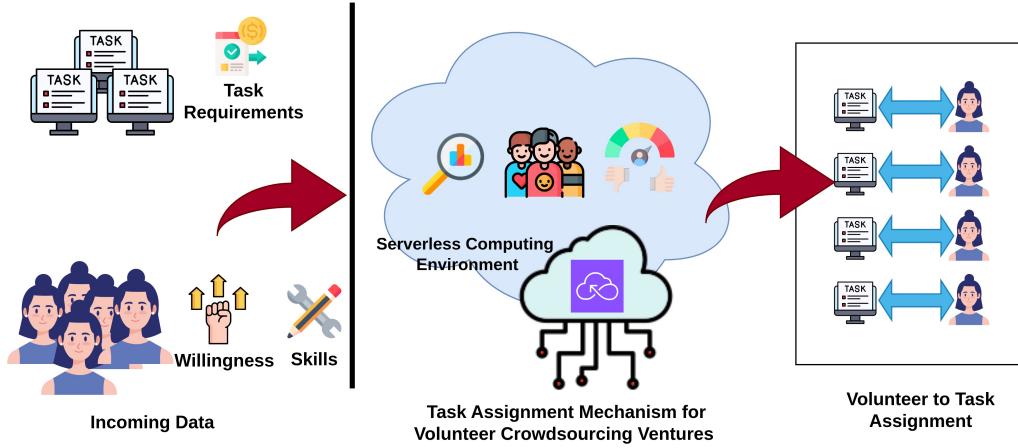
managing servers, it reduces infrastructure costs, making it more accessible for nonprofits and social enterprises with limited resources to develop and deploy applications aimed at solving societal challenges [26]. Serverless architecture enables rapid prototyping and deployment, facilitating the creation of innovative solutions for humanitarian aid, disaster response, healthcare, education, and environmental conservation. Its event-driven nature also allows for efficient processing of real-time data, enabling organizations to respond swiftly to emerging crises and deliver critical services to communities in need. Overall, serverless computing empowers organizations to concentrate on their missions without being burdened by infrastructure constraint. This ultimately has a positive impact on society.

In this work, we focus on volunteer crowdsourcing (VCS) [25] as a pivotal aspect of improving social impact. Volunteer crowdsourcing harnesses citizens' voluntary interaction to tackle challenges by leveraging individuals' knowledge and skills. The integration of smart mobile devices with Web 2.0 technology has significantly enhanced the effectiveness of *skill-based volunteer crowdsourcing* in domains such as healthcare, emergencies, sustainable development, situational awareness, and social awareness. In Asia, several Non-governmental organizations (NGOs) including Goonj, ActionAid, and Smile [5, 12, 28] have successfully utilized volunteer crowdsourcing to tap into the expertise and skills of volunteers. Globally, platforms like VolunteerMatch and Idealist [14, 31] facilitate the alignment of volunteers' interests and skills with the needs of various NGOs. However, matching tasks with the most suitable volunteers remains a significant challenge. Complex tasks often necessitate collaboration among volunteers with diverse skill sets [29]. In addition to skills, the willingness of volunteers to engage in community activities is crucial [24]. Despite their competence, a volunteer's reluctance can hinder success, impacting both task outcomes and the overall utility of the VCS platform. This issue is particularly critical for volunteer services operating within limited budgets and requiring prompt, efficient action. Volunteers may also face costs for transportation, accommodation, sustenance, healthcare, and equipment [25], highlighting the importance of considering volunteer remuneration. These VCS platforms regularly experience a dynamic influx of tasks and volunteers. Determining the required computing resources for volunteer to task matching process is challenging due to unpredictable platform demands in terms of volunteer and tasks requests [19]. Therefore, these platforms must strategically allocate tasks to volunteers in a scalable manner. Small businesses, start-ups, and NGOs are actively seeking cost-effective and quickly deployable solutions to enhance their operational efficiency.

In this study, we propose a serverless-assisted framework for task assignment in the VCS paradigm. This framework leverages the benefits of serverless computing to address the challenges of managing dynamic task assignments for a crowdsourcing system. The **Figure. 1** shows the overview of the proposed framework.

### 1.1 Existing Limitations and Research Gap

The academic research community has largely overlooked the potential of integrating serverless architecture with crowdsourcing applications, despite its significant potential to advance social good (see Related Work, Section 5). Existing literature on skill-based task assignments often neglects critical factors such as skill diversity and workers' willingness, which are crucial for effective task assignment policies in volunteer crowdsourcing (VCS) ventures. These factors become particularly critical in online (dynamic) assignment modes, reflecting real-world scenarios where participants exhibit dynamic arrival and departure patterns following probability distributions. In such online modes, achieving dynamic resource scalability with minimal human intervention is essential. However, current literature lacks discussions on the scalability of crowdsourcing platforms from the deployment standpoint of task assignment mechanisms. Integrating serverless architecture with crowdsourcing applications offers promising solutions to these challenges, enabling platforms to dynamically scale and efficiently handle fluctuating workloads of task assignment decisions in real-world applications.



**Fig. 1.** Overview of proposed Serverless-assisted Framework for task assignment in Volunteer Crowdsourcing Paradigm

## 1.2 Contributions:

In the pursuit of addressing the above mentioned limitations and challenges in the domain of skill-oriented task assignment paradigm for VCS ventures, the main contributions of this paper are:

- (1) We propose the *Skill and Willingness-Aware Volunteer Matching (SWAM)* algorithm, which allocates volunteers to tasks based on their skills, willingness, and tasks' skill requirements.
- (2) We develop a *serverless* framework to deploy the SWAM algorithm , enhancing end-to-end performance by reducing latency and average task waiting time.
- (3) We demonstrate the effectiveness of using a serverless ecosystem for executing SWAM, focusing on latency and task waiting time. We compare SWAM with state-of-the-art skill-oriented task assignment algorithms, evaluating task completion ratio and overall platform utility using a real dataset from Meetup [2].

