International Journal of Applied Metaheuristic Computing
Volume 13 • Issue 1

GRASP-Tabu Search Algorithms for the Route Planning Problem in Spatial Crowdsourcing

Mourad Bouatouche, SIMPA, Université des Sciences et de la Technologie d'Oran Mohamed Boudiaf, Oran, Algeria*
Khaled Belkadi, SIMPA, Université des Sciences et de la Technologie d'Oran Mohamed Boudiaf, Oran, Algeria

ABSTRACT

With the speedy progress of mobile devices, a lot of commercial enterprises have exploited crowdsourcing as a useful approach to gather information to develop their services. Thus, spatial crowdsourcing has appeared as a new platform in e-commerce and which implies procedures of requesters and workers. A requester submits spatial tasks request to the workers who choose and achieve them during a limited time. Thereafter, the requester pays only the worker for the well-accomplished task. In spatial crowdsourcing, each worker is required to physically move to the place to accomplish the spatial task, and each task is linked with location and time. The objective of this article is to find an optimal route to the worker through maximizing her rewards with respecting some constraint, using an approach based on GRASP with Tabu. The proposed algorithm is used in the literature for benchmark instances. Computational results indicate that the proposed and the developed algorithm is competitive with other solution approaches.
Keywords GRASP, Metaheuristics, Optimization, Orienteering Problem, Spatial Crowdsourcing, Tabu, Task Planning

INTRODUCTION

Crowdsourcing is a generic term for a variety of approaches that exploit the capacity of large crowds by issuing calls for contributions for specific tasks. Although crowdsourcing approaches can take many different forms, today it is increasingly done via the Web, which allows interaction with a plurality of contributors from around the world. Several crowdsourcing approaches include web platforms, Well-known examples are Wikipedia, Mechanical Turk applications. In recent years, Crowdsourcing research has attracted a lot of attention in variety of fields such as IT, Geographic Information Systems, management and many other areas that have discovered crowdsourcing as a useful approach. Collaboration in a crowdsourcing process is essential to the successful resolution of problems outsourced by the company. In these times crowdsourcing has become a better approach to enhance the company's service (Amrollahi & Ahmadi, 2019).

Since the rapid growth of mobile technology, a new framework called spatial crowdsourcing which replaced traditional Web-based crowdsourcing. The spatial crowdsourcing is used to enable workers to achieve spatial tasks. Each worker is physically needed to move to the place to perform the task close to his actual positions. For instance, the companies are interested in gathering photos or videos of their products for taking statistics for sale, verifying stokes from different areas of a city.

DOI: 10.4018/IJAMC.292502 *Corresponding Author

Spatial crowdsourcing has been commonly used in numerous applications in different fields such as business intelligence (TaskRabbit, Gigwalk, and Fieldagent,) and spatiotemporal data collection (OpenStreetMap). (Fielagent, 2019; Gigwalk, 2019; OpenStreetMap, 2019; TaskRabbit, 2019).

The Spatial Crowd Sourcing Platform specifies workers to accomplish near spatial tasks that enable workers to physically move to a specific location to perform those tasks. With spatial crowdsourcing, the companies emit his request to a spatial crowdsourcing platform, as a result, the spatial crowdsourcing platform crowrdrsouces the request included in the available workers in the close tasks. Once the workers complete their nearby tasks, the outcomes will be returned to the companies. There are two major modes in spatial crowdsourcing: one is the server assign tasks, and the other is the worker select tasks (Kazemi & Shahabi, 2012).

Recently, many of spatial crowdsourcing platforms have allocated tasks to workers on the basis of the nearest worker available. The spatial crowdsourcing platform assigns the closest worker upon obtaining a spatial task.

In practice, the previous approaches have the following weaknesses:

First, it assigns tasks depending on the worker's travel distance to the task. Second, it does not take into account rewards to plan an optimal route. Third, tasks may not be assigned to suitable workers because the workers look for the nearest task through their position. In addition, despite most of the existing techniques are only available for the matching and assignment task problem, these techniques utilize a heuristic method (Tong et al, 2019), and few researchers have dealt with the route planning in spatial crowdsourcing.

To overcome these weaknesses mentioned above, in this paper, the authors study a Route Planning problem in Spatial Crowdsourcing so that the objective is to find an optimal route based on maximizing the rewards of workers, provided that the tasks must be carried out with respect of some constraints (total time and deadline). The Route Planning problem can be compared to the Orienteering Problem (OP) and its variants. Each worker should then solve a variant of the OP (Gunawan et al., 2016). How to plan routes for appropriate workers is one of the most relevant issues in spatial crowdsourcing study.

This paper's principal contribution can be summarized as follows:

- The author define Route Planning problem in spatial crowdsourcing to satisfy the need of the real world to plan a route for worker in order to maximize the reward of the worker.
- The authors propose an approach based on GRASP with tabu search to solve the Route Planning problem in Spatial Crowdsourcing. GRASP-Tabu algorithm's main feature integrates the benefits of its constituent algorithms. This algorithm allows the workers to choose tasks optimally.
- The authors perform extensive experiments through Solomon's and Cordeau et al.'s instances. The experimental results indicate that the proposed algorithm is efficient and effective.

The rest of the paper is arranged according to the following. Section 2 surveys the literature. Section 3 presents the formal definition of the problem of Route Planning .Section 4 describes the details approach based GRASP with Tabu Search. Experimental results are reported in Section 5. The last section includes results and some suggestions about future research.

RELATED WORK

This section presents the different methods to solve the problem related to the crowdsourcing mobile and orienteering problem and its platform.

Spatial Crowdsourcing Platform

Most existing actual spatial crowdsourcing platforms help the worker in selecting destinations, they provide. Several instances of such spatial crowdsourcing markets have emerged commercially

including such as Gigwalk, FieldAgent, and TaskRabbit typical tasks reward users for taking photos of buildings, product placement checks in stores, traffic checks, and price checks. (Fielagent, 2019; Gigwalk, 2019; TaskRabbit, 2019).

These platforms use the current location of a worker to supply him with a set of geographically nearby tasks. The worker is permitted to attract tasks that they can fulfill but he must figure out how to maximize the gains, even while reducing travel costs and adhering to the task deadlines Chen et al. (2014) but no algorithm is used by these platforms to find the optimal route.

The Planning of a Worker's Route

Ludwig et al. (2009) supposed a smartphone application combining the suggestion of events and pedestrian navigation with (live) support for public transport. They introduced A*-method like algorithm. Liao and Hsu (2013) presented an algorithm which is planned to create a detour path and to maximize the worker's reward from the current position. They employed a dynamic programming algorithm.

