JKAU: Comp. IT. Sci., **Vol**. **8** No. 1, pp: 45 – 57 (1440 A.H. / 2019 A.D.)
Doi: 10.4197/Comp. 8-1.5

Task-Scheduling Based on Multi-Objective Particle Swarm Optimization in Spatial Crowdsourcing

Afra A. Alabbadi and **Maysoon F. Abulkhair**

Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, Saudi Arabia

aalabaade@stu.kau.edu.sa

Abstract. As a result of the rapid growth of internet and smartphone technology, a novel platform that attracts individuals and groups known as crowdsourcing emerged. Crowdsourcing is an outsourcing platform that facilitates the accomplishment of costly tasks that consume long periods of time when traditional methods are used. Spatial crowdsourcing (SC) is based on location; it introduces a new framework for the physical world that enables a crowd to complete spatial-temporal tasks. The primary issue in SC is the assignment and scheduling of a set of available tasks to a set of proper workers based on different factors, such as the location of the task, the distance between task location and hired worker location, temporal conditions, and incentive rewards. In the real-world, SC applications need to optimize multi-objectives simultaneously to exploit the utility of SC, and these objectives can be in conflict. However, there are few studies that address this multi-objective optimization problem within a SC environment. Thus, the authors propose a multi-objective task scheduling optimization problem in SC that aims to maximize the number of completed tasks, minimize total travel cost, and ensure worker workload balance. To solve this problem, we developed a method that adapts the multi-objective particle swarm optimization (MOPSO) algorithm based on a proposed novel fitness function. The experiments were conducted with both synthetic and real datasets; the experimental results show that this approach provides acceptable initial results. As future work, we plan to improve the effectiveness of our proposed algorithm by integrating a simple ranking strategy based on task entropy and expected travel costs to enhance MOPSO performance.

Keywords: Task-scheduling, Spatial crowdsourcing, MOO, MOPSO.

1. Introduction

It is the authors' responsibility to ensure that the manuscript is novel, original and never published in the past in any form any media. The corresponding author should provide a declarative statement that the paper is not an extended or modified version of any conference or journal, this manuscript is 100% original and unpublished. The manuscript has not been published in parts (figures/text/tables) in any conference proceedings or journal in any media or language or format. Recently, crowdsourcing has become a trending outsourcing platform that facilitates the hiring of workers to accomplish tasks that can consume long periods of time when traditional methods are used. Spatial crowdsourcing (SC) is an extension of crowdsourcing in which tasks are treated as spatial tasks that can only be performed in specific physical locations. In other words, the main difference between traditional crowdsourcing and SC is that the worker must physically move to the task's location to complete the work. In SC, there are

three primary participants: Crowdsourcing requesters, crowdsourcing platforms, and crowd workers (workers). The requester submits a task to the crowdsourcing platform, which manages the task and connects the worker and the requester, and the worker performs the task (Fig. 1). The most common issues in SC, as shown in Fig. 2, are task assignment [1],[2],[3], security and privacy [4],[5], incentive mechanism [6], and quality control[7],[8]; however, the core challenge is task assignment [9],[10],[11].

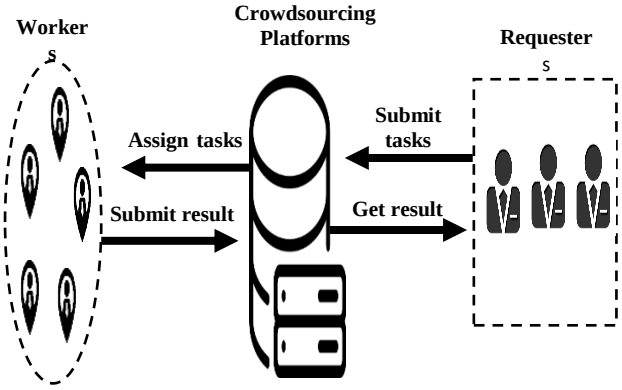

Fig. 1. **Fig. 1. The Spatial Crowdsourcing Components.**

Task assignment involves selecting a suitable worker to complete a specific spatial task or set of tasks correctly, under some predefined constraints.

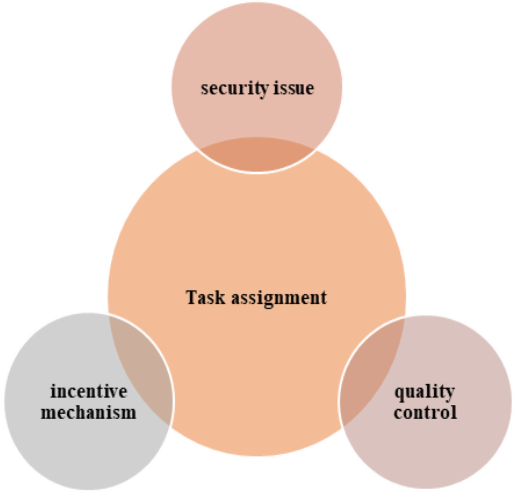

Fig. 2. The Common Issues in Spatial Crowdsourcing.

L. Kazemi and C. Shahabi categorized task assignment into the two models presented in [1]: Worker selected tasks (WSTs) and server assigned tasks (SATs). In the WST model, the worker selects and completes a task from the available orders based on their own selection criteria. In the SAT model, the task is assigned by the server to a suitable worker in accordance with certain constraints.

Researchers formulated task assignment issues as task matching [1], [2], [11], [12], or task scheduling [9],[13],[14]. Due to the continual movement between task locations in SC, each worker must develop an ideal way for accomplishing all of their assigned tasks, accounting for the predefined constraints. Thus, the task assignment problem requires concentration on the task-scheduling problem. In the real-world, SC applications need to concurrently optimize potentially conflicting multi-objectives to exploit the utility of SC. Therefore, multi-objective optimization must be considered in SC. Our study investigates the following fundamental questions:

- Q1: How to schedule the tasks optimally in SC?

