Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 60 (2015) 1081 – 1091

Procedia Computer Science

19th International Conference on Knowledge Based and Intelligent Information and Engineering Systems

Meeting Scheduling based on Swarm Intelligence

Hajer Salem[a], Ahlem Ben Hassine[b]

[a]*COSMOS Laboratory, Institut Suprieur de Gestion de Tunis (ISG Tunis), Tunisia*
[b]*COSMOS Laboratory, Ecole Nationale des Sciences de l'Informatique (ENSI), Tunisia*

Abstract

The meeting scheduling problem is complicated and time-consuming. In fact, finding the right combination to satisfy all attendees' availabilities, preferences and privacy seems to be arduous. Besides, some meetings need specific equipment (Telepresence video-conferencing). The objective of our research is to propose a new approach that organizes weekly meetings in a company. The main contributions of this work are *i*) a novel valued constraint satisfaction optimization formalization (VCSOP) of Meeting Scheduling (MS) problem and *ii*) a new MS solver based on Particle Swarm Meta-heuristic (MSSI for *Meeting Scheduling approach based on Swarm Intelligence*). This approach is intended to maximize both users' preferences and meetings importance while exploiting available resources properly. Meeting Scheduling based on Swarm Intelligence (MSSI) is an initiative to adapt swarm intelligence to MS problems. The experimental comparative evaluation shows that the MSSI approach is scalable and overcomes existing ones in terms of number of planned meetings for over constrained problems.

Peer-review under responsibility of KES International
Keywords: Meeting Scheduling problem, Particle Swarm optimization, Valued Constraint Optimization Problem ;

1. Introduction

The MS problem defines when and where a meeting can be scheduled. Finding the right compromise between attendees and resources availabilities is hard to fulfill. In this work we aim to solve the MS problem using the Particle Swarm Optimization (PSO) as a mean to handle over constrained instances and to keep a good quality of the solution. The main contribution of our work is a generic solution based on swarm intelligence for meeting scheduling problem (that we called MSSI for *Meeting Scheduling based on Swarm Intelligence*). The intended target is to represent ideally any real model for the MS problem, while respecting meetings importance, users preferences and keeping them confidential. In order to empirically prove the approach scalability, performance and limits, experiments were conducted and our proposed algorithm was compared to an existing approach in the literature.
The reminder of this paper is organized as follows: section 2 is a brief overview of recent works interested in the

* Corresponding author. Tel.: +216-52-88-4657 ;
E-mail address: hejer.salem@gmail.com

Peer-review under responsibility of KES International
doi:10.1016/j.procs.2015.08.153

meeting scheduling problem, section 3 is an introduction of Particle Swarm Optimization meta-heuristic and some of its applications in scheduling problems, section 4 depicts the problem definition and the proposed VCSOP model. In section 5, the basic notions of the PSO meta-heuristic are recalled. Then, in section 6 we adjust a PSO algorithm to the MS problem. Finally, we conclude the work in section 8.

2. The Meeting Scheduling Problem

Several research efforts, dealing with MS problems, were conducted and all aim to come up with the *best* solution. Amongst, [4] proposed an optimizing agent based meeting scheduling approach and focused on users preferences. Authors elaborated algorithms based on agent negotiation and reduced the search space to time intervals limited by the availability of participants. The main contribution of their work is to consider the search as a process of negotiation between software agents acting on behalf of participants in a meeting. Consequently, the approach is fully automated and planning responsibilities are entrusted entirely to software agents. In the work proposed in [1], authors handled this problem as a Dynamic Valued Constraint Satisfaction Problem and proposed the MSRAC approach (for *meeting scheduling with reinforcement of arc consistency*) in a multi-agent and dynamic environment. Authors considered users preferences and meetings importance. Agents have local goal (satisfying their local constraints), and global goal (maximizing the number of planned meetings). The authors prior work focused on maintaining arc-consistency while relaxing the user preferences, preserving at most the privacy of involved users and minimizing the number of exchanged messages.
The MS problem belongs to the NP-hard class of problems. Even if the difficulty of the problem depends on the instances, for over constrained instances, finding an acceptable solution may be very tedious. However for small instances (like team meeting) the solving process may be performed by the use of a complete algorithm such as in the work of [1]. In fact, the ABT (Asynchronous Backtracking) was used as a witness approach and had proved its efficiency for small instances (less than 15 meetings). Most significant researches focused on dealing with the MS problem in a distributed way in order to keep users privacy. However the majority of these efforts fall short to solve over constrained instances and have considered only small number of meetings and participants. In effect for a greater number of participants, the number of exchanged messages grows exponentially and consequently penalizes the execution time.