Chen et al. (2014) investigated the problem of large-scale mobile crowdsourcing. The problem considered a wide pool of crowd-workers to accomplish several urban tasks unique to the area. The aim is to maximize the total gathered rewards from accomplished task nodes while meeting whole time constraints. They developed a method called Greedy-ILS. Chen et al. (2015) extended the mentioned problem Chen et al. (2014) by supposing that each user has a finite list of possible paths with a known probabilistic distribution. They used the Lagrangian relaxation method. Costa and Nascimento (2020) proposed two heuristic approaches, one is based on local optimizations, and the other one is based on incremental solutions to solve the the Online In-Route Task Selection (Online-IRTS) problem.

Task Assignment

Yuen et al. (2015) introduced a new framework for crowdsourcing activities to enable employees to continue working on long-term crowdsourcing projects. The concept uses a worker's past job preference and results to generate a set of tasks available to better suit the worker's selection process. It aims at improving the productivity of tasks. Kazemi and Shahabi (2012) presented maximum task assignment, they concentrated on the assigning jobs to employees in an optimal manner, suggesting the server had global information about the positions of all the employees and jobs.

Zhuan Shi et al. (2016) researched the optimal task assignment problem in crowdsensing systems, which can maximize the percentage of work accomplishment, taking into account the total time of the workers. Li et al. (2015) presented an online task scheduling problem so that the worker can get the maximum reward from tasks along the route when he arrives in time. Deng and Shahabi (2016) researched the spatial crowdsourcing problem in which the workers ndependently choose their tasks, they presented two exact methods based on dynamic programming and branch-and-bound. Y. Wang et al (2020) studied the MQC-TA (Maximum Quality and Minimum Cost Task Assignment) problem, they proposed combined of genetic algorithm and ant colony optimization algorithm. Chen et al. (2020) presented the minimizing maximum delay spatial crowdsourcing (MMD-SC) problem, they suggested a space embedding based online random algorithm and two heuristic algorithms namely the threshold based greedy method and the batch-based method.

Orienteering Problem

The route planning problem for spatial crowdsourcing can be considered to be identical with the variants of orienteering problem (OP). Many algorithms deal with OPTW or (T) OPTW that are discussed in the literature. These algorithms could be also applied to the spatial crowdsourcing problem. The closest variant of OP to this problem is orienteering problem with window time (OPTW), where the worker is planned to accomplish different tasks to gather reward within a time budget and window time (Gunawan et al., 2016). Righini and Salani (2008) created an exact algorithm based on

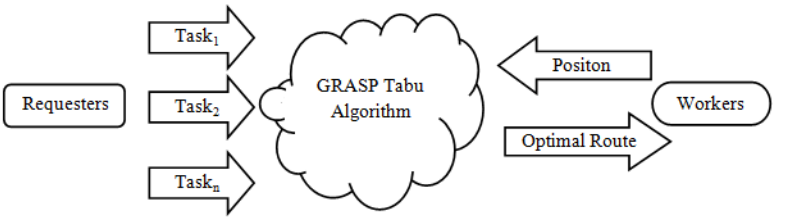
bidirectional dynamic programming to solve OPTW. Vansteenwegen et al. (2009) proposed a very fast ILS algorithm to solve the TOPTW is proposed.

Montemanni and Gambardella (2009) developed the algorithm for the ant colony system (ACS). Numerous other metaheuristics are available also in literature such as a hybridization of a greedy randomized adaptive search procedure (GRASP) and an evolutionary local search (ELS) method (Labadie et al., 2011), Simulated Annealing (SA) (Lin & Yu,2012), A hybrid algorithm based on GRASP and ILS (Souffriau et al.,2013), an LP-based granular variable neighborhood search (GVNS) (Labadie et al., 2012), and an artificial bee colony approach(ABC) (Cura, 2014). Moreover, an iterative framework (I3CH) based on two algorithms, a local search (LS) method, and SA, is suggested by Hu and Lim (Hu & Lim, 2014). Finally, a hybridization of SA and ILS are proposed by Gunawan et al. (Gunawan et al., 2017).

PROBLEM DESCRIPTION AND MATHEMATICAL FORMULATION

Figure 1 illustrates the spatial crowdsourcing system, which includes four components, spatial tasks, workers, the platform and requesters (Liao & Hsu, 2013; Tong et al, 2019).

Figure 1. The Spatial Crowdsourcing platform



Preliminaries

We first present some terminology and then define our problem formally:

Definition 1 (Spatial tasks): with spatiotemporal constraints (e.g., the positions $[x_t, y_t]$ and deadlines of tasks d_t) are submitted to the platform. To complete a task, a worker has to physically move to the position of the task illustrates the considered.

Definition 2 (Workers): send their spatiotemporal information like their positions $[x_w, y_w]$ and deadlines d_w to the platform. If request q_i is accomplished by a worker between time [Oi, Ci], that worker receives reward r during the reception the route, the worker follows it to accomplish requests.

Definition 3 (The platform): the spatial crowdsourcing platforms link tasks with workers. Its core functions include assigning tasks to suitable workers, collecting the results sent by workers, setting rewards for workers.

Definition 4 (the Requester): sends requests with a time window $[T_o, T_c]$ reward r, position $[x_t, y_t]$, and service time s_t to the platform.

With the above terminology, we can begin to define our problem formally below. The aim is to plan the route that will help the worker to accomplish a task. The problem can be considered as a version of the Orienteering Problem with Time Windows witches is used to model a single worker. A set of

n tasks is given, where each of them (i=1… n) relates to a reward r_i, a typical service time (d_i), an opening (O_i) and closing (C_i) time. The route of work is limited to a maximal time T_{max}. The time t_{ij} required traveling from location i to j, and vice versa is known for all locations of tasks. Generally, not all tasks can be fulfilled throughout the route, since the duration of the route is limited to T_{max}. Each task can be fulfilled at most once. The service start time (s_i) of task i is within a time window. The task will expire after the corresponding deadline; the worker will leave the platform after his deadline. A worker must be assigned to it before the next task appears.

The route planning problem in Spatial Crowdsourcing aims at finding a worker's route and he allows maximizing her total reward with respect to some constraint.

The authors will use the system architecture as illustrated in Figure 1. Requesters publish new tasks. Each time there is a new worker, the crowdsourcing platform sends optimal route P for the worker using the GRASP with Tabu algorithm. The worker receives the optimal route on his mobile device, and then he follows him to accomplish his task.