- Q2: How can we optimize three conflicting objectives that include maximizing the number of completed task, minimizing the total travel cost and ensuring the workload balancing between workers?

In this study, a multi-objectives task scheduling optimization (MOTSO) problem in SC was formulated based on the preceding three conflicting objectives. In previous studies, the meta-heuristic algorithms was used to resolve task-scheduling problems, such as the particle swarm optimization (PSO) [15],[16],[17], the Genetic algorithm (GA) [18],[19],[20], and Ant Colony Optimization (ACO) [21],[22],[23]. PSO is a meta-heuristic based optimization algorithm discovered by R. Eberhart [9]. and it is inspired by animal social

behavior. This study proposed a multi-objective particle swarm optimization (MOPSO) algorithm based on the authors' special function to solve the proposed problem. This paper aims to maximize the number of completed task, minimize total travel costs, and ensure balance between worker workloads. PSO was used to develop this work because of its many advantages, which are summarized as follows: it is a simple concept, is easy to implement, completes fast computations, produces a quick convergence to an optimal solution, and has durable control parameters. Further, the PSO can be used in various applications to obtain the optimal solution. The main contributions of this paper are summarized as follows:

- The MOTSO problem that aims to maximize the number of completed tasks, minimize the total travel cost, and ensure the workload is balanced between workers was formulated.
- To the best of the authors' knowledge, there are no studies that deal with the MOTSO problem based on the three outlined objectives in relation to SC. This is the first study to solve a MOTSO problem that involves three conflicting objectives in a SC environment.
- To solve this MOTSO problem, the authors adapted a MOPSO based on the novel fitness function that will be explained in the methods section of this paper.
- The experiments were conducted to evaluate the performance of the proposed solution using both real and synthetic datasets.

This paper is organized as follows: Section II discusses the related works regarding task assignment in SC. This study's proposed methodology is explained in section III. Section IV presents the proposed model's performance evaluation. and section V concludes this article.

2. Related Work

Recently, the SC field, including platforms such as Gigwalk and TaskRabbit, has attracted significant attention from both research and industry communities. In this section, we will review the works of the many researchers who have studied task assignment problems in SC. We will discuss how some have formulated the task assignment problem as a task matching problem, as in [1], [3], [11], [12], [24] and [25] or a task scheduling problem, as in [9], [13] and [14] to achieve different objectives. Despite the attention paid to this field, studies highlighting the multi-objective optimization (MOO) problem in SC have been limited. In this section, we will present the related works based on the three following aspects.

A. Task Matching Technique

A task matching technique has been used in several studies to achieve different objectives. For instance, Kazemi *et al.* [1] presented a framework in the Server Assigned Task (SAT) model in SC with the assumption that the worker is reliable. The main objective in [1] was maximizing task assignment (MTA), which was formulated as a matching problem and then solved by reducing the maximum flow problem. The reliability of the worker was subsequently integrated into the framework developed in [1] by Kazemi *et al.* [12]. In [12], the authors tried to maximize the number of assigned tasks, the achievement of which required many reliable workers. the study in [24] also extended the model of [1] by considering the expert scores of the workers. The framework developed in [24] aimed to maximize the score assignment task (MSA) by proposing three heuristic algorithms based on maximum weighted bipartite matching (MWBM). Real-time task assignment is considered in SC, as mentioned in [3],[11] [25]. While the real-time framework

introduced in [25] to solve a Hyper-local Spatial Crowdsourcing problem. This study aims to maximize assigned task under budget constraints. In fact, a Hyper-local Spatial Crowdsourcing is not the focus of our study. Where the Hyper-local approach discussed in [25] does not require workers to travel in order to complete spatial tasks that may not fit with most of SC applications like home delivery services and Uber applications. A trichromatic online matching problem involving three sides or objects (i.e., a task, worker, and workplace) was introduced and reduced to 3-D matching in [11]. However, due to the nature of continual movement between locations in SC, each worker must have an ideal way to accomplish all of the tasks that are assigned to him under predefined constraints. Therefore, solving the task assignment problem requires the consideration of the task scheduling problem. Thus, in this study, we concentrate on the task scheduling problem based on MOO.

B. Task Scheduling Technique

Some researchers consider the task scheduling problem in SC based on several objectives, for instance, Deng *et al.* [14] were the first to formulate the task scheduling problem in SC within a WSTs model. They solved the maximum task scheduling problem by maximizing the number of completed tasks while considering the task deadline and travel costs between the task and the worker. Dealing with the SATs, [13] combined task scheduling and matching to solve the maximum task scheduling problem by improving the number of completed tasks and travel costs under the expiration time. Sun *et al.* [9] proposed load-balancing-based spatial task scheduling (LBSTS) to minimize the waiting time for users by avoiding workers' overload. All of these previous researches optimized only one primary objective without considering the MOO problem. A real-time application would need to optimize many objectives to better utilize SC advantages.

C. Multi-Objective Optimization (MOO) in SC

The researchers of [26] proposed and solved a MOO problem in SC requiring the optimization of two objectives (i.e., the travel cost and task reliability) by reducing it to a minimum-cost MWBM problem. Only a few studies have used meta-heuristic algorithms to resolve MOO problems in SC. For instance, Tran *et al.* [34] improved their approach proposed in [33] to solve a MOO problem related to hyper-local SC using a genetic algorithm. Their two-objective optimization involved maximizing task coverage and minimizing the highest workload across all workers to avoid workers' overload under budget variants. Wang *et al.*[27] investigated heterogeneous spatial crowdsourcing task allocation (HSC-TA) based on a two-objective optimization problem involving task coverage and incentive budget. They proposed two algorithms to obtain the Pareto optimal solution. None of the existing studies have dealt with a multi-objective task scheduling problem in SC with three conflicting objectives, such as maximizing the number of completed tasks, minimizing the total travel costs, and maintaining workload balancing among workers.