The remainder of this paper is structured as follows: Section 2 presents preliminaries required for designing the proposed framework. In Section 3, gives the details of the methodology including the working of proposed SWAM algorithm and outline of the serverless deployment framework that enables our SWAM algorithm to be deployed. Section 4 includes an evaluation of the performance of the proposed model compared to the baseline algorithms. In Section 5, we review the relevant literature and prior work in the field. Finally, in Section 6, we provide the conclusion and future work.

## 2 Preliminaries

We proceed to present necessary definitions, preliminaries and important data structures needed in this study.

**DEFINITION 1. (*Volunteer*)** The set of interested volunteers is represented by  $V = \{v_1, v_2, v_3, \dots, v_n\}$ , where each volunteer  $v_j$  is represented by a tuple  $\langle q_v, p_v, \delta_v, e_v \rangle$ . The tuple includes the volunteer's skill set ( $q_v$ ), desired remuneration ( $p_v$ ), arrival time ( $\delta_v$ ), and expected time-to-live ( $e_v$ ).

**DEFINITION 2. (*Task*)** The set of tasks is represented by  $T = \{t_1, t_2, t_3, \dots, t_m\}$ , where each task  $t_i$  is represented by a tuple  $\langle q_t, b_t, \delta_t, e_t \rangle$ . The tuple includes the required skill set ( $q_t$ ), pre-defined budget ( $b_t$ ), arrival time ( $\delta_t$ ), and expected time-to-live ( $e_t$ ). The task is either withdrawn from the system or re-posted after  $e_t$  expires.

The proposed framework operates under the assumption that a task's completion depends on the coverage of skills required for that task. Drawing from the research by [9, 24, 25, 29], it is assumed that both the skills required for a given task and the skills possessed by a volunteer are consistently drawn from a predefined and constant set of skills, denoted as  $Q$ . This implies that for all volunteers  $v \in V$  and tasks  $t \in T$ , the sets of skills  $q_v$  and  $q_t$  are subsets of  $Q$ . Additionally, following the concept of one-to-many mapping as discussed in [9, 24, 25, 29], it is possible for a task to have multiple volunteers at a given time frame, but the reverse scenario is not true.

**DEFINITION 3. (*Willingness*)** It is a crucial aspect in determining the participation of a volunteer in a task. For a volunteer, the willingness to participate in a task denoted as  $W(v, t)$ , is defined as the probability of a volunteer's intention to participate towards the completion of a task. The willingness is formulated as:

$$W(v, t) = \frac{1}{1 + \exp(-\omega(v, t))} \quad (1)$$

where  $\omega(v, t)$  is the summation of the volunteer's efficiency ( $eff_v$ ) and bias ( $\beta_v$ ) towards the task, given by:  $\omega(v, t) = eff_v + \beta_v$

The willingness is measured by the combination of two factors: the volunteer's efficiency ( $eff_v$ ) and bias ( $\beta_v$ ). Assuming that the variables  $\beta_v$  and  $eff_v$  range from 0 to 1,  $\omega(v, t)$  falls within the range of 0 to 2, ensuring that  $W(v, t)$  always ranges between 0.5 and 1. The efficiency ( $eff_v$ ) represents the volunteer's proficiency in the required skills for the task, while the bias ( $\beta_v$ ) expresses the volunteer's preference for the task. Efficiency can be derived from the volunteer's job completion history or performance ratings, and bias can be inferred from the types of tasks the volunteer has previously accepted or completed. For this study, we assume that these data are available to the algorithm from the utilized dataset. The willingness of a volunteer is task-specific and varies based on factors influencing the volunteer's decision-making, such as payment, duration, and prior experience. A highly willing volunteer is more likely to dedicate time and effort to tasks, resulting in improved outcomes, which is crucial for enhancing social community services [24].

**DEFINITION 4. (*Utility Score*)** It provides a numerical representation of the suitability of a task for a volunteer, with higher scores indicating a better fit. Given a volunteer  $v$  and a task  $t$ , the utility score of  $v$ , denoted by  $U_{v,t}$  is calculated by the formula as follows:

$$U_{v,t} = b_t \times W(v, t) \quad (2)$$

It is obvious that, from a volunteer standpoint, a task with a higher fund limit can be expected to cover the incurred expenditure to a certain threshold or to pay  $v$ 's share of remuneration without fail. Thus,  $b_t$  is positively correlated with  $U_{v,t}$ . Additionally, the willingness factor has a positive impact on the  $U_{v,t}$  [24].

**DEFINITION 5. (*Batch-based Assignment*)** Given a set of fixed rotation intervals,  $Time = [X_0, X_1, X_2, \dots, X_k]$ , where  $X_k = k \times X_1$ , a batch-based assignment involves an assignment map  $\mu$  represented as a bipartite graph  $(V, T, E)$ , where  $V = \{v_1, v_2, v_3, \dots, v_n\}$  denotes volunteer nodes and  $T = \{t_1, t_2, t_3, \dots, t_m\}$  denotes task nodes. The edges in the bipartite graph  $E$  are defined as pairs  $(v, t) \in V \times T$ .

The batch-based strategy combines elements of both *offline* and *online* methods, where tasks and volunteers accumulate over a specified time interval before assignment decisions are made. This periodic process aims to minimize assignment costs while maintaining responsiveness in real-time scenarios [27].