Next, based on (Gunawan et al., 2016; Liao & Hsu, 2013), the Route Planning problem in Spatial Crowdsourcing can be formulated as an integer programming, where we make use of a decision variable. For every route from 1 to N, if task i is followed by task j we set the variable y_{ij} equal to 1 or equal to 0 otherwise. M is a constant. With this notation above we have the following relations:

$$\max \sum_{i=2}^{N-1} \sum_{j=2}^{N} r_i y_{ij} \tag{1}$$

$$\sum_{j=2}^{N} y_{1j} = \sum_{i=1}^{N-1} y_{iN} = 1 \tag{2}$$

$$\sum_{i=1}^{N-1} y_{ir} = \sum_{j=2}^{N} y_{rj} \leq 1, for\, all\, r = 2,\ldots,N-1 \tag{3}$$

$$\sum_{i=1}^{N-1} \sum_{j=2}^{N} t_{ij} y_{ij} \leq T_{Max} \tag{4}$$

$$s_i + t_{ij} - s_j \leq M\left(1 - y_{ij}\right) \; for\, all\, i, j = 1\ldots N \;, \tag{5}$$

$$O_i \leq s_i \; for\, all\, i = 1\ldots N$$

$$s_i + d_i \leq C_i \; for\, all\, i = 1\ldots N \tag{7}$$

$$y_{ij} \in \left\{0,1\right\}. \tag{8}$$

The objective function (1) is to maximize the total reward of an accomplished task. Constraint (2) ensures that the route starts at task 1 and ends at task N. Constraint (3) ensures that the route starting at task 1 and ending at task N is connected and each task is accomplished at most once. Constraint (4) ensures that the route meets the maximal time (deadline) of each worker. Constraint (5) ensures the timeline of the worker's route. Finally, constraints (6) and (7) ensure each task can be accomplished only within specific time windows.

GRASP-TABU SEARCH

General GRASP-TB Method

The combination of various metaheuristic principles to construct a solid algorithm has yielded effective results and improved the advantages of using a single methodology to get better solutions. Among them GRASP-Tabu algorithms which have been largely employed to resolve difficult problems and used in different application such as the 0–1 quadratic knapsack, Pickup and Delivery Operations, School Timetabling, maximal covering location, and the Unconstrained Binary Quadratic Programming (UBQP),. GRASP conjunction with Tabu search was initially investigated in Laguna and González-Velarde (Díaz et al., 2017; RG González-Ramírez et al. ,2017 ;Souza et al., 2017; Souza et al., 2003; Wang et al., 2013**;** Yang et al., 2013).

GRASP-Tabu algorithm's main feature integrates the benefits of its constituent algorithms. The GRASP offers strong initial solutions and is used as an effective tool for diversification, while the algorithm Tabu has an excellent potential for the local method, also the solution is refined through it.

In this paper, the authors suggest a method that combines GRASP and Tabu search. When GRASP is completely constructing a new solution, the authors apply the tabu search procedure to optimize this solution to improve more the best solution found by GRASP, a simple tabu search algorithm is supposed to avoid the restrictions of local optimality (Yahyaoui et., 2018 ; Yang et al., 2013).

The GRASP algorithm is typically implemented as a multistart procedure, including of a randomized greedy solution construction step and the local search step to optimize the objective function. These two steps are achieved iteratively until the condition of a halt is reached.

The basic GRASP-Tabu Search algorithm (denoted by GRASP-TB) for finding an optimal route in spatial crowdsourcing follows this general scheme (Algorithm 1) and uses a dedicated greedy heuristic for solution construction(section GRASP Method) as well as tabu search (section Tabu Search Procedure) as its local optimizer (Feo & Resende,1995; Glover & Laguna, 1997). In the GRASP-TB method, the authors do not use the tabu algorithm inside the GRASP algorithm loop instead of the local search, they use it outside the loop to reduce the runtime.

```
Algorithm 1: GRASP –TB
Input: Instance of the problem ()
Output: Optimal route S*(T1, T2… Tn)
 f* = -∞ // objective value of S*
 S*= {};
Begin
     Repeat
       //see section GRASP Method.
       S0 ← Construct a greedy randomized adaptive solution ();
       // see section Tabu Search Procedure
       S ← Tabu_search (S0);
        If f(x)> f* then
           S* =S;
           f*=f(S);
        End
```

```
      Until a stopping criterion is met
End
```

Greedy randomized adaptive solution S_0 is constructed by GRASP Method and solution S is improved by Tabu Search Procedure.

GRASP Method

Algorithm Listing 2 illustrates of GRASP for the Route Planning Problem.

Generally, GRASP searches by repeated solution construction. In each iteration GRASP begin with an empty route (solution), iteratively adds Spatial Tasks (STask) to the partial route and returns a complete solution route. Next, a list of STask is created from existing STasks which contains only the start and end position of each route's worker, resulting in a list of tasks, namely STask list.

A heuristic value is specified for the whole Tasks. A threshold is determined by multiplying the difference between the highest and lowest heuristic values in the STask list through the parameter of GRASP (Alpha) (Vansteenwegen et al., 2010). STask list is ltered and the Restricted Candidate List (RCL STask) contains only STasks candidates with a heuristic value greater than the threshold.

Finally, a STask of the RCL is randomly selected and if STask is feasible, it added to the partial solution route. This construction procedure is repeated until no more feasible STasks candidates are found. When iterations conclude, the constructed solution route is then improved using another search method such as the tabu method.

```
Algorithm 2: GRASP
While (it is possible to accomplish new task) do
      Route = {}; //empty solution
      STask_list = add_existing_STasks(); //sent by requester
  While (Task_list not empty) do
        Calculate threshold value;
        RCL_STask = {}; // empty Restricted_Candidate_STask List
    For (each Task of STask_list) do
         Calculate H_STask // heuristic value of STask (H_STask)
       If (H_STask >= threshold) then
          Add STask to RCL_STask // build RCL Task out of the best
candidates;
          RCL_STask ← RCL_STask U {Stask};
        End
    End
          R_STask←Select Random_STask from RCL_STask
    If (R_STask feasible) then
           Add Random_Task to route// Update route;
           Route ← Route U {R_STask};
     End
           Remove TR from STask_list; // Update Task_list
           Remove all Tasks from RCL_STask; // Update RCL_Task
           Update STask_list;
           Update route;
   End
End
Return Best found Route; // (Solution_GRASP)
```

Tabu Search Procedure

Glover introduced the tabu search algorithm in 1986, describing controlled randomization to avoid optimal from the local and proposing a deterministic method. The algorithm for tabu search became very famous in solving problems with optimization by an approximate approach. It is now one of the most prevalent methods of metaheuristics. The specific characteristic of tabu search is the usage of memory, which saves information linked to the search process .Tabu search is successfully applied in solving many problems in different domains of application such as assignment, touring vehicle, logistic .etc. (Hvattum, 2016; Talbi, 2009). The key elements of the method suggested are described as follows:

Solution Representation

In solution encoding, the auteur represented the solution by using list contains the sequence of tasks for accomplish, Separate list is also used to store a set of tasks that are not within the solution. Example of encoding given in Figure 2 (a).