3. The particle Swarm Optimization

The Particle Swarm Optimization (PSO) is a population based optimization introduced by [10]. It is biologically inspired from the social behaviour of flocks of birds and fishes. The underlying algorithm is a simulation of the social environment. Analogically, the search space in PSO algorithm is called a swarm of particles. Each particle is a potential solution to the problem. Particles are in move in order to reach the optimal solution according to an objective function. A particle by itself has no power to solve a problem, nonetheless the algorithm power results from the particles' interactions. Since its introduction, the algorithm had known many enhancements leading to several versions of PSO. In order to improve its efficiency, authors proposed many adjustments related to the population dynamic, the swarm topologies, the parameters regulations, etc. The interested reader can, however, refer to [2] for an overview.
In the sequel, we will focus our interest on PSO algorithm for combinatorial problems. PSO was fast adapted to discrete problems. At the beginning, Kennedy and Eberhart [11] proposed a discrete binary version. Then in an attempt to streamline the discrete PSO, it has been illustrated by the Travel Salesman Problem (TSP) in [5]. Firstly, the work focused on the adaptation of PSO elements for the TSP, mainly the position, the velocity and the objective function. Secondly, authors tried to offer an approach to warn the algorithm from stagnation by the bias of an adaptive memory (called ReHope Method) that remembers how many time steps has been spent from the last improvement and according to that it adopts the adequate ReHope strategy. Another work based on PSO for TSP was elaborated in [15]. Authors proposed the concept of Swap operators and Swap Sequence.
Moreover, PSO gained considerable momentum for the solving of scheduling problems. Among others a proposed approach in [8] has applied PSO to solve a resource constrained project scheduling problem in a multi agent environment. More recent research [6] tackled the bi-objective load balancing problem by coming up with a discrete multi-objective particle swarm optimization approach.

In our research, we are interested in the generic meeting scheduling problem. Nonetheless, many variants exist in the literature such as the educational timetabling problems (course timetabling, exam timetabling, school timetabling), medical timetabling (patient scheduling, nurse scheduling, doctors scheduling), organizational timetabling, etc. PSO was widely applied to these problems and seems to be very competitive especially for educational timetabling ([3,13,7]. One of the most recent research in this field is the work proposed in [14]. Authors directed their research on the high school timetabling problem. The PSO algorithm was followed by a local search algorithm (a problem specific heuristic) in order to improve the quality of the obtained PSO solution. Authors replaced the basic notion of velocity by a *swap by probability* procedure where two time slots in the same particle structure are swapped according to a probability. This idea was also performed in the work of [3]. Nevertheless, the algorithm consisted of taking a copy of a slot from the local best particle and another from the global best particle, then performed a mutation to the current particle. Another type of timetabling was tackled in [9]. Researches directed their efforts to the medical timetabling and tackled the patient scheduling problem. PSO was used as an optimization agent in a multi-agent environment. The main objective of the PSO agent was to minimize the waiting time of a patient in a hospital. This work main limit is the complexity of the particle structure. In fact, a particle representation is a schedule of patients' operations. In other words, a set of patients will perform, each one, a set of operations on a set of resources. A particle position is the resulted schedule. The particle structure could not be directly evaluated by an objective function computation, so authors used a decode method at each step which is time consuming.