Unlike all of the previous studies discussed, our study explores the multi-objective task scheduling optimization (MOTSO) problem in SC and produces a novel solution that aims to maximize the number of completed tasks, minimize the total travel costs, and maintain workload balancing among workers by avoiding workers' overload. We adopted the multi-objective particle swarm optimization (MOPSO) algorithm to improve task scheduling in SC.

3. Methodology

This section presents a detailed description of the research methodology applied to achieve the primary objectives of the research. In the following sections, we will present the SC scenario and then explain the steps of the proposed MOPSO.

A. Spatial Crowdsourcing Scenario

Our Spatial Crowdsourcing Scenario is shown in Fig. 3 can be summarized as the following:

- The requesters send (SC-Query) which contains the task with its constraints $t_j = < l_{t_j}, d_{t_j} >$ to SC-server. Where l_{t_j} denoted to the location of the task while d_{t_j} is the time duration of a task.
- The workers send their locations l_{wi} to the server.
- In this step, the server will activate our MOPSO by providing it with the two sets: tasks $T = \{t_1, t_2, t_3, \ldots . t_n\}$ and workers W = $\{w_1, w_2, w_3, \ldots . w_m\}$, since the server, has a global picture of each task in the tasks' set, and each worker in the workers' set.

Then our algorithm will assign each task to an optimal worker, depending on our objectives. Where the assumptions of our work are presented in the following steps:

- Workers are volunteers, a volunteer worker who is ready to accomplish spatial tasks without compensation.
- Each task should be assigned to only one worker.
- We assume all the workers are reliable since this research is considering that all workers presented their work with the same quality.

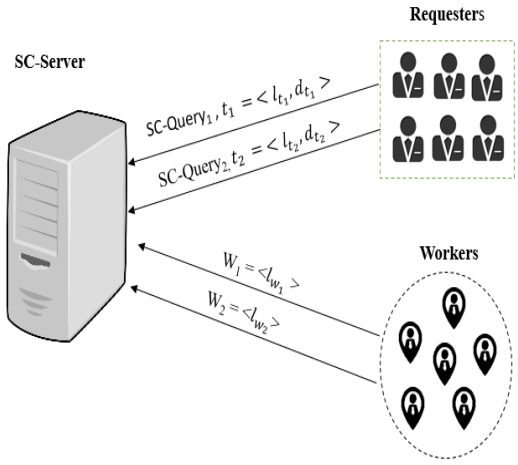

Fig. 3. The Spatial Crowdsourcing Scenario of our study.

B. Multi-Objective Particle Swarm Optimization Task-Scheduling

The Particle Swarm Optimization (PSO) concept consists of the swarm, which is the population of particles. The particles are individuals that are moving in a solution space; every particle has a position and velocity. While $\boldsymbol{gBest}$ is the best position achieved by all particles, the best particle's position achieved by the same particle is known as $\boldsymbol{pBest_z}$. The PSO algorithm is widely used for different optimization problems, including MOO problems. Thus, the PSO approach used to solve a MOO problem is known as MOPSO [28]. In other words, MOPSO integrates the advantages of MOO and PSO to improve the finding of a solution. MOPSO is appropriate for solving our problem because we aim to achieve optimal task scheduling based on multiple objectives, including maximizing the number of completed tasks, minimizing the total travel costs, and ensuring the workload balancing among workers. MOPSO consists of significant steps that will be included in each iteration, as illustrated in Fig. 4.

Specifically, the MOPSO algorithm starts by initializing all particles in the swarm with velocity, position, and $\boldsymbol{pBest_z}$ values.

Each velocity and position of a particle is initialized randomly and structured as $i \times j$, a matrix where i is the number of tasks and j is the number of workers. The fitness value is computed. At each iteration, $\boldsymbol{pBest_z}$ and ***gBest*** must be updated to guide all of the particles to the optimal solution. By comparing the new position, $\boldsymbol{pos_{z_{ij}}}$, of the particle with the $\boldsymbol{pBest_z}$ value, we can store the best value for each particle. By comparing the $\boldsymbol{pBest_z}$ of each particle with the global best particle, ***gBest***, we can store the best position of all particles as the global best, ***gBest***.

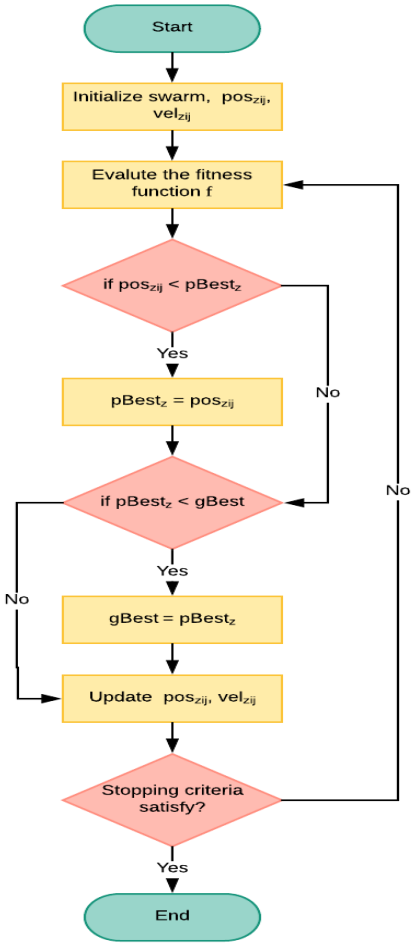

Fig. 4. The Flowchart of PSO.