Initial Solution

To starting the algorithm, an initial feasible solution is needed. For the proposed algorithm, the initial solution is created by using GRASP method.

Neighborhood Exploration

For the suggested method, the following three move operators can be used

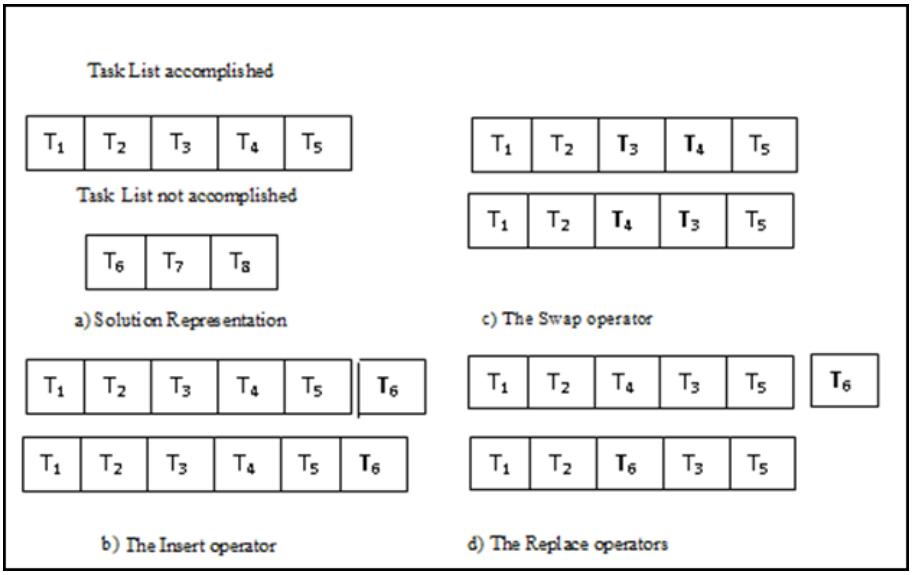
The Insert Operator

The Insert operator tries to insert (The insert operator is used for insertion of new Task in the route) a new task. For example insert task T6 in end of route. (see Figure 2 (b)).

The Swap Operator

The Swap operator is used between any two Tasks in the route (finding the shortest route between the selected tasks. For example swap between T3 and T4 (see Figure 2 (c)).

Figure 2. Solution Representation and Neighborhood Exploration



The Replace Operator

The Replace operator replaces a STask from the list of a STask out route (not yet accomplished). Replace task T_6 by T_4. (see Figure 2(d)).

Evaluation Function

The goal is to find an optimal route based on maximizing the total reward of an accomplished task. According to equation (1).

Tabu List

The key element of Tabu Search is the Tabu List. The aim of using Tabu list is to avoid revisiting solutions previous used.

Stopping Criteria

Usually, after a specified number of iterations, an algorithm is stopped or for a certain solution that are non-improving. For the proposed tabu search algorithm is finished after a fixed number of iterations.

The algorithm starts with the solution obtained by GRASP method and empty tabu memories. Then the algorithms are placed in a loop running M iterations. The overall neighborhood is being explored in a deterministic way. The algorithm employed the neighborhood procedure, which contains three basic moves, namely Swap, Insert, Replace and are used to create neighbor candidates. The Insert operator tries to insert (The insert operator is used for insertion of new STask in the route) a new task, from the non-included ones into one of the routes, while the Replace operator replaces a STask from the list of a STask out route (not yet accomplished). The Swap operator is used between any two STask in the route (finding the shortest route between the selected tasks).

Tabu restrictions are used to prevent any solution visited recently to be revisited. To avoid cycling in a small set of solutions, some attributes of recently visited solutions are stored in a tabu list which prevents them from occurring for a limited time. For our problem, the attribute used is a pair of tasks that have been swapped or replaced or inserted recently. A Tabu structure stores the iterations number for which a given pair of tasks are prohibited from the neighborhood. In each iteration, a non-tabu solution is searched by attempting all feasible combinations within applying the actual move. The best non-tabu solution is selected as the new solution currently available in the neighborhood. Typically, a better solution is adapted if a suitable new one is discovered otherwise the algorithm terminates if the criterion is satisfied (Sylejmani et al., 2012).

```
Algorithm 3: Tabu Search
S0=GRASP (); // Create initial solution,
             // given by grasp (Solution_Grasp) set of task
Sc=S0; // Current solution (Sc)
SBest =Sc; // Best solution (SBest)
Initialize tabu list (T) = {};
Move List = {insert, Swap, Replace};
While termination criterion not satisfied do
   For each move in Move List do
       Sc → SN   ; //Create complete neighborhood (SN) of current
solution (Sc)
                 //by applying current move
   For Sc∈SN   do
        If Sc ∉ T // Select best non tabu solution SBest from SN
(neighborhood)
       If (F (Sc) > F(SBest)) // F objective function
                SBest= Sc;       // Update best found solution
```

```
        End
      End
      S_c = S_Best; //Switch over solution, current solution S_c is
replaced by S_best
      Update tabu list T;
   End
 End
Return S_Best; /Best found solution
```

COMPUTATIONAL EXPERIMENTS

Test Instances

The algorithm is developed by using Java 1.7. The experiments are conducted by using a machine with an Intel CoreDuo 2.5 GHz processor with a Windows 7 operating system and 4GB of RAM. In the following subsection, the authors .describe the test set, and, we present the experimental results.

The algorithm is experimented by utilizing several instances. Such as The test problems for the OPTW in the literature were primly suggested by Righini and Salani (Righini & Salani, 2008), which are created from Solomon's and Cordeau et al.'s instances (Solomon, 1987; Cordeau et al., 1997). 48 Solomon's instances contain 100 nodes of series (c100, r100 and rc100). The instances in Cordeau et al. include 10 instances with several node numbers, ranging from 48 to 288 nodes (pr01–pr10). A further 37 instances were generated, and 27 instances are adapted from the Solomon-based instances (c200, r200 and rc200) and 10 instances are adapted from the Cordeau-based instances (pr11–pr20).