**DEFINITION 6. (*Skill-Task Mapper*)** is defined as a bipartite graph denoted by  $GST = (Q, T, E)$ , where an edge  $(x, y) \in E$  exists if and only if a skill  $x \in Q$  is required for a task  $y \in T$ .

The primary intent of the graph is to serve as an index for the skills required for each task and to monitor the extent to which skill requirements are met. The Skill-Task Mapper is implemented as a two-dimensional matrix that undergoes dynamic updates subsequent to each allocation. Only the final row encompasses the budgetary value of every task in order. The process of constructing the matrix involves amalgamating all active tasks to form the column, while each tuple is represented by every skill from the set  $Q$ . Consequently, the dimensions of the  $G_{ST}$  matrix will be  $(|Q| + 1) \times |T|$ .

**DEFINITION 7.** (**Volunteer-Skill Mapper**) is a bipartite graph represented by  $G_{VS} = (Q', T, E)$ , where an edge  $(x, y)$  exists if and only if a skill  $x \in Q'$  is required for any task  $y \in T$ .  $G_{VS}$  is constructed in specific for a volunteer under observation. Thus,  $Q'$  is equivalent to  $q_v$ , which is the skill set available with the volunteer  $v$ . In the worst case,  $Q' \equiv Q$ .

This data structure is predominantly utilised for identifying tasks that are most compatible with the skill set possessed by a volunteer. The implementation of  $G_{VS}$  employs a two-dimensional matrix to map a volunteer's available skills to the required skill set of available tasks. The  $G_{VS}$  will have dimensions of  $|q_v| \times |T|$ . The production of  $G_{VS}$  can be derived in a direct manner from  $G_{ST}$  as a sub-graph.

**DEFINITION 8.** (**Volunteer Utility Vector**) is a vector that stores volunteer utilities for each task. It is denoted by  $UtilVec$ . The utility score is calculated using Equation-2.

Thus the corresponding  $UtilVec$  of  $v_3$  is given by:

$$\begin{aligned} UtilVec &= [U_{v,t_1}, U_{v,t_2}, \dots, U_{v,t_n}] \\ &= [(b_{t_1} \times W(v, t_1)), (b_{t_2} \times W(v, t_2)) \dots (b_{t_n} \times W(v, t_n))] \end{aligned} \quad (3)$$

$UtilVec$  is used in the SWAM's allocation decision. and its length is equal to  $|T|$ .

### 3 Methodology

Our work employed the Server Allocated Task (SAT) model [9] to effectively match volunteers to tasks. We used AWS Lambda [3] for serverless computation and AWS S3 [1] for storing incoming, intermediate, and result data. Given the continuous arrival of task and volunteer data, we implemented a *batch-based split and allocate strategy*. At the start of each time interval ( $X_i$ ), based on data received in the previous batch cycle ( $X_{i-1}$ ), Lambda functions split the incoming data into chunks. We used Amazon EventBridge [4] to trigger these functions periodically.

The splitting process could be regulated by predetermined rules or occur stochastically. In our approach, volunteers were partitioned into  $\kappa$  subgroups, as the number of volunteers typically exceeds the number of tasks on most crowdsourcing platforms, including the Meetup dataset used in this study. The entire task data is processed to construct the bipartite graph  $G_{ST}$  (see Definition 7). This graph is then replicated and distributed to each of the  $\kappa$  Lambda functions, denoted as  $L_{SWAM}$ , along with one of the  $\kappa$  partitioned volunteer datasets. The  $\kappa$  allocation maps are generated by concurrently executing  $\kappa$  instances of the SWAM algorithm as  $\kappa$  Lambda functions that is  $L_{SWAM}$ . Finally, to consolidate these multiple allocation maps into a single final result, we execute a Lambda function, denoted as  $L_{Consol}$ .

#### 3.1 Skill and Willingness-Aware Volunteer Matching Algorithm

This section describes the working of the SWAM algorithm (see Algorithm 1) in detail with example settings depicted in Table 1 and Table 2. The SWAM algorithm takes as its input the pre-processed  $G_{ST}$  and a partitioned volunteer data denoted as  $V'$ , where  $V' \in V$ . The resulting output comprises a penultimate allocation map denoted as  $\mu'$  and an updated  $G'_{ST}$ . The map  $\mu$  essentially functions as a dictionary that maps tasks to the volunteers assigned to them.

**Table 1.** Details of the tasks

Tasks	Skill requirements	Budget
$t_1$	Teaching, Nutritionist, Nursing	\$650
$t_2$	Child care, Teaching, Cooking	\$500

**Table 2.** Details of the volunteers

Volunteers	Skills	Remuneration
$v_1$	Child-care, Guitarist, Nursing	\$25
$v_2$	Cooking, Nursing, Nutritionist	\$20
$v_3$	Teaching, Cooking, Nursing	\$30