One of the reasons for the popularity and spread of PSO is its simplicity since it depends on only two equations to update the particle's position, as shown in Equations (1) and (2). Equation (1) is as follows:

$$\boldsymbol{pos_{z_{ij}} = \left(pos_{z_{ij}} + vel_{z_{ij}} \right)} \quad (1)$$

where $\boldsymbol{pos_{z_{ij}}}$ indicates the position of particle z and $\boldsymbol{vel_{z_{ij}}}$ represents the particle's velocity. Equation (2) is as follows:

$$\begin{aligned} \boldsymbol{vel_{z_{ij}}} = & \left(\boldsymbol{W} * \boldsymbol{vel_{z_{ij}}} \right) \\ & + \left(\boldsymbol{C1} * \boldsymbol{r1} * \left(\boldsymbol{pBest_z - pos_{z_{ij}}} \right) \right) \\ & + \left(\boldsymbol{C2} * \boldsymbol{r2} \right. \\ & \left. * \left(\boldsymbol{gBest - pos_{z_{ij}}} \right) \right) \end{aligned} \quad (2)$$

where $pos_{z_{ij}}$ indicates the position of particle z and $\boldsymbol{vel_{z_{ij}}}$ is the velocity of particle z. $\boldsymbol{pBest_z}$ shows the personal best position of particle z, which indicates the best value of the particle so far. gBest is the best position for particle z among all particles, which means the best result achieved by the swarm so far [29]. The flying of a particle in the search space is guided by both $\boldsymbol{pBest_z}$ and gBest over all iterations [28]. Whereas w is the inertia weight, C1, C2 are the acceleration coefficients and r1, r2 are the random numbers between 0 and 1 [30]. The pseudo code of our task-scheduling-based MOPSO is given in Algorithm 1.

The fitness function is a characterization of objective optimization that evaluates a particle's position. The fitness value of the swarm was calculated based on our objectives; we formulated our objectives into three functions to evaluate the swarm at each iteration:

i. *The number of completed tasks, |V|:* The first objective is maximizing the number of completed tasks. We established V⊆T, which is a set of all completed tasks, V = { vij … vmn}, where vij ∈ V denotes the task, tj, which is completed by workers, wi. W and T are sets of all workers and tasks, respectively. Thus, (wi,

wm) ⊆ W and (tj, tn) ⊆ T. From this, |V| can denote the number of completed tasks.

ii. *Total Travel Cost (TTC):* The travel cost is a crucial element in SC because the worker must physically travel from his location to the location of the task to execute the spatial task. In our MOPSO, for each particle in each iteration of gBest, we computed the TTC using Equation (3). We measured the travel cost between the worker, wi, and the completed task, vij, using the Euclidean distance:

$$Total\ Travel\ Cost\ (TTC) = \sum_{i=1}^{m}(cost(w_i, v_{i1}) + \sum_{j=1}^{n-1} cost(v_{ij}, v_{i(j+1)})),\ v_{ij} \in V\ \ and\ (w_i, w_m) \subseteq W\ and\ (t_j, t_n) \subseteq T \quad (3)$$

where *cost* (v_{ij} ,$v_{i(j+1)}$) is the cost of the same worker, w_i, traveling from task v_j to the next task, $v_{i(j+1)}$, calculated using the Euclidean distance. *Cost* (w_i ,$v_{i(j+1)}$), on the other hand, is the cost of worker w_i traveling from their starting point to the location, v_{ij}.

1) Workload Balancing (WLB): WLB among workers is essential in SC because it helps to prevent overload among workers. Establishing a balanced workload among workers is helpful for accelerating the completion of requested tasks. The workload of workers in this research can be computed using Equation (4):

$$WL(w_i) = cost\ (w_i, v_{i1}) + \sum_{j=1}^{n-1} cost(v_{ij}, v_{i(j+1)}) + \sum_{j=1}^{n} (d_{ij} * s_{w_i})\ , v_i \in V\ \ and\ (t_j, t_n) \subseteq T \quad (4)$$

where cost *(vij ,vi(j+1))* is the cost of worker wi traveling from task vij to the next task, *vi(j+1)*, and cost *(wi ,vi(j+1))* is the cost of worker wi traveling from their location to the location of the first task *vij*. Further, *dij* is the duration spent by the same worker, *wi*, to complete each task, *vij*, while *Swi* is the speed of the worker. As suggested by [31], we can use standard deviation as a metric to quantify the WLB among workers. Thus the third our objective optimization is a minimizing standard deviation of workload balancing between all workers. The formula for WLB can be calculated as Equation (5):

$$WLB = \sqrt{\frac{1}{|W|}\sum_{i=1}^{m}(WL(w_i)) - AWL)^2} \quad (\mathbf{5})$$

where |W| is the total number of workers, in W; WL *(w_i)* is the workload of the worker, w_i ; and AWL is the average workload of ***gBest***. In this study, the task scheduling is based on the optimization of three conflicting objectives. To unify their direction, we utilized the normalization approach based on Equation (6) as follows:

$$Normalization = \frac{f_i(X) - f_i^{min}(X)}{f_i^{max}(X) - f_i^{min}(X)} \quad (6)$$

where $f_i(X)$ is the value of the objective function, $f_i^{min}(X)$ is the minimum value of the objective, and $f_i^{max}(X)$ is the maximum value of the objective.

From all of the above, we can calculate the value of the fitness function for all of our objectives as follows:

$$f = NWLB + NTTC + (1 - NV) \quad (7)$$

where NWLB is the normalized WLB, NTTC is the normalized TTC, and NV is the normalized |V|. (1-NV) is a reversal of the NV, i.e., it is a reversal of the objective of maximizing the number of completed tasks into another form. This minimizes the number of uncompleted tasks to unify the direction of our objectives for optimization. The pseudo code for the fitness function is presented in Algorithm 2.

4. Performance Evaluation

To evaluate the proposed algorithm experimentally, we relied on certain

specifications, such as Intel (R) Core (TM) i9-8950 CPU @2.90 GHz with 32 GB RAM, to run the algorithm. The proposed algorithm was implemented using a Java environment. We used both synthetic and real datasets (Gowalla) to test our proposed algorithm. Using SCAWG, we generated a synthetic dataset (SYN) with uniform (UNI) distributions in a 200 X 200 grid. We adopted the Gowalla check-in dataset for simulation [32] which has been used in several related works, such as [3], [9], [13] and [25]. The Gowalla does not include task durations; thus, in our evaluation, the 2,400 tasks and workers were randomly selected from Gowalla venues using SCAWG with random task durations, as was done in [3]. Our algorithm is affected by many control parameters. Table 1 displays the settings of the parameters used to implement our MOPSO.