Table 1 details the T_{max} of Solomon's and Cordeau's instances. In this paper, the number of routes is equal to 1, which related to the OPTW problem. The GRASP-TB was tested by performing 10 runs

Table 1. T_{max} of Solomon's and Cordeau's instances

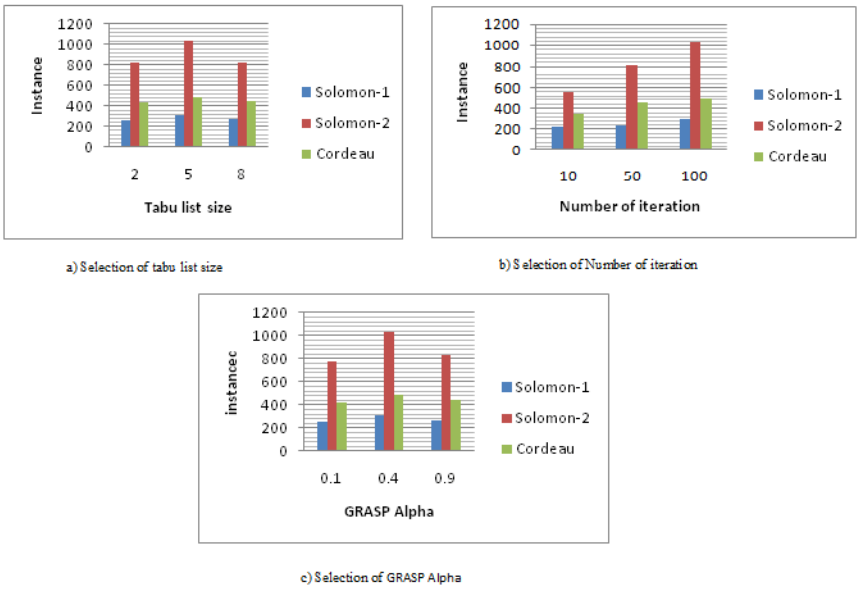
	Solomon's instances						Cordeau 's instances	
	C100	R100	RC100	C200	R200	RC200	pr01–pr10	pr11–pr20
Tmax	1,236	230	240	3,390	1,000	960	1000	1000

per each instance. The results of the proposed GRASP-TB are compared with the literature methods: Iterated Local Search (ILS), Ant Colony System (ACS), Enhanced Ant Colony System (Enhanced ACS), Slow Simulated Annealing (SSA), Granular Variable Neighborhood Search (GVNS), Iterative the Three Components Heuristics (I3CH) and Iterated Local Search (ILSn) (Gunawan et al., 2015; Gunawan et al., 2016; Karbowska-Chilinska & Zabielski, 2014).

Parameter Tuning

The experiments are conducted by executing the algorithm 10 times for every instance. Afterwards, the averages values of individual executions of instances are calculated. To analyze the algorithm performance, the authors use the obtained optimal values for the parameters, which are as follows: Tabu list size=5, Iteration number=100, the parameter of GRASP Alpha=0.4. The best combination of these parameters is calculated through computational experiments. From figure 3, it can be concluded that parameter produces better results.

Figure 3. Parameters of GRASP-TB method



a) Selection of tabu list size

b) Selection of Number of iteration

c) Selection of GRASP Alpha

Experimental Results

For each run, the percentage gap between the solution value achieved by the GRASP-TB and The BK is computes by:

$$Gap = \frac{BK - GRASP\_TB}{BK} * 100\%$$

Table 2, Table 3, and Table 4 provide detailed results of the GRASP-TB performance (the best reward and the computational time in seconds). The GRASP-TB was run 10 times; the best reward and the total time of the 10 runs are given in the tables. These tables also show the percentage gap between the best solution values (BK), and the GRASP TB (Gap-GB), to compare, the gap between BK and ILS too (Gap-ILS). The BK and algorithms results are taken from (Gunawan et al., 2016; Karbowska-Chilinska & Zabielski, 2014). These tables are composed of two same structure parts. The first column contains the instance name, the second column show values obtained by the GRASP-TB method. The third column reports the best known solution value (BK), fourth and fifth columns contain the Gap GB and Gap ILS receptively. Finally, the last column contains computational time.

The GRASP-TB method offered 47 new best known solutions, precisely for the Solomon-based instances, The GRASP-TB method improved the best known solutions by 62.5%, and for the Cordeau-based instances, it can enhance the best known solutions of up to 60%. The new BK obtained by the GRASP-TB is indicated in bold. As shown in Table 2, for the instances c100 and r100 the GRASP-TB method provides a better solution than ILS unlike with instance rc100 our algorithm is ineffective.

Notably from Table 3 for instances c200, r200, and rc200, we can notice that our method gives better solutions compared to other algorithms. Table 4 presents the results given on Cordeau's data sets. In the pr11-20 instances, wider time windows were given than in pr01-10, the GRASP-TB outperforms compared with other algorithms but in instances, pr05, pr18, pr19, and pr20, the reward of our algorithm is less than the others. As indicated in Table 2-4, we observe in three tables the time increases with node increase.

Table 2. Performance Results of GRASP-TB method for Solomon's test problems (Instance set 1)

Name	GRASP-TB	ILS	BK	Gap GB	Gap ILS	Time
C101	320	320	320	0	0	0.256
C102	400	360	360	-11.1	0	0.319
C103	410	390	400	-2.5	2.5	0.425
C104	410	400	420	2.4	4.8	2.54
C105	360	340	340	-5.9	0	2.107
C106	320	340	340	5.9	0	1.909
C107	390	360	370	-5.4	2.7	2.106
C108	390	370	370	-5.4	0	2.383
C109	390	380	380	-2.6	0	2.546
R101	286	182	198	-44.4	8.1	3.299
R102	297	286	286	-3.8	0	3.304
R103	298	286	293	-1.7	2.4	3.4
R104	298	297	303	1.7	2	2.275
R105	284	247	247	-15	0	3.7
R106	297	293	293	-1.4	0	2.855
R107	297	288	299	0.7	3.7	3.28
R108	298	297	308	3.2	3.6	2.324
R109	287	276	277	-3.6	0.4	2.575
R110	286	281	284	-0.7	1.1	2.754
R111	285	295	297	4	0.7	0.976
R112	287	295	298	3.7	1	0.045
RC101	259	219	219	-18.3	0	2.1
RC102	260	259	266	2.3	2.6	2.164
RC103	260	265	266	2.3	0.4	2.467
RC104	287	297	301	4.7	1.3	3.112
RC105	228	221	244	6.6	9.4	1.569
RC106	243	239	252	3.6	5.2	0.077
RC107	245	274	277	11.6	1.1	2.617
RC108	247	288	298	17.1	3.4	2.646

Table 5 explains the results of ILS, ACS, GVNS, SSA, I3CH, and the GRASP-TB method results. The table reports the average of Gap BK (AG) for each instance set. The average results obtained by the GRASP-TB are better than the average results of other algorithms. The GRASP-TB method is the best compared with other methods because the grand mean of AG is -4.03, whereas those other methods are 3.64, 2.09, 0.94, 1.71, 1.44, and 0.23. For instances c100, r100, c200, rc200, and pr01-pr10 our method gives the best score results compared with other methods (The results show that the algorithm performs well). The AG of GRASP-TB ranges from -12.81% to -4.47%, and the AG of others algorithms ILS, I3CH, ACS,SSA, GVNS and ILSn have wider ranges from 1.11% to 9.56%, 0.00% to 4.28%, 0 to 11.13, 0 to 3.71,0.55 to 3.17 and -0.04 to 1.33 respectively.