The initial stage involves generating a localised replica of  $G_{ST}$  to preserve the integrity of the original  $G_{ST}$  throughout the algorithmic process, as it is required for subsequent stages. The volunteer set is traversed, the one with the lowest remuneration is picked, and his or her  $G_{VS}$  is constructed. The appropriate volunteer from Table 2 is  $v_2$  (seeking a minimum pay of \$20). Next, the  $UtilVec$  of  $c$  is created by the equation-3. We assume that all volunteers have a  $W(c, t)$  of 0.5. The  $UtilVec$  of  $v_2$  for tasks  $t_1$  and  $t_2$  is  $[(650 * 0.5), (500 * 0.5)] = [325, 250]$ . The column-wise sum of  $G_{VS}$  matrix is performed. Thus, the resultant  $col\_sum$  for  $G_{VS}$  of  $v_2$  is  $[2, 1]$ . To obtain the vector  $t_{pref}$ , each  $i^{th}$  element of the  $col\_sum$  vector is multiplied by each  $i^{th}$  element of the  $UtilVec$  vector. As a result,  $t_{pref}$  for  $v_2$  is  $[(2 \times 325), (1 \times 250)] = [650, 250]$ . If  $col\_sum$  of  $v_2$  had been  $[2, 3]$ ,  $t_{pref}$  of  $v_2$  would have been  $[650, 750]$ . The while loop begins. Initially, the argument of the element with the highest value is chosen from  $t_{pref}$ . The argument, in this case, is  $t_1$ . The current budget of  $t_1$  allows for volunteer  $v_2$ . As a result,  $v_2$  is assigned to  $t_1$ , and  $b_{t_1}$  is updated. A new entry for the pair  $(v_2, t_1)$  is also added to the  $\mu'$ . Then, the control comes out of the while loop. If  $b_{t_1}$  was insufficient for recruiting  $v_2$ ,  $v_2$  is removed from  $t_{pref}$  and the next immediate argument (a.k.a task) is chosen from  $t_{pref}$ . This continues until  $t_{pref}$  is empty or completely traversed. At the end,  $G'_{ST}$  is updated. Finally, SWAM concludes by returning  $G'_{ST}$  and  $\mu'$ .

**Algorithm 1** SWAM (Skill and Willingness-Aware Volunteer Matching)

---

**Input:** Volunteer  $V'$ , Skill-Task Mapper  $G_{ST}$   
**Output:** Allocation Map  $\mu'$ ,  $G_{ST}'$

- 1: Start
- 2:  $G'_{ST} \leftarrow G_{ST}$
- 3: **for** all  $c \in V'$  **do**
- 4:     Select volunteer  $c$  from  $V'$  with  $\min(p)$
- 5:     Generate  $G_{VS}$  of  $c$
- 6:     Generate the  $UtilVec$  vector (Equation-3)
- 7:      $col\_sum \leftarrow$  Column-wise sum of  $G_{VS}$
- 8:      $t_{pref} \leftarrow$  Null
- 9:     **for**  $i = 0$  to  $|T|$  **do**
- 10:        $t_{pref}[i] \leftarrow col\_sum[i] \times UtilVec[i]$
- 11:     **end for**
- 12:     **while**  $t_{pref}$  is not empty **do**
- 13:        $t_{reco} \leftarrow \text{argmax}(t_{pref})$
- 14:       **if**  $b_{t_{reco}} \geq p_c$  **then**
- 15:            $t_a \leftarrow t_{reco}$
- 16:           Add allocation  $(c, t_a)$  to  $\mu'$
- 17:            $b_{ta} = b_{ta} - p_c$
- 18:           **break**
- 19:       **else**
- 20:           Remove  $t_{reco}$  from list  $t_{pref}$
- 21:       **end if**
- 22:     **end while**
- 23:     Update  $G'_{ST}$
- 24: **end for**
- 25: **return**  $G'_{ST}, \mu'$
- 26: End