Moreover, we evaluated our algorithm by defining two different cases with the synthetic dataset: the first case (case 1) had 600 workers and 600 independent spatial tasks, while the second case (case 2) had a doubled amount with 1,200 workers and 1,200 independent spatial tasks. In this work, we assumed the same speed of 80 for all workers, while the duration of the task was randomly generated with a maximum value of 90. In addition, the baseline algorithm (BLA) in our study is based on the approach in [9] We compared our MOPSO with the BLA to evaluate the effectiveness of our algorithm.

Table 1. The setting of our test.

	Parameters	**Value**
MOPSO Configuration	Swarm size	50
	Iteration number	80
	W	[0,1]
	C1, C2	2.00, 2.00
	r1, r2	[0,1]
SC Configuration	S_{wi}	80
	Max d_{tj}	90

We ran each experiment ten times and calculated the average value for each of our factors. In the following section, we present and analyze our results in terms of the number of completed tasks, TTC, and WLB.

Algorithm 1: the pseudo code of MOPSO algorithm.

```
// MOPSO
Input: worker set W, Task set T
Output: gBest
Initialization a set of swarm particles (population) P, iteration = 0
Foreach particle p_z ∈ P do
  Foreach task t_j ∈ T do
    Foreach worker w_i ∈ W do
      pos_zij =Random (poz_min, poz_max_ij )
    // initialize particle position and PBest randomly
      vel_zij = Random (vel_min, vel_max)
    // initialize particle velocity randomly
    End Foreach
  End Foreach
End Foreach
While the termination criterion not satisfied do
Foreach particle p_z ∈ P do
  Foreach task t_j ∈ T do
    Foreach worker w_i ∈ W do
      f= Evaluate (p)   \\ equation fitness function
      pBest_z = Update_pBest(f)
      gBest = Update_gBest(f)

      compute vel_zij // using equation (2)
      compute pos_zij // using equation (1)
    End Foreach
  End Foreach
End Foreach
End While
Return gBest
The End
```

Algorithm2: Pseudo code for the fitness function

```
// Evaluate particle
Input: particle p
Output: double f as fitness
Foreach task t_j ∈ T do
  Foreach worker w_j ∈ W do
    Schedule = Assign tasks j to worker i for lowest pos_zij
  End Foreach
End Foreach
NWLB =Normalized(WLB(Schedule))
NTTC = Normalized (TTC(Schedule))
NV = Normalized(Completed Tasks (Schedule))
f = NWLB + NTTC + (1- NV)
Return f
```

1. The number of completed tasks, |V|: The results of the first objective (i.e., maximizing the number of completed tasks) are presented in Table 2. Fig. 5, 6, and 7

illustrate the respective percentages of completed tasks in the task set, T, calculated using our algorithm: about 65% for case 1, 64% for case 2, and 63% for Gowalla. The percentages of uncompleted tasks are lower, around 35%, 36%, and 37% for case 1, case 2, and Gowalla, respectively. By contrast, the percentages of completed tasks calculated using the BLA are about 79% for case 1, 82% for case 2, and 79% for Gowalla, which are higher than those produced by our MOPSO. It is clear from Fig. 8 that we need to develop our proposed algorithm further to improve task scheduling with the goal of maximizing the number of completed tasks. In fact, considering location entropy will help to maximize task completion, as mentioned in [3]. Based on this, we will integrate task entropy in our algorithm to increase the number of completed tasks. Task entropy takes into consideration the number of visitors to a task's location.

Table 2. The result of the Number of Completed Task.

The number of the completed task using our MOPSO	
dataset	**|V|**
Case1	391
Case2	771
Gowalla	1523

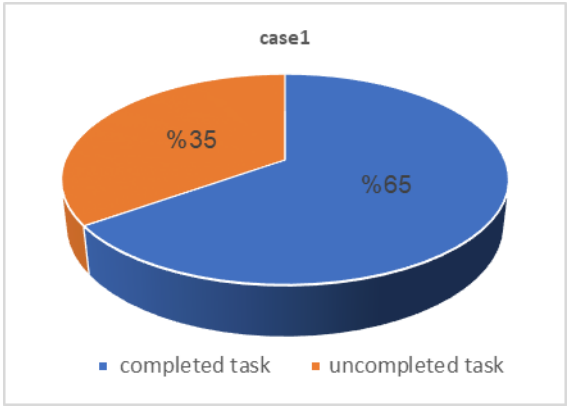

Fig. 5. The number of completed task in case1.

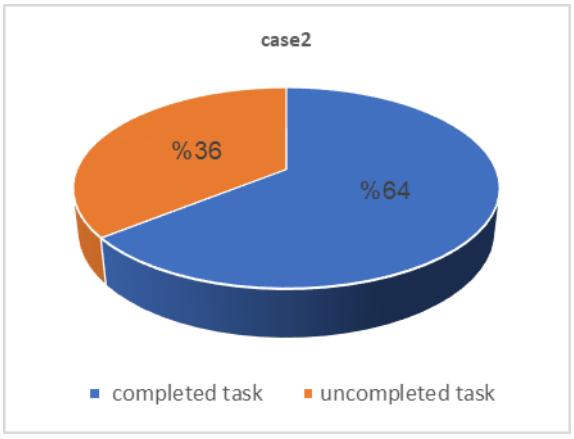

Fig. 6. The number of completed task in case2.