Table 3. Performance Results of GRASP-TB method for Solomon's test problems (Instance set 2)

Name	GRASP-TB	ILS	BK	Gap GB	Gap ILS	Time
C201	930	840	870	-6.9	3.4	8.754
C202	950	910	930	-2.2	2.2	11.4
C203	970	940	960	-1	2.1	10.435
C204	950	950	980	3.1	3.1	9.438
C205	930	900	910	-2.2	1.1	7.654
C206	950	910	930	-2.2	2.2	8.191
C207	930	910	930	0	2.2	9.083
C208	950	930	950	0	2.1	7.941
R201	1069	788	797	-34.1	1.1	11.343
R202	1071	880	930	-15.2	5.4	20.869
R203	1078	980	1021	-5.6	4	18.318
R204	1086	1073	1086	0	1.2	14.847
R205	1067	931	953	-12	2.3	13.871
R206	1085	996	1029	-5.4	3.2	18.09
R207	1092	1038	1072	-1.9	3.2	14.501
R208	1093	1069	1112	1.7	3.9	14.508
R209	1079	926	950	-13.6	2.5	22.758
R210	1051	958	987	-6.5	2.9	19.245
R211	1080	1023	1046	-3.3	2.2	20.06
RC201	1048	780	795	-31.8	1.9	10.481
RC202	1029	882	936	-9.9	5.8	23.086
RC203	1045	960	1003	-4.2	4.3	11.309
RC204	1111	1117	1140	2.5	2	12.79
RC205	1028	840	859	-19.7	2.2	18.068
RC206	1114	860	895	-24.5	3.9	16.55
RC207	1085	926	983	-10.4	5.8	13.415
RC208	1100	1037	1053	-4.5	1.5	13.056

For the instance rc100 the GRASP-TB method is not efficient its AG is 3.74, while ILS, ACS, SSA, GVNS, I3CH, and ILSn their AG are 2.92, 0, 0, 1.88, 1.66, and 0 respectively. For the instance r200 the GRASP-TB method gives better solutions compared with other algorithms, ILS, ACS, SSA, GVNS, I3CH, and ILSn. The AG of the GRASP-TB is -8.72 but the AG of other algorithms is 2.9, 2.19, 1.3, 2.45, 1.05, and 0.11 respectively. In instances, pr11-pr20 our algorithm performs well compared with ILS and ACS because the AG of the GRASP-TB is 4.47 and ILS, ACS is 9.56, 11.13 respectively and the solution quality achieved by method SSA, GVNS, i3CH, and ILSn is much better than GRASP-TB, their AG is 3.71, 3.17, 4.28, and 1.33 respectively. For the whole instance set, a negative value of AG indicates the improvement of some BK.

Figure 4 illustrates the variance of rewards over executions of Instance.

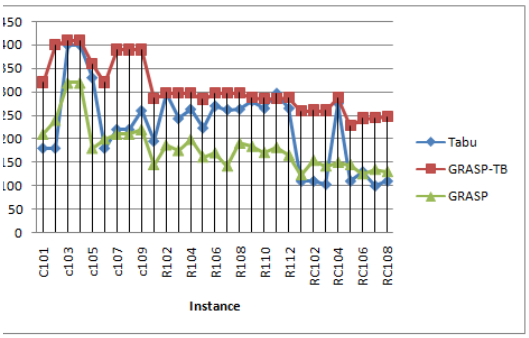
Table 4. Performance Results of GRASP-TB method for Cordeau's test problems

Name	GRASP-TB	ILS	BK	Gap GB	Gap ILS	Time
48 pr01	380	304	308	-23.4	1.3	0.967
96 pr02	467	385	404	-15.6	4.7	6.497
144pr03	530	384	394	-34.5	2.5	24.731
192 pr04	491	447	489	-0.4	8.6	50.854
240 pr05	583	576	595	2	3.2	118.767
288pr06	597	538	590	-1.2	8.8	149.796
72pr07	352	291	298	-18.1	2.3	3.271
144pr08	511	463	463	-10.4	0	22.15
216 pr09	513	461	493	-4.1	6.5	55.139
288pr10	558	539	594	6.1	9.3	143.811
48pr11	381	330	330	-15.5	0	0.816
96pr12	512	431	442	-15.8	2.5	6.007
144pr13	489	450	461	-6.1	2.4	25.183
192 pr14	512	482	567	9.7	15	53.033
240 pr15	564	638	685	17.7	6.9	90.162
288 pr16	528	559	674	21.7	17.1	241.778
72 pr17	382	346	359	-6.4	3.6	1.648
144 pr18	496	479	535	7.3	10.5	28.244
216 pr19	517	499	562	8	11.2	86.813
288 pr20	506	570	667	24.1	14.5	245

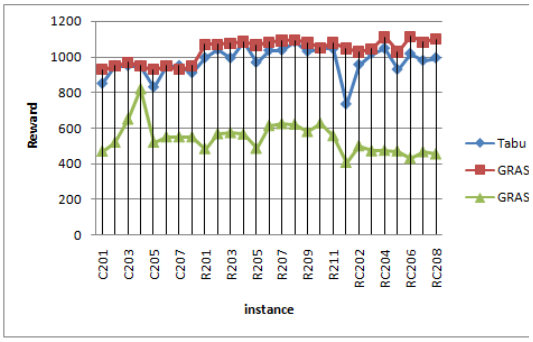
Table 5. Comparison of GRASP-TB to the state-of-the-art methods

		GRASP -TB	ILS	ACS	SSA	GVNS	i3CH	ILSn
instances		AG (%)	AG(%)	AG(%)	AG(%)	AG(%)	AG(%)	AG(%)
Solomon	c100	-2.73	1.11	0	0	0.56	0	0
	r100	-4.78	1.90	0	0.11	1.72	0.56	0
	rc100	3.74	2.92	0	0	1.88	1.66	0
	c200	-1.43	2.28	0.40	0.13	0.55	0.40	0
	r200	-8.72	2.90	2.19	1.3	2.45	1.05	0.11
	rc200	-12.81	3.43	1.23	0.96	2.53	2.68	-0.04
Cordeau	pr01-pr10	-9.96	4.74	1.06	0.98	0.56	1.07	0.34
	pr11-pr20	4.47	9.56	11.13	3.71	3.17	4.28	1.33
Average		-4.03	3.64	2.09	0.94	1.71	1.44	0.23

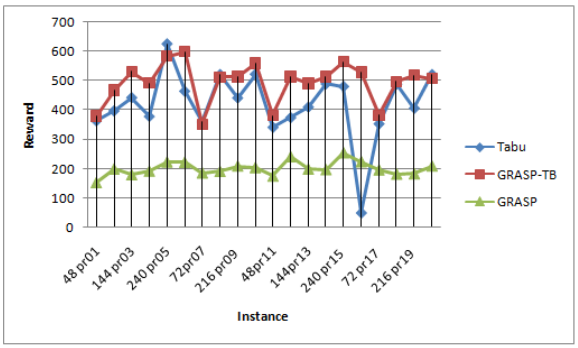
Figure 4. Comparison of GRASP-TB to Tabu and GRASP



iparison of GRASP-TB to Tabu and GRASP Solomon's test problems (Instance set 1).



b) Comparison of GRASP-TB to Tabu and GRASP Solomon's test problems (Instance set 2).



c) Comparison of GRASP-TB to Tabu and GRASP cordeau's test problems.