---

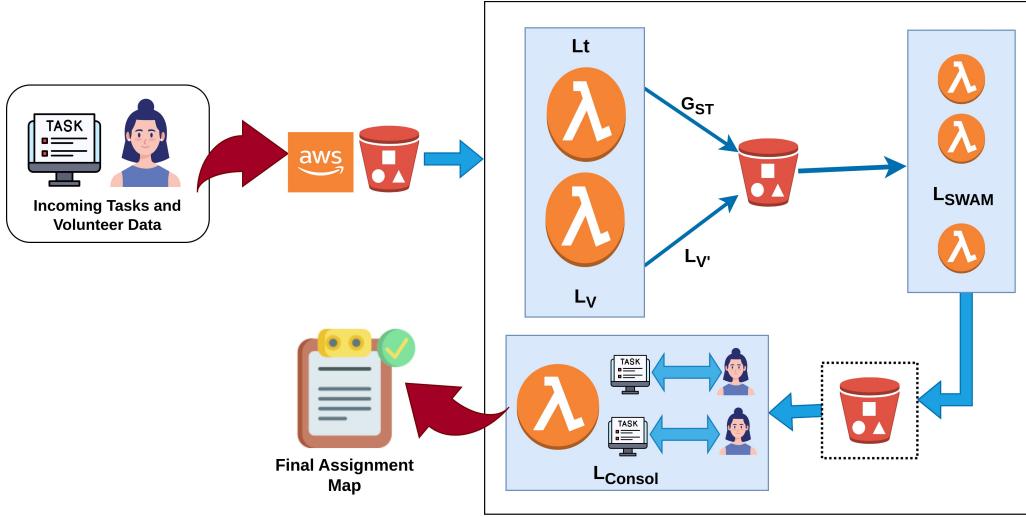


Fig. 2. Serverless Deployment Framework for Task Assignment Mechanism

### 3.2 Serverless Backbone

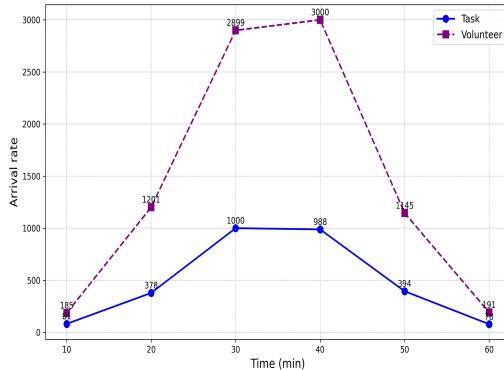
Our framework integrates crowdsourcing with serverless computing platforms, particularly leveraging the *Function-as-a-Service* (FaaS) model known for its automatic scalability and resource provisioning capabilities. The modularity of FaaS architecture contrasts with traditional monolithic applications, making it suitable for managing dynamic data arrival patterns in real-time. This capability is crucial for efficiently handling sudden data influxes without manual intervention, a key advantage of serverless computing over conventional cloud platforms where scalability configuration is developer-dependent. Our serverless deployment workflow involves several key steps: (i) Task and volunteer data are initially stored as separate objects in AWS S3 buckets. (ii) Lambda functions ( $L_t$  for tasks and  $L_v$  for volunteers) process these data to generate the bipartite graph  $G_{ST}$  and segmented volunteer data chunks  $V_{split}$  respectively. The number of segments for volunteer data can be adjusted based on dataset size and complexity. (iii) Each segment of volunteer data is stored in AWS S3, with each bucket mapped to a dedicated Lambda function ( $L_{SWAM}$ ). Concurrently,  $G_{ST}$  is stored in a separate S3 bucket. (iv) The arrival of segmented volunteer data in S3 triggers respective  $L_{SWAM}$  functions, which fetch  $G_{ST}$  to produce assignment maps  $\mu'$  and update skill-task mappings  $G'_{ST}$ . (v) These processed outputs are passed directly to  $L_{Console}$  without intermediate S3 storage, ensuring efficient data flow. (vi) Finally,  $L_{Console}$  consolidates data, removes redundancy, and generates the final assignment map  $\mu$ .

The orchestration of this serverless workflow is managed by *AWS Step Functions*, providing visual workflows for coordinating distributed application tasks.

## 4 Performance Evaluation

### 4.1 Dataset

We use the benchmark event-based social network dataset **Meetup** [2] to simulate our problem, which is a common practice in evaluation of skill-oriented crowdsourcing platform [27, 32]. In this dataset, the events are utilised as tasks and the users are designated as crowd workers. The labels are perceived as skills. We pruned out the skills of the tasks which cannot be covered and that of the crowd workers which are not demanded much by the tasks. There are 1234 tasks ( $|T|$ ) and the total number of skills ( $S$ ) is 554. Each task requires 5 to 10 skills, giving an average of 3.6 skills per



**Fig. 3.** Arrival pattern of tasks and volunteers

task. The mean task budget following Gaussian distribution is \$428, with a standard deviation of \$255. There are 3275 ( $|V|$ ) crowd workers, each having 1 to 5 skills, giving an average of 1.7 skills per crowd worker. The mean remuneration of each crowd worker following Gaussian distribution is \$40, with a standard deviation of \$50. The  $eff_v$  and  $\beta_v$  are generated using a uniform distribution between 0 and 1.

**Figure 3** illustrates the arrival patterns for both tasks and volunteers, using data generated from a Normal Distribution [21] to demonstrate the auto-scaling capability of the proposed framework compared to baseline methods. The x-axis represents time in minutes within an hour interval, while the y-axis shows the arrival rate. The mean is set to 30 minutes, and the standard deviation is 10 minutes. Initially, there is a positive slope for both tasks and volunteers, indicating increasing arrival rates. This upward trend facilitates the evaluation of serverless resource up-scaling. Conversely, a downward slope starting at the 40-minute mark allows for the evaluation of resource down-scaling. The experiment uses a batch size of 10 minutes.

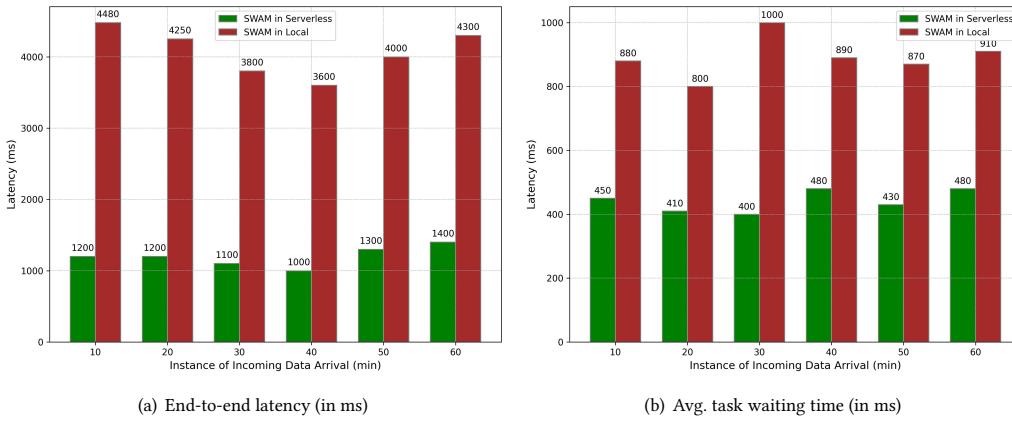
#### 4.2 Experimental Results and Analysis

The evaluation was conducted in two phases. In the first phase, we compared the performance of the SWAM algorithm locally and in a serverless environment, capturing the **end-to-end latency** and **waiting time** in microseconds. In this context, waiting time is defined as the total time a task has to wait before it gets assigned with all the volunteers it needed [27]. In the second phase, we compared the proposed SWAM algorithm with two baseline methods: **Online Greedy (OG)** [29] and **initial Volunteer Task Mapping (i-VTM)** [25]. The comparison was based on the **task completion ratio** and **overall utility score**. The task completion ratio for a specific batch instance is the ratio of successfully completed tasks (those meeting skill requirements) to the total available tasks. A task is considered ‘complete’ only if all associated skills are addressed. The utility score is calculated using Equation 2.