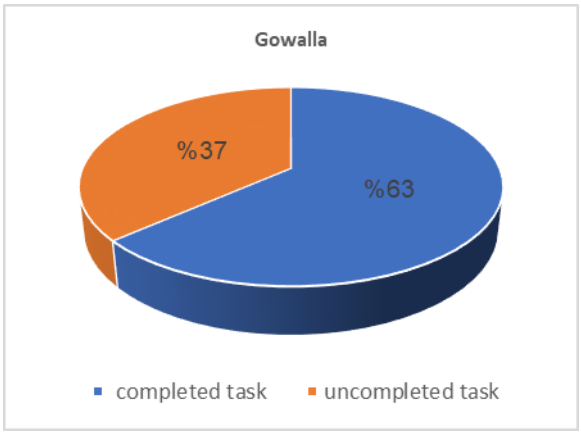

Fig. 7. The number of completed task in case2.

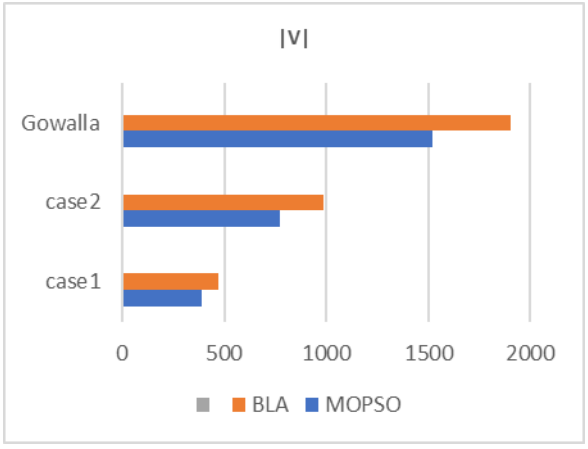

Fig. 8. The number of completed by BLA and MOPSO.

2. TTC: In this work, we aimed to minimize the TTC. The results of the TTC obtained from our algorithm are presented in Table 3. Fig. 9 shows a small gap between the performance of MOPSO and that of the BLA in reducing the TTC. Compared with the BLA, MOPSO reduced the TTC by around 5% for case 1, 6% for case 2, and 7% for Gowalla. This means that the results did not show a significant difference. Thus, we must improve our MOPSO to reduce the TTC.

Table 3. The workload balancing with the MOPSO.

The workload balancing with MOPSO		
dataset	***WLB***	***AWL***
Case1	43.95842	37.595999
Case2	44.52464	37.099551
Gowalla	44.1647	36.425756

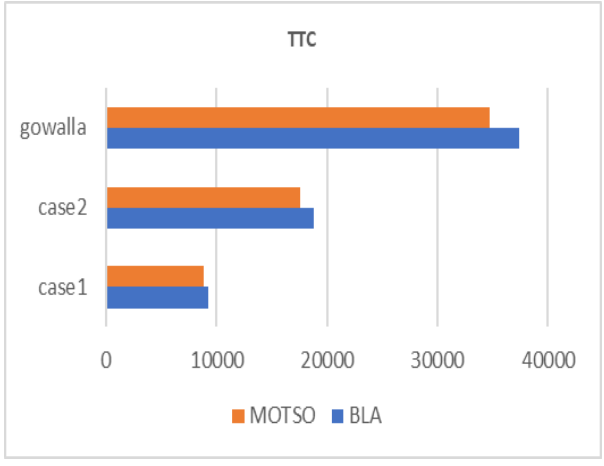

Fig. 9. The total travel cost by MOPSO and BLA.

3. WLB: Table IV presents the results of maintaining the WLB (i.e., the standard deviation of the worker's workload) and the average workload among workers computed using MOPSO. To evaluate the effectiveness of our proposed algorithm, we compared the results obtained using our algorithm with those obtained using the BLA. There are no significant differences between the results of two algorithms regarding the WLB, as shown in Fig. 10 & 11, however, shows that there are apparent differences between the averages of the workload calculated using MOPSO and the BLA. Compared with the BLA, MOPSO improved the averages of the workload by around 79% for case 1 and 80% for both case 2 and Gowalla.

Based on all previous initial results, we plan to improve our algorithm by adopting a simple ranking strategy based on task entropy and expected travel cost to enhance the performance of MOPSO and to achieve the essential research objectives.

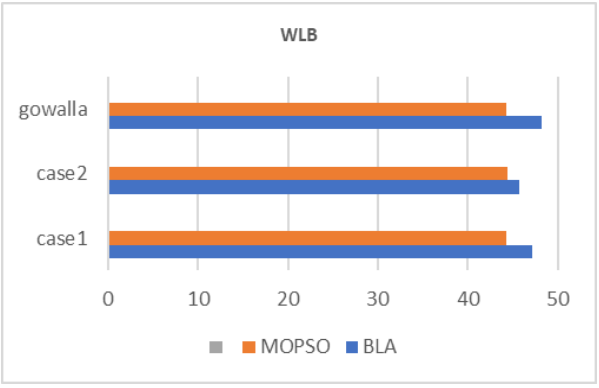

Fig. 10. The workload balancing between workers by MOPSO and BLA.

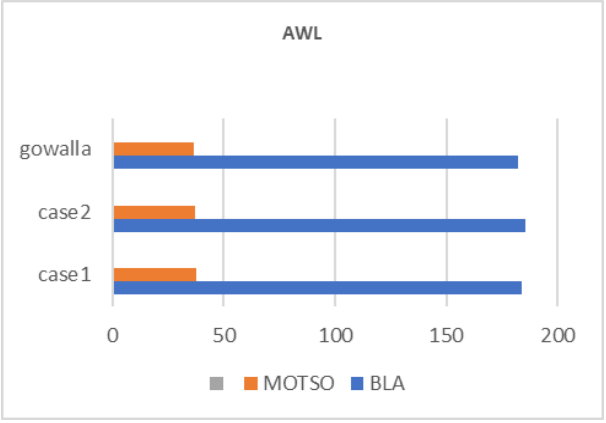

Fig. 11. The Average workload between workers by MOPSO and BLA.