Figure 4 (a) presents results of GRASP-TB method, Tabu and GRASP for Solomon's instances (Instance set 1). As can be observed, the proposed GRASP-TB method gives better solutions compared to Tabu and GRASP algorithms. Figure 4 (b) shows results of GRASP-TB, Tabu and GRASP for Solomon's instances (Instance set 2). The authors can notice that GRASP-TB method gives better solutions compared to Tabu and GRASP algorithms. Figure 4 (c) describes results of GRASP-TB, Tabu and GRASP for Cordeau et al.'s instances. As can be observed, better quality solutions are obtained when the proposed GRASP-TB method is used.

CONCLUSION

The principal contribution of this study is an algorithm that solves the route planning problem in spatial crowdsourcing with GRASP with Tabu Search. The GRASP is a multi-start process. The more iterations of it mean the more solution space will be explored, and The Tabu search procedure was implemented in combination with the insert, replace and swap of three simple operators. It is used instead of the local search in the algorithm's improvement phase. Computational results have shown that the proposed algorithm is effective. Overall, the GRASP with Tabu Search algorithm surpassed the existing methods on the Cordeau et al. and the Solomon instances, also it provided better solutions compared to Tabu and GRASP algorithms. In further research, our intention to conduct experiments on large networks of real tasks with many workers instead of one worker, as well as other metaheuristic methods will be used for solving the Route planning problem such as bio-inspired algorithms.

REFERENCES

Amrollahi, A., & Ahmadi, M. H. (2019). What Motivates the Crowd: A literature review on motivations for crowdsourcing. In R. Lenart-Gansiniec (Ed.), *Crowdsourcing and Knowledge Management in Contemporary Business Environments* (pp. 103–133). IGI Gloabal.

Chen, C., Cheng, S. F., Gunawan, A., Misra, A., Dasgupta, K., & Chander, D. (2014). TRACCS: Trajectory-aware coordinated urban crowd-sourcing. In *Proceedings of the 2nd AAAI Conference on Human Computation and Crowdsourcing* (pp. 30-40). AAAI Press.

Chen, C., Cheng, S. F., Lau, H. C., & Misra, A. (2015). Towards city-scale mobile crowdsourcing: task recommendations under trajectory uncertainties. In *Proceedings of the International Joint Conference on Articial Intelligence* (pp.1113-1119). Buenos Aires, Argentina: AAAI Press.

Chen, Z., Cheng, P., Zeng, Y., & Chen, L. (2019).Minimizing maximum delay of task assignment in spatial crowdsourcing. *35th IEEE International Conference on Data Engineering*, 1454–1465. https://doi.org/doi:10.1109/ICDE.2019.00131

Cheng, P., Lian, X., Chen, L., Han, J., & Zhao, J. (2016). Task assignment on multi-skill oriented spatial, crowdsourcing. *IEEE Trans. Know. Data Eng, 28*(8), 2201-2215. DOI: 10.1109 /TKDE .2016.2550041

Cordeau, J.F., Gendreau, M., & Laporte, G. (1997). A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks, 30*(2), 105–119. doi: <105::AID-NET5>3.0.CO;2-G10.1002 /(SICI)1097-0037(199709)30:2

Costa, C. F., & Nascimento, M. A. (2020). Online In-Route Task Selection in Spatial Crowdsourcing. *Proceedings of the 28th International Conference on Advances in Geographic Information Systems*, 239–250. doi: doi:10.1145/3397536.3422215

Cura, T. (2014). An artificial bee colony algorithm approach for the team orienteering problem with time windows. *Computers & Industrial Engineering*, *74*, 270–290. doi:10.1016/j.cie. 2014.06.004

Deng, D., & Shahabi, C. (2016). Task selection in spatial crowdsourcing from worker's Perspective. *GeoInformatica*, *20*(3), 529–568. doi:10.1007/s10707-016-0251-4

Díaz, J.A., Luna, D.E., Camacho-Vallejo, J.F., & Casas-Ramírez, M.S. (2017). MS GRASP and hybrid GRASP-tabu heuristics to solve a maximal covering location problem with customer preference ordering. *Expert Systems with Application, 82*, 67-76. doi: 10.1007/978-1-4615-6089-0

Feo, T., & Resende, M. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization*, *6*(2), 109–133. doi:10.1007/BF01096763

Field Agent. (2019). Retrieved from https://www.fieldagent.net/

Gigwalk. (2019). Retrieved from https://www.gigwalk.com/

Glover, F., & Laguna, M. (1997). *Tabu Search*. Kluwer Academic Publishers.

González-Ramírez, R. G., Smith, N. R., Askin, R. G., Camacho-Vallejo, J.-F., & González-Velarde, J. L. (2017). A GRASP-tabu heuristic approach to territory design for pickup and delivery operations for large scale instances. *Mathematical Problems in Engineering*, *2017*, 1–13. doi:10.1155 /2017 / 4708135

Gunawan, A., Lau, H. C., & Lu, K. (2015). An iterated local search algorithm for solving the orienteering problem with time window. In G. Ochoa & F. Chicano (Eds.), *Evolutionary Computation in Combinatorial Optimization (LNCS)* (Vol. 9026, pp. 61–73). Springer. doi:10.1007/978-3-319-16468-7_6

Gunawan, A., Lau, H. C., & Vansteenwegen, P. (2016). Orienteering problem: A survey of recent variants, solution approaches and applications. *European Journal of Operational Research*, *255*(2), 315–332. doi:10.1016 /j.ejor.2016.04.059

Gunawan, A., Lau, H. C., Vansteenwegen, P., & Kun, L. (2017). Well-tuned algorithms for the team orienteering problem with time windows. *The Journal of the Operational Research Society*, *68*(8), 861–876. doi:10.1057/s41274-017-0244-1

Hu, Q., & Lim, A. (2014). An iterative three-component heuristic for the team orienteering problem with time windows. *European Journal of Operational Research*, *232*(2), 276–286. doi:10.1016/ j.ejor.2013.06.011