We concluded that PSO presents satisfactory results for constrained optimization problems. However, as far as we know, it has not been applied to a generic meeting scheduling problem. This work is an attempt to find high quality solutions for meetings like Employers organization meetings, where on the one hand, employers have already busy schedules and want to keep them private and on the other hand they have to attend organization meetings to discuss their future policies. The Employers scheduling problem is treated as an example and our model can be easily adapted to any kind of meetings.

4. MS proposed formalism

As stated before, the MS problem is subject to many restrictions related to availability constraints, resources constraints, preferences, meetings importance, etc. Besides, many attributes participate in this process. In fact, we define a meeting by the following attributes:

- An initiator: A person that proposes the meeting. The initiator usually has a preference for a room because of its capacity or the existing technical equipments. This preference is expressed as follows:
  - $Pref_{P_{ij}}^{S_i}$: A degree of preference of a participant P_{ij} to a room meeting S_i.
- A duration : Each meeting has an expected duration.
- A room : The room were the meeting will take place. This room has a limited capacity where:
  - $Capacity(s_i)$: the number of seats in a meeting room S_i.
- An importance: Each meeting has an importance degree that is function of the initiator grade, the meeting category, the meeting subject and basically the number of attendees. It is computed according to Eq:(1):

$$Imp_{x_i} = f(subject, initiator, nbr_{part}, category) \tag{1}$$

- $P = \{P_1, P_2, P_3, .., P_p\}$: Each meeting has a set of participants. Let p be the attendees number to the meeting i. Each participant has an order of preferences for meetings timing expressed by a preferences calendar.
  - Preferences calendar: The preference is computed to reflect an attendee non availability and his/her most suitable time to attend a meeting. A cell in the calendar expresses a user preference for a time slot. The preference is a number between 0 and 1; 0 expresses the non availability of the employee for a time slot to be on meeting; 1 expresses the most suitable time to have a meeting. We use a global constant AP to ensure the fairness between all the employees. It is equal to the sum of preferences in a user calendar [4]. Hence, each participant has to dispense his preferences over his agenda time slots. AP differs according

to the user position in the company. In fact, the employees' availabilities depend on their degrees of responsibility in the company.

In order to find a feasible calendar to schedule that maximizes the total number of planned meetings, especially important ones and that maximizes the solution quality (users preferences, meetings' importance, etc.), we propose to consider two types of constraints: hard constraints and soft constraints.

- *Hard constraints* are related to the availability of attendees. In fact, two different meetings could not have the same participant while they have overlapping time slots and in the same way, they could not be assigned to a same room Eq.(2). Other hard constraints are related to rooms capacities because a room cannot accept a number of attendees greater than its number of seats Eq.(3). A precedence constraint ensures the precedence order between meetings that have related subjects Eq.(4). For instance, a decision made in the first meeting will influence decisions in the other meetings or two related courses have certainly a precedence order.
- *Soft constraints* are related to the improvement of the quality of the solution. One important aspect of a good solution is the percentage of important planned meetings. For this reason, we added a constraint that encourages the scheduling of important meetings at first Eq.(6). We also directed our concern to the concentration capacity of attendees and decided that a meeting should not overtake four hours Eq.(5).

In order to evaluate the quality of potential solutions and to reach the optimal one, we propose an *objective function* defined by the instantiation of all variables of the problem. Solving a MS problem consists in finding $sl^* \in sl$ the optimal solution while satisfying all the hard constraints and maximizing the satisfaction of the soft constraints. The objective function Eq.(8) ensures the maximization of the planned meetings number Eq.(10), a compromise between users preferences Eq.(11, Eq.12), the percentage of satisfaction of important meetings Eq.(13) and resources allocation evaluation Eq.(14). As depicted in Eq.(8), the objective function embodies coefficients to adjust the weights of its different components. The user can settle weights according to his/her needs. We have to notice that users' preferences are computed first according to a local satisfaction of a user preferences which is ensured by Eq.(12). It is the sum of his/her different preferences to all meetings that he/she has to attend. Secondly, the global preference returns the sum of all participants local preferences. Moreover, the fair resources allocation is set according to Eq.(14) where $Cost(Sl_i)$ represents the cost of unused room seats for all planned meetings. The objective is to minimize unused resources and try to not allow the using of large rooms for small number of attendees. This idea may be generalized to any other type of resources (technical equipment, room classes etc.).
The formulation proposed in this paper is inspired by the work of [1]. It is an attempt to overcome its limits such as real world limited resources, available rooms, precedence relationships between meetings, etc. Therefore, we propose in the following a novel VCSOP formalization for any MS problem that reflects most ideally real world problems. We consider a MS problem as a Valued Constraint Satisfaction Optimization Problem (VCSOP) defined by a quintuple (X, D, C, S, φ, F) [1][2] where:

- $X = \{x_1, ..., x_n\}$ Set of n meetings to plan, each x_i is defined by a pair $(x_i.E, x_i.S)$ where $x_i.E$ is the time slot for a meeting x_i and $x_i.S$ is the available room assigned to the meeting x_i. For each x_i we assign a value $\lambda \in [0..1]$ reflecting its importance,[1].

- $D = \{D_1, ..., D_n\}$ is a set of n domains where:
  $D_i = (E_j, S_k) / E_j = (st_j, et_j)$, $j \in \{1..q\}$, $st_j < et_j$, and $S_k \in \{S_1, .., S_r\}$, $k \in \{1..r\}$,
  where st is the meeting start time and et is the meeting end time.
- $C = \{C_1, ..., C_k\} = C_{hard} \cup C_{soft}$ the set of constraints between variables. As stated before, constraints are divided into *Hard constraints* and *Soft constraints* and are formulated as follows:

[1] Note that for each no planned meeting in a solution, its importance is set to zero

1. *Hard constraints*
   - The availability constraint:
   $$C_{ij}^{alldiff} : x_i \neq x_j \begin{cases} if Part(x_i) \cap Part(x_j) \neq \oslash \\ or\ overlap(x_i.E_i, x_j.E_j) \oslash \\ or\ overlap(x_i.S_i, x_j.S_i) \end{cases} \quad (2)$$
   Represents the set of hard constraints between x_i and x_j having a common participant or taking place at the same room at overlapping[2] times.
   - The capacity constraint:
   $$C_i^{Cap} : Capacity(x_i.s) \geq \|Part(x_i)\| \quad (3)$$
   - The precedence constraint:
   $$C_{ij}^{Preced} x_i.E < x_j.E \ \ if\ precedes\ (x_i, x_j) \quad (4)$$
2. *Soft constraints*
   - The duration constraint:
   $$C_i^{Duration} : duration(x_i) \leq 4^h \ \forall i \in \{1, .., n\} \quad (5)$$
   - The importance constraint :
   $$C_{ij}^{imp}\ x_i.E < x_j.E\ if\ \lambda_i > \lambda_j \quad (6)$$
   where λ_i indicates the degree of the meeting x_i.

- S the valuation structure defined by $(E, >, \otimes)$ where E is the set of possible valuations. E is a set which is totally ordered by the operator $>$, provided with a minimum element $\perp$ which means the maximum constraints dissatisfaction and a maximum element T which means the maximum constraints satisfaction. E is equipped with an internal composition law $\otimes$.

- φ : The valuation function: $\varphi : C \rightarrow E$

$$C \rightarrow \varphi(C). \quad (7)$$

where:
$\forall C_i \in C_{hard}, \varphi(C_i) = 1$
$\forall C_i \in C_{soft},\ \varphi(C_i) = w_i\ , w_i \in [0, 1]$

- $F(sl_i)$ is the objective function to optimize and $sl_i \in$ sl a solution of the problem defined by the instantiation of all the variables of the problem where sl is the set of all possible solutions.

$$F(sl_i) = (\otimes_{sl_i \in sl}(NbrPlanned_{Meeting}(sl_i), Global_{Pref}(sl_i), Importance(sl_i), Cost(sl_i))) \quad (8)$$

$$F(sl^*) = Max_{sl_i \in sl} F(sl_i) \quad (9)$$

The used normalized functions ($NbrPlanned_{Meeting}$, $Global_{pref}$, $Importance$, $Cost$) are formulated as follow:

$$NbrPlanned_{Meeting}(sl_i) = \left\|X_p\right\|, \quad (10)$$
where $X_p \leq X\ \forall x_j \in X_p,\ x_j.E \neq 0,$ *and* $x_i.e \neq 0\ j \in \{1, ..., n\}$

$$Global_{pref} = \sum_{x_i \in X, i \in \{1,..,n\}} \frac{Local_{pref}(x_i)}{n} \quad (11)$$