5. Conclusion

The developing crowdsourcing environment introduced the SC framework to complete spatial tasks, but SC optimization contains many challenges, specifically task assignment. To tackle this challenge, this study proposed a MOTSO problem that sought to

maximize the number of completed tasks, minimize total travel costs, and ensure a workload balance between workers. To achieve the optimal solution, the MOPSO algorithm based on a particular fitness function formulated to provide the optimal solution was developed. The proposed MOPSO algorithm was evaluated using real and synthetic datasets and revealed acceptable initial outcomes. To further investigate these findings, future work will enhance the performance of the MOPSO algorithm by integrating a simple ranking strategy based on task entropy and expected travel cost to further validate this study's results.

Acknowledgment

King Abdulaziz City for Science and Technology supported this paper (Grant No. 1-17-00-009-0030).

References

[1] **Kazemi, L.** and **Shahabi, C.,** "GeoCrowd: Enabling Query Answering with Spatial Crowdsourcing," in *Proceedings of the 20th International Conference on Advances in Geographic Information Systems, New York, NY, USA*, 2012, pp. 189–198.

[2] **Cheng, P., Lian, X., Chen, L., Han, J.** and **Zhao, J.,** "Task Assignment on Multi-Skill Oriented Spatial Crowdsourcing," *IEEE Trans. Knowl. Data Eng.*, **28**(8): 2201–2215, Aug. 2016.

[3] **Tran, L., To, H., Fan, L.** and **Shahabi, C.,** "*A Real-Time Framework for Task Assignment in Hyperlocal Spatial Crowdsourcing*," ArXiv170406868 Cs, Apr. 2017.

[4] **Hu, J., Huang, L., Li, L., Qi, M.** and **Yang, W.**, "Protecting Location Privacy in Spatial Crowdsourcing," in W*eb Technologies and Applications*, vol. **9461**, R. Cai, K. Chen, L. Hong, X. Yang, R. Zhang, and L. Zou, Eds. Cham: Springer International Publishing, 2015, pp. 113–124.

[5] **To, H., Ghinita, G.** and **Shahabi, C.,** "A framework for protecting worker location privacy in spatial crowdsourcing," *Proc. VLDB Endow.*, **7**(10): 919–930, Jun. 2014.

[6] **Liu, J.-X., Ji, Y.-D., Lv, W.-F.** and **Xu, K.,** "Budget-Aware Dynamic Incentive Mechanism in Spatial Crowdsourcing," *J. Comput. Sci. Technol.*, **32**(5): 890–904, Sep. 2017.

[7] **Yu, H., Miao, C., Shen, Z. and Leung, C.,** "Quality and Budget Aware Task Allocation for Spatial Crowdsourcing," in *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, Richland, SC*, 2015, pp. 1689–1690.

[8] **Cheng, P. et al.,** "Reliable diversity-based spatial crowdsourcing by moving workers," P*roc. VLDB Endow.*, **8**(10): 1022–1033, Jun. 2015.

[9] **Sun, D., Gao, Y.** and **Yu, D.,** "Efficient and Load Balancing Strategy for Task Scheduling in Spatial Crowdsourcing," in *Web-Age Information Management*, S. Song and Y. Tong, Eds. Springer International Publishing, 2016, pp. 161–173.

[10] **Tong, Y., She, J., Ding, B., Wang, L.** and **Chen, L.,** "Online mobile Micro-Task Allocation in spatial crowdsourcing," in *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, 2016, pp. 49–60.

[11] **Song, T. et al.,** "Trichromatic Online Matching in Real-Time Spatial Crowdsourcing," in *2017 IEEE 33rd International Conference on Data Engineering (ICDE), San Diego, CA, USA*, 2017, pp. 1009–1020.

[12] **Kazemi, L., Shahabi, C.** and **Chen, L.,** "GeoTruCrowd: Trustworthy Query Answering with Spatial Crowdsourcing," in *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, New York, NY, USA*, 2013, pp. 314–323.

[13] **Deng, D., Shahabi, C.** and **Zhu, L.,** "Task Matching and Scheduling for Multiple Workers in Spatial Crowdsourcing," in *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems, New York, NY, USA*, 2015, pp. 21:1–21:10.

[14] **Deng, D., Shahabi, C.** and **Demiryurek, U.,** "Maximizing the Number of Worker's Self-selected Tasks in Spatial Crowdsourcing," in *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, New York, NY, USA*, 2013, pp. 324–333.

[15] J.-T. Tsai, J.-C. Fang, and J.-H. Chou, "Optimized task scheduling and resource allocation on cloud computing environment using improved differential evolution algorithm," Comput. Oper. Res., vol. 40, no. 12, pp. 3045–3055, Dec. 2013.

[16] **Zhang, G.** and **Zuo, X.,** "Deadline Constrained Task Scheduling Based on Standard-PSO in a Hybrid Cloud," in A*dvances in Swarm Intelligence*, vol. **7928**, Y. Tan, Y. Shi, and H. Mo, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 200–209.

[17] **Jana, B., Chakraborty, M.** and Mandal, T., "A Task Scheduling Technique Based on Particle Swarm Optimization Algorithm in Cloud Environment," in *Soft Computing: Theories and Applications*, vol. **742**, K. Ray, T. K. Sharma, S. Rawat, R. K. Saini, and A. Bandyopadhyay, Eds. Singapore: Springer Singapore,

2019, pp. 525–536.

[18] W**ang, T., Liu, Z., Chen,Y., Xu, Y.** and **Dai, X.**, "Load Balancing Task Scheduling Based on Genetic Algorithm in Cloud Computing," in *2014 IEEE 12th International Conference on Dependable, Autonomic and Secure Computing, Dalian, China*, 2014, pp. 146–152.

[19] **Keshanchi, B., Souri, A.** and **Navimipour, N. J.,** "An improved genetic algorithm for task scheduling in the cloud environments using the priority queues: Formal verification, simulation, and statistical testing," *J. Syst. Softw.*, **124**:1–21, Feb. 2017.