In the first phase of our experimentation, we evaluated the SWAM algorithm’s performance in a scalable serverless architecture with stateless modular parallel executing functions, comparing it against a local environment. Table 3 details the characteristics of both deployment environments. We focused on end-to-end latency as the evaluative parameter. **Figure 4(a)** illustrates the behavior of the SWAM algorithm in both environments. In the local environment, SWAM exhibited a latency ranging from 3665 ms to 4433 ms. In contrast, leveraging serverless functionalities reduced latency to a range of 1022 ms to 1322 ms. **Figure 4(b)** presents the average waiting times observed for SWAM. In the local environment, average waiting times varied between 786 ms and 988 ms. With the serverless ecosystem, maximum and minimum waiting times were reduced to 487 ms and 401 ms, respectively.

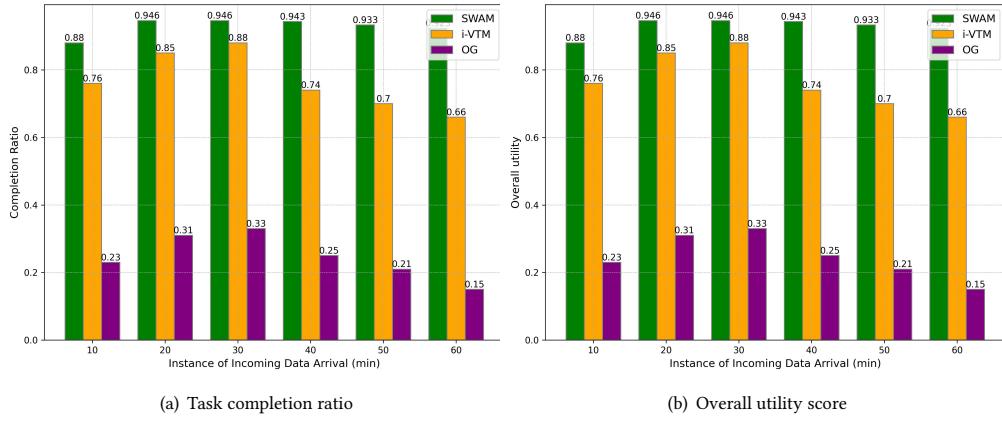
**Table 3.** Experimental Settings

<b>Local Configurations</b>	Processor - Intel i3 dual-core CPU Frequency - 2GHz RAM - 4GB OS - Windows 10 Programming Language - Python 3.9
<b>Serverless Configurations</b>	Platform - Amazon Web Service Computational Unit - AWS Lambda Storage unit - AWS S3 Iam Role - S3FullAccess; CloudwatchFullaccess Lambda memory - 128 MB CPU - Auto provisioned by AWS Programming Language - Python 3.9


**Fig. 4.** Illustration of total end-to-end latency and average task waiting time of SWAM in serverless and local settings.

During the second phase of our experimentation, we conducted a comparative analysis among the SWAM algorithm, OG algorithm, and i-VTM algorithm to assess their effectiveness in achieving task completion ratios and overall utility scores. **Figure 5(a)** illustrates the comparison of completion ratios. The SWAM algorithm achieved an average completion rate of 0.92, which is significantly higher compared to OG and i-VTM. Specifically, SWAM showed a fourfold increase over OG and a one-and-a-half-fold increase over i-VTM. As depicted in the figure, after the initial 40-minute period, there was a decrease of approximately 60% in the number of available volunteers, while the number of tasks experienced a 50% decline. Consequently, fewer volunteers were available to match with tasks based on skill requirements. Additionally, tasks from previous batches accumulated in the pool of incomplete tasks, either due to inadequate funding or unmet skill requirements among participating volunteers.

The previous analysis highlights the trend of the net utility score, depicted in **Figure 5(b)**. The average utility score for SWAM is 355, while i-VTM and OG achieve scores of 272 and 223, respectively. The inclusion of willingness as a factor further enhances SWAM's performance compared to i-VTM, underscoring its direct impact on utility scores. Additionally, the serverless computing framework significantly enhances the SWAM algorithm's performance. Leveraging Function-as-a-Service (FaaS), serverless computing enables parallel execution of data through modular, stateless functions. This modular execution accelerates data processing, and the platform's automatic scalability handles



**Fig. 5.** Illustration of task completion ratio and overall utility score of SWAM, i-VTM and OG in the local setting.

dynamic and unpredictable volunteer and task data influxes without developer intervention. The responsibility for configuring scalability resides entirely with the service provider, alleviating the developer's burden.

## 5 Related Work

### 5.1 Skill-based Task Assignment in Crowdsourcing

Notwithstanding the advancements made, the present body of literature predominantly focuses on the allocation of micro-tasks that do not necessitate substantial cognitive exertion or specialized skills. This leaves a gap in the research regarding the allocation of tasks that require specific skills and cognitive effort. To fill this research gap, several studies have been initiated to explore the impact of skill orientation on task allocation in crowdsourcing. Nonetheless, within this body of literature, there is a scarcity of research that has explicitly concentrated on the allocation of tasks in volunteering services.

The article referenced as [11] addresses the challenge of connecting the skills of workers to the appropriate tasks in crowdsourcing markets. It proposes a mechanism that uses incentives to encourage workers with appropriate expertise to self-select tasks and the objective function includes the quality of work, the worker's expertise, and the cost of completing the task. Cheng et al. [9] propose an approach for task allocation in spatial crowdsourcing, where tasks demand multiple skills to be completed with two specific objectives. The first objective function minimizes the total time required to complete all tasks, including the time for task execution and worker travel time and the second objective function maximizes the total number of tasks completed within a specific time frame, while also considering the skills required for each task and the skills possessed by each worker. Liu et al. [17] further advanced the problem of complex task allocation by proposing an approach that considers the tasks-workers-skills tripartite graph. The approach utilizes two objective functions, one which minimizes the total time required to complete all tasks and another which maximizes the total number of tasks completed within a given time frame. All the above studies are done in offline settings. The allocation of multi-skill tasks in online settings was studied in [29] and [24], with the latter adding a willingness component to the approach. Ni et al. [20] defined dependency-aware spatial crowdsourcing, focusing on maximizing the number of successfully assigned tasks with constraints such as worker skills and deadlines. Liang et al. [16] put forward a cost-oriented greedy strategy for reducing platform expenses through the pairing of appropriate groups of

workers. The selection of volunteers based on skills and proximity was discussed in [25]. Riya et al. [23] addresses a situation of choosing participants in crowdsourcing for disaster management.