Hvattum, L. M. (2016). On the value of aspiration criteria in tabu search. *International Journal of Applied Metaheuristic Computing*, *7*(4), 39–49. doi:10.4018/ijamc.2016100103

Karbowska-Chilinska, J., & Zabielski, P. (2014). Genetic algorithm solving the orienteering problem with time windows. Advances in Systems Science, 609-619. doi : doi:10.1007/978-3-319-01857-7_59

Kazemi, L., & Shahabi, C. (2012). Geocrowd: enabling query answering with spatial crowdsourcing. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems* (pp. 189-198). ACM. doi: doi:10.1145/ 2424321.2424346

Labadie, N., Mansini, R., Melechovsk'y, J., & Calvo, R. W. (2011). Hybridized evolutionary local search algorithm for the team orienteering problem with time windows. *Journal of Heuristics*, *17*(6), 729–753. doi:10.1007/ s10732-010-9153-z

Labadie, N., Mansini, R., Melechovsk'y, J., & Calvo, R. W. (2012). The team orienteering problem with time windows: An lp-based granular variable neighborhood search. *European Journal of Operational Research*, *220*(1), 15–27. doi:10.1016/j.ejor.2012.01.030

Li, Y., Yiu, M. L., & Xu, W. (2015). Oriented online route recommendation for spatial crowdsourcing task workers. In Advances in Spatial and Temporal Databases (LNCS) (vol. 9239, pp. 137-156). Springer. doi: doi:10.1007/978-3-319-22363-6_8

Liao, C. C., & Hsu, C. H. (2013). A detour planning algorithm in crowdsourcing systems for multimedia content gathering. In *Proceedings of the 5th Workshop on Mobile Video* (pp. 55-60). Oslo, Norway: ACM. doi: doi:10.1145/2457413.2457426

Lin, S.W., & Yu, V.F. (2012). A simulated annealing heuristic for the team orienteering problem with time windows. *European Journal of Operational Research*, *217*(1), 94–107. doi:10.1016/j.ejor.2011.08.024

Ludwig, B., Zenker, B., & Schrader, J. (2009). Recommendation of personalized routes with public transport connections. In D. Tavangarian, T. Kirste, D. Timmermann, U. Lucke, & D. Versick (Eds.), *Intelligent Interactive Assistance and Mobile Multimedia Computing: Communications in Computer and Information Science* (Vol. 53, pp. 97–107). Springer. doi:10.1007/ 978-3-642-10263-9_9

Montemanni, R., & Gambardella, L. M. (2009). Ant colony system for team orienteering problem with time windows. *Found Comput Decis Sci*, *34*(4), 287–306.

OpenStreetMap. (2019). Retrieved from https://www.openstreetmap.org/

Righini, G., & Salani, M. (2008). New dynamic programing algorithms for the resource constrained elementary shortest path problem. *Networks*, *51*(3), 155–170. doi:10.1002/ net.20212

Shi, Z., Huang, H., Sun, Y. E., Wu, X., Li, F., & Tian, M. (2016). An efficient task assignment mechanism for crowdsensing systems. In Cloud Computing and Security (LNCS) (vol. 10040, pp.14-24). Springer. doi.1007/ 978-3-319-48674-1_2

Solomon, M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, *35*(2), 254–265. doi:10.1287/opre.35.2.254

Souffriau, W., Vansteenwegen, P., Vanden Berghe, G., & Van Oudheusden, D. (2013). The multiconstraint team orienteering problem with multiple time windows. *Transportation Science*, *47*(1), 53–63. doi:10.1287/ trsc.1110.0377

Souza, M., Ochi, L., & Maculan, N. (2003). A GRASP-Tabu search algorithm for solving school timetabling problems. In Metaheuristics: Computer Decision-Making. Kluwer Academic Publishers. doi: doi:10.1007/978-1-4757-4137-7_31

Talbi, E. G. (2009). *Metaheuristics from Design to Implementation* (Vol. 74). John Wiley & Sons. doi:10.1002/9780470496916

TaskRabbit. (2019). Retrieved from http://www.taskrabbit.com/

The multiconstraint team orienteering problem with multiple time windows. (n.d.). *Transp Sci, 47*(1), 53-63. doi:10.1287/trsc.1110.0377

Tong, Y., Zhou, Z., Zeng, Y., Chen, L., & Shahabi, C. (2020). Spatial crowdsourcing: A survey. *The VLDB Journal*, *29*, 217–250. doi:10.1007/s00778-019-00568-7

Vansteenwegen, P., Souffriau, W., Berghe, G. V., & Oudheusden, D. V. (2010). The City Trip Planner: An expert system fortourists. *Expert Systems with Applications*, *38*, 6540–6546. doi:10.1016/j.eswa.2010.11.085

Vansteenwegen, P., Souffriau, W., Vanden Berghe, G., & Van Oudheusden, D. (2009). Iterated local search for the team orienteering problem with time windows. *Computers & Operations Research*, *36*(12), 3281–3290. doi:10.1016/j.cor.2009.03.008

Wang, Y., Lu, Z., Glover, F., & Hao, k. (2013). Probabilistic GRASP-Tabu Search algorithms for the UBQP problem. *Computers & Operations Research*, *40*, 3100–3107. doi:10.1016/ j.cor.2011.12.006

Wang, Y., & Zhao, C., & Xu, S. (2020). Method for Spatial Crowdsourcing Task Assignment Based on Integrating of Genetic Algorithm and Ant Colony Optimization. *IEEE Access : Practical Innovations, Open Solutions*, *8*, 68311–68319. doi:10.1109/ACCESS.2020.2983744

Yahyaoui, H., Krichen, S., & Dekdouk, A. (2018). A decision model based on a grasp genetic algorithm for solving the vehicle Routing Problem. *International Journal of Applied Metaheuristic Computing*, *9*(2), 72–90. doi:10.4018/ijamc.2018040104

Yang, Z., Wang, G., & Chu, F. (2013). An effective GRASP and tabu search for the 0–1 quadratic knapsack problem. *Computers & Operations Research*, *40*(5), 1176–1185. doi:10.1016/j.cor.2012.11.023

Yuen, M.-C., King, I., & Leung, K.-S. (2015). Task matching in crowdsourcing. In *Proceedings of the International Conference on Internet of Things and 4th International Conference on Cyber, Physical and Social Computing* (pp. 409–412). Dalian, China: IEEE. DOI: doi:10.1109/ iThings/CPSCom.2011.128

Bouatouche Mourad is a Ph.D. Student at the USTO-MB University, interested in Optimization and Metaheuristics.

Khaled Belkadi is a Professor at the USTO-MB University, Oran, Algeria.