[2] $overlap(x_i.E_i, x_j.E_j)$ a function that indicates whether the time slot of the meeting x_i overlaps the of x_j.

$$Local_{pref}(x_i) = \begin{cases} \sum_{P_j \in Part(x_i)} \frac{Pref_i^{P_j}}{\|Part(x_i)\|} \\ 0 \; otherwise \end{cases} \tag{12}$$

$$importance(sl_i) = \frac{\sum_{x_j \in sl_i, j \in \{1,..,n\}} \lambda_j}{n} \tag{13}$$

$$Cost(S\,l_i) = \frac{\sum_{x_i \in X_p, i \in \{1,..,n\}} ct(x_i)}{\left\|X_p\right\|} \tag{14}$$

5. Particle Swarm Optimization

As stated before, the Particle Swarm Optimization is a meta-heuristic inspired from the social behaviour of animals that live in groups. Researchers imitate this behaviour to solve computational problems. A particle in a PSO swarm has a velocity, an actual position, remembers its best previous position which represents its internal memory and is informed about the position of the best neighbour which represents its social memory. A particle move in the search space is determined by a compromise between three influences: the particle follows its own way according to its velocity, follows its best previous position and follows the position of the best neighbour. Hence, a velocity controls the direction of a particle. It is obtained according to the equation Eq.(15) as mentioned in [5] where the position of the best neighbour called *gbest*, the best previous position of the particle called *pbest* and its actual position is called *present*. The present position is obtained according to equation Eq.(16) [5]. The PSO meta-heuristic aims to explore the whole search space by intensifying and diversifying the search in order to find the optimal solution which maximizes the objective function.

$$V_{i,t+1} = w \otimes V_{i,t} \oplus c_1 \otimes r_1 \otimes (pbest_{i,t} \ominus present_{i,t}) \oplus c_2 \otimes r_2 \otimes (gbest_{i,t} \ominus present_{i,t}) \tag{15}$$

$$present_{i,t+1} = present_{i,t} \oplus V_{i,t+1} \tag{16}$$

where

- V : the velocity
- X_i : the present position of a particle
- c_1, c_2 : two coefficients that control the involvement of the particle memory (his best position) and the neighbourhood memory (the best position in the neighborhood) in future decisions (future positions of the particle).
- r_1, r_2 : two random functions [0,1] perform accuracy variations.
- w : is a weighting coefficient that controls the involvement degree of the particle previous velocity in its future decisions.

6. Meeting Scheduling approach based on Particle Swarm Optimization

6.1. A particle encoding

The swarm is composed of p particles. At each iteration of the PSO algorithm, each particle looks for a possible schedule of meetings while maximizing an evaluation function that is in this work the objective function proposed in section 4 Eq.(8). Hence, each particle returns a solution to the problem. As aforementioned in (sec:4), a meeting is defined by its initiator, subject and participants. A solution consists in assigning times and places to meetings under constraints. Hence, a particle is coded as a sequence of meetings assigned to their respective time slots $E = (st, et)$ and rooms (S). We depict an example of a particle position as follows:

[1] ct(x_i)= Capacity($x_i.s$)-$\|Part(x_i)\|$

$X = (x_1.E_1, x_1.S'), (x_4.E_1, x_4.S''), (x_7.E_2, x_7.S'), (x_8.E_3, x_8.S''')$

This particle code means that : i) meetings x_1 and x_4 are planned at the same time slot E_1 but in different meeting rooms, ii) meetings x_1 and x_7 are planned at different time slots and in the same meeting room, and iii) x_8 is planned at another time slot and another meeting room. We have to notice that the sequence is in a temporal order. In this example x_1 and x_4 are planned before x_7 and x_7 before x_8.