[20] **Yin, S., Ke, P.** and **Tao, L.** "An improved genetic algorithm for task scheduling in cloud computing," in *2018 13th IEEE Conference on Industrial Electronics and Applications (ICIEA), Wuhan*, 2018, pp. 526–530.

[21] M. A. Tawfeek, A. El-Sisi, A. E. Keshk, and F. A. Torkey, "Cloud task scheduling based on ant colony optimization," pp. 64–69, Nov. 2013.

[22] **Li, K., Xu, G., Zhao, G., Dong,Y.** and **Wang, D.,** "Cloud Task Scheduling Based on Load Balancing Ant Colony Optimization," in *2011 Sixth Annual Chinagrid Conference*, 2011, pp. 3–9.

[23] **Reddy, G. Narendrababu** and **Kumar, S. Phani**, "Modified Ant Colony Optimization Algorithm for Task Scheduling in Cloud Computing Systems," in *Smart Intelligent Computing and Applications*, vol. 104, S. C. Satapathy, V. Bhateja, and S. Das, Eds. Singapore: Springer Singapore, 2019, pp. 357–365.

[24] **To, H., Shahabi, C.** and **Kazemi, L.,** "A Server-Assigned Spatial Crowdsourcing Framework," *ACM Trans Spat. Algorithms Syst*, **1**(1): 2:1–2:28, Jul. 2015.

[25] **To, H., Fan, L. Tran, L.** and **Shahabi, C.,** "Real-time task assignment in hyperlocal spatial crowdsourcing under budget constraints," in *2016 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, 2016, pp. 1–8.

[26] **ul Hassan, U.** and **Curry, E.,** "Efficient task assignment for spatial crowdsourcing: A combinatorial fractional optimization approach with semi-bandit learning," *Expert Syst. Appl.*, **58**: 36–56, Oct. 2016.

[27] **Wang, L., Yu, Z., Han, Q., Guo, B.** and **Xiong, H.,** "Multi-objective Optimization based Allocation of Heterogeneous Spatial Crowdsourcing Tasks," *IEEE Trans. Mob. Comput.*, pp. 1–1, 2017.

[28] **Zhang, C., Li, Q., Chen, P., Feng, Q.** and **Cui, J.,** "An improved multi-objective particle swarm optimization and its application in raw ore dispatching:," *Adv. Mech. Eng.*, Feb. 2018.

[29] H. Zhang, P. Li, Z. Zhou, and X. Yu, "A PSO-Based Hierarchical Resource Scheduling Strategy on Cloud Computing," in Trustworthy Computing and Services, 2012, pp. 325–332.

[30] Shi, Y. and Eberhart, R., "A modified particle swarm optimizer," in *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360), Anchorage, AK, USA*, 1998, pp. 69–73.

[31] Pearce, O., Gamblin, T., de Supinski, B. R., Schulz, M., and Amato, N. M., "Quantifying the Effectiveness of Load Balance Algorithms," in *Proceedings of the 26th ACM International Conference on Supercomputing, New York, NY, USA*, 2012, pp. 185–194.

[32] "*SNAP*: Network datasets: Gowalla." [Online]. Available: http://snap.stanford.edu/data/loc-gowalla.html. [Accessed: 17-Jan-2017].

جدولة المهام باستخدام خوارزمية استمثال عناصر السرب متعدد الأهداف في التعهيد الجماعي المكاني

عفراء عبد الله العبادي و **ميسون أبو الخير**

كلية الحاسبات وتقنية المعلومات، جامعة الملك عبدالعزيز، جدة، المملكة العربية السعودية

aalabaade@stu.kau.edu.sa

المستخلص. أدى النمو السريع للإنترنت والهواتف الذكية، إلى ظهور منصة جديدة تجذب الأفراد والمجموعات، المعروفة باسم التعهيد الجماعي. والتعهيد الجماعي هو الاستعانة بمصادر خارجية هي منصة الاستعانة بمصادر خارجية لتسهل إنجاز المهام الصعبة التي تستهلك وقتا طويلا بالطرق التقليدية. ظهر التعهيد الجماعي المكاني كإطار جديد في العالم الفعلي لتمكين المشاركين من إكمال المهام المكانية والزمانية. أي أنه في هذا النوع من منصات التعهدالجماعي، العمال يحتاجون للسفر لإنجاز المهام المكانية خلال فترة زمنية محددة. إحدى القضايا المهمة في هذه المنصات هي جدولة مجموعة من المهام المتاحة لمجموعة من العاملين المناسبين لأداء المهام الموكلة إليهم بطريقة مثالية، بناء على عدة عوامل مختلفة، مثل موقع المهمة، والمسافة المقطوعة للوصول إلى المهمة، والمكافآت التحفيزية وغيرها. من ناحية أخرى تحتاج تطبيقات التعهيد الجماعي المكاني إلى تحسين الأهداف المتعددة مع، وقد تكون هذه الأهداف متعارضة. ومع ذلك، هناك نقص في الدراسات التي تعالج مشكلة التحسين متعددة الأهداف داخل بيئة التعهيد الجماعي المكاني. في هذا البحث، يكشف الباحثون مشكلة جدولة المهام متعددة الأهداف التي تهدف إلى زيادة عدد المهام المكتملة، وتقليل تكاليف السفر (أي مسافة السفر)، مع ضمان توازن عبء العمل للعاملين. ولحل هذه المشكلة، تم استخدام خوارزمية استمثال عناصر السرب متعددة الأهداف. وأجريت التجارب على مجموعة من البيانات الاصطناعية والحقيقية من أجل تقييم فعالية المقترح. أثبتت التجارب أن النهج المقترح يعطي نتائج أولية مقبولة. وفي ضوء تلك النتائج نخطط لتحسين نهجنا المقترح من خلال النظر الشعبية الموقع بالإضافة إلى حساب المسافة المتوقعة باستخدام استراتيجية تصنيف بسيطة.

الكلمات المفتاحية: جدولة المهام، التعهيد الجماعي المكاني، MOO، MOPSO.