## 5.2 Serverless Computing

A considerable amount of research has been done in the field of serverless computing. Papers such as [16] and [13] provides an overview of serverless computing, including its features, opportunities, and challenges. The scalability of serverless platforms and their potential impact on the industry have been discussed in [8] and [15]. Additional research efforts, such as those presented in [10] and [33], investigate different aspects of serverless computing, including cost-effectiveness, performance, and security. Moreover, [22] propose a serverless architecture for highly parallel file-processing tasks and discuss a middleware implementation that facilitates the execution of custom runtime environments within serverless computing. Authors in [6] performed an analysis on various services of different commercially available serverless platforms which claim effective parallelism of workloads. [7] brings the concept of function orchestration and discusses different sequential composition workflows whereas [18] talks about an extensible Trigger-based Orchestration architecture for serverless workflows. The field of serverless computing is constantly evolving, and new research is being conducted to further enhance its capabilities and address existing limitations.

## 6 Conclusion

Matching tasks with the most suitable volunteers remains a significant challenge in volunteer crowdsourcing ventures. Complex tasks often necessitate collaboration among volunteers with diverse skill sets. In addition to skills, the willingness of volunteers to engage in community activities is crucial. These volunteer crowdsourcing (VCS) platforms regularly experience a dynamic influx of tasks and volunteers. Determining the required computing resources for volunteer to task matching process is challenging due to unpredictable platform demands in terms of volunteer and tasks requests. Therefore, these platforms must strategically allocate tasks to volunteers in a scalable manner. The utilisation of serverless technology presents a feasible methodology.

In this paper, we proposed a serverless-assisted framework for task assignment in the VCS paradigm. The proposed Skill and Willingness-Aware Volunteer Matching (SWAM) algorithm allocates volunteers to tasks based on their skills, payments, willingness, and tasks' skill requirements. We developed a serverless framework to deploy the SWAM algorithm, achieving a performance efficiency of 71% in the serverless environment in terms of total end-to-end latency. The investigations were conducted using the real-world MeetUp dataset. We achieved a task completion ratio of 92% and reduced the waiting time for submitted tasks before assignment to volunteers by 56%. Furthermore, the overall utility gain was approximately 30% higher than the baselines.

In future research, we plan to use cooperative game theory to model the collaborative dynamics among volunteers (workers), tasks, and platforms. Integrating cooperative game theory into the skill-oriented task assignment process will promote equitable, productive, and collaborative outcomes. Additionally, we aim to test our framework on a larger dataset and improve the algorithm for assigning tasks to volunteers. This will help in further refining the task matching process and enhancing the efficiency and effectiveness of volunteer crowdsourcing platforms. By continuing to innovate in this space, we hope to further support grassroots movements and foster greater citizen engagement and transparency, ultimately contributing to the global social good.

## References

- [1] 2022. Cloud Object Storage Compute - Amazon S3 - Amazon Web Services. <https://aws.amazon.com/s3/>. Last accessed 11 Dec 2022.

- [2] 2022. Meetup - We are what we do. <https://www.meetup.com/>. Last accessed 21 June 2022.
- [3] 2022. AWS Lambda - Serverless Compute. <https://aws.amazon.com/lambda/>. Last accessed 21 June 2022.
- [4] 2023. AWS. <https://aws.amazon.com>. Last accessed 02 Jan 2023.
- [5] Action-Aid-India. 2023. Action Aid India. <https://www.actionaidindia.org/>. Last accessed 31 January 2023.
- [6] Daniel Barcelona-Pons, Pedro García-López, Álvaro Ruiz, Amanda Gómez-Gómez, Gerard París, and Marc Sánchez-Artigas. 2019. Faas orchestration of parallel workloads. In *Proceedings of the 5th International Workshop on Serverless Computing*. 25–30.
- [7] Urmil Bharti, Deepali Bajaj, Anita Goel, and SC Gupta. 2021. Sequential Workflow in Production Serverless FaaS Orchestration Platform. In *Proceedings of International Conference on Intelligent Computing, Information and Control Systems: ICICCS 2020*. Springer, 681–693.
- [8] Kyle Chard and Ian Foster. 2019. Serverless Science for Simple, Scalable, and Shareable Scholarship. In *2019 15th International Conference on eScience (eScience)*. 432–438. <https://doi.org/10.1109/eScience.2019.00056>
- [9] Peng Cheng, Xiang Lian, Lei Chen, Jinsong Han, and Jizhong Zhao. 2016. Task assignment on multi-skill oriented spatial crowdsourcing. *IEEE Transactions on Knowledge and Data Engineering* 28, 8 (2016), 2201–2215. <https://doi.org/10.1109/TKDE.2016.2550041> arXiv:1510.03149
- [10] Ivan Chiliman et al. 2021. Serverless Computing: A Comprehensive Survey. *IEEE Communications Surveys and Tutorials* 23, 1 (2021), 677–719.
- [11] Gagan Goel, Afshin Nikzad, and Adish Singla. 2014. Allocating tasks to workers with matching constraints: truthful mechanisms for crowdsourcing markets. In *Proceedings of the 23rd International Conference on World Wide Web*. 279–280.
- [12] Goonj. 2023. Goonj. <https://goonj.org/>. Last accessed 31 January 2023.
- [13] Joseph M Hellerstein, Jose Faleiro, Joseph E Gonzalez, Johann Schleier-Smith, Vikram Sreekanti, Alexey Tumanov, and Chenggang Wu. 2018. Serverless computing: One step forward, two steps back. *arXiv preprint arXiv:1812.03651* (2018).
- [14] Idealistic. 2023. Idealist. <https://www.idealistic.org/en>. Last accessed 31 January 2023.
- [15] Hyungro Lee, Kumar Satyam, and Geoffrey Fox. 2018. Evaluation of production serverless computing environments. In *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*. IEEE, 442–450.
- [16] Zhejun Liang, Wenan Tan, Jiu Liu, and Kai Ding. 2022. Multi-skill collaboration-based task assignment in spatial crowdsourcing. In *International Conference on Computer Application and Information Security (ICCAIS 2021)*, Vol. 12260. SPIE, 42–48.
- [17] Jiaxi Liu, Haogang Zhu, and Xiao Chen. 2016. Complicated-skills-based task assignment in spatial crowdsourcing. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 9998 LNCS (2016), 211–223. [https://doi.org/10.1007/978-3-319-47121-1\\_18](https://doi.org/10.1007/978-3-319-47121-1_18)
- [18] Pedro Garcia López, Aitor Arjona, Josep Sampé, Aleksander Slominski, and Lionel Villard. 2020. Triggerflow: trigger-based orchestration of serverless workflows. In *Proceedings of the 14th ACM international conference on distributed and event-based systems*. 3–14.
- [19] Xiaoye Miao, Huanhuan Peng, Yunjun Gao, Zongfu Zhang, and Jianwei Yin. 2022. On Dynamically Pricing Crowdsourcing Tasks. *ACM Transactions on Knowledge Discovery from Data (TKDD)* (2022).
- [20] Wangze Ni, Peng Cheng, Lei Chen, and Xuemin Lin. 2020. Task allocation in dependency-aware spatial crowdsourcing. *Proceedings - International Conference on Data Engineering 2020-April* (2020), 985–996. <https://doi.org/10.1109/ICDE48307.2020.00090>
- [21] Jagdish K Patel and Campbell B Read. 1996. *Handbook of the normal distribution*. Vol. 150. CRC Press.
- [22] Alfonso Pérez, Germán Molto, Miguel Caballer, and Amanda Calatrava. 2019. A programming model and middleware for high throughput serverless computing applications. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*. 106–113.
- [23] Riya Samanta and Soumya K Ghosh. 2022. FogiRecruiter : A fog-enabled selection mechanism of crowdsourcing for disaster management. *Concurrency and Computation: Practice and Experience* (2022), 1–9. <https://doi.org/10.1002/cpe.7207>
- [24] Riya Samanta, Soumya K Ghosh, and Sajal K Das. 2021. SWILL-TAC: Skill-oriented Dynamic Task Allocation with Willingness for Complex Job in Crowdsourcing. In *2021 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 1–6.
- [25] Riya Samanta, Vaibhav Saxena, Soumya Ghosh, and Sajal K. Das. 2022. Volunteer Selection in Collaborative Crowdsourcing with Adaptive Common Working Time Slots. In *2022 IEEE Global Communications Conference: Selected Areas in Communications: Social Networks (Globecom2022 SAC SN)*. Rio de Janeiro, Brazil.
- [26] Biswajeet Sethi, Sourav Kanti Addya, and Soumya K Ghosh. 2023. LCS: Alleviating Total Cold Start Latency in Serverless Applications with LRU Warm Container Approach. In *24th International Conference on Distributed Computing and Networking*. 197–206.
- [27] Biswajeet Sethi, Riya Samanta, Soumya K Ghosh, and Sajal K Das. 2023. Scalable Skill-oriented Task Allocation in Crowdsourcing within a Serverless Ecosystem. In *24th International Conference on Distributed Computing and Networking*. 135–139.
- [28] Smile-Foundation. 2023. Smile Foundation. <https://www.smilefoundationindia.org/>. Last accessed 31 January 2023.
- [29] Tianshu Song, Ke Xu, Jiangneng Li, Yiming Li, and Yongxin Tong. 2020. Multi-skill aware task assignment in real-time spatial crowdsourcing. *GeoInformatica* 24, 1 (2020), 153–173. <https://doi.org/10.1007/s10707-019-00351-4>
- [30] Gianni Tumedei, Francesco Boschi, Catia Prandi, Luís Gomes, Rui Calheno, Rui Abreu, Shuhao Ma, Valentina Nisi, Nuno Nunes, and Augusto Esteves. 2021. Promoting a safe return to university campuses during the covid-19 pandemic: Crowdsensing room occupancy. In *Proceedings of the Conference on Information Technology for Social Good*. 145–150.
- [31] Volunteer-Match. 2023. VolunteerMatch. <https://www.volunteermatch.org/>. Last accessed 31 January 2023.
- [32] Yuan Xie, Yongheng Wang, Kenli Li, Xu Zhou, Zhao Liu, and Keqin Li. 2023. Satisfaction-aware Task Assignment in Spatial Crowdsourcing. *Information Sciences* 622 (2023), 512–535. <https://doi.org/10.1016/j.ins.2022.11.081>
- [33] Yanyan Zhang et al. 2019. Serverless computing: A survey. *Journal of Systems and Software* 157 (2019), 39–58.