6.2. The global dynamic of MSSI

6.2.1. Generating the first population

To test the proposed approach MSSI, we implemented a Meeting Scheduling based on Swarm intelligence prototype. In the first place, we generated a first population of l particles. In other words, we tried to find l solutions to the problem while respecting hard constraints. These solutions are not optimal. Particles are then evaluated respectively according to their objective functions. *Gbest* is assigned to the best particle and for each particle we initialized *pbest* equal to *present* as illustrated in the flowchart in Fig.(1).
The search for a possible schedule of meetings is done according to a Generate and Test algorithm. Firstly, for each particle the algorithm tries to find a possible schedule. Schedules are different for each particle in order to ensure diversity in the first population. We suggest, for each particle to begin the search from a different meeting that is selected randomly. We have to notice that in this random search we are not interested in maximizing the objective function. Our unique interest is to find l first schedules to initialize the first population of the PSO algorithm.

6.2.2. The velocity

In order to define the PSO dynamic, firstly we have to define an operator which when applied to a position gives another one. In PSO, it is ensured by the velocity. Hence we perform the difference between two positions illustrated in Eq.(15) by the operator ($\ominus$) according to a ***difference procedure***:

We consider the difference between two positions as a meeting that do not figure in the same time slots in two different positions or pairs of meetings that are different and have the same time slots in two different positions as illustrated in the next example.

Example 1:
$Position\ p_1 : (x_1.E_1, x_1.S'), (x_4.E_1, x_4.S''), (x_7.E_2, x_7.S'), (x_8.E_3, x_8.S''')$
$Position\ p_2 : (x_1.E_1, x_1.S'), (x_4.E_2, x_4.S'), (x_5.E_2, x_5.S'''), (x_8.E_3, x_8.S''')$
$Difference : ((x_4.E_1, x_4.S''), (x_4.E_2, x_4.S')), ((x_7.E_2, x_7.S'), (x_5.E_2, x_5.S'''))$

This procedure could be elucidated by the fact that a position p_1 has a better objective function than another position p_2 if meetings in p_1 are planned at better moments and/or places. Let us consider the precedent example and let us suppose that p_1 is better than p_2. Hence, we deemed that meetings $((x_4.E_1, x_4.S''), ((x_7.E_2, x_7.S')$ are responsible of p_1 efficiency.
We remind that according to Eq.(15) , we need to compute two differences :

- $pbest_{i,t} - present_{i,t}$: The difference between these two positions provides best movements done for a *pbest* particle compared to the *present* particle. Therefore, these permutations are the **internal** effects that help the particle to converge to the optimal position.

- $gbest_{i,t} - present_{i,t}$: The difference between these two positions provides best movements done for a *gbest* particle compared to the *present* particle. Therefore, these permutations are the **external** effects that help the particle to converge to the optimal position.

The result of *the difference procedure* is a set of couples of meetings from both ($pbest_{i,t} - present_{i,t}$) and ($gbest_{i,t} - present_{i,t}$). We apply coefficients (c_1, c_2) to the obtained couples and such as in the method suggested in [5], we obtain two smaller sets of couples. Regardless of the redundant positions, we sum the two sets resulted from ($pbest_{i,t} - present_{i,t}$) and ($gbest_{i,t} - present_{i,t}$). Anew, the result is a set of meetings' couples that represent the velocity.

6.2.3. Compute the new position

As depicted in Eq.(16), a position $X_{i,t+1}$ is the result of a velocity on a previous particle position $X_{i,t}$. In order to define the operator ($\oplus$) between a position and a velocity for a MS problem, we suggest to replace meetings' time slots/rooms in the current position ($X_{i,t}$) by meetings' time slots/rooms such as in the best position: ***swap procedure***. Hence, many situations may occur. Namely a meeting could figure in a side of a velocity meetings couple and did not in the other. Here, we distinguish two cases: *i*) a meeting exists in the position *pbest/gbest* and does not in the position $X_{i,t}$. Then, we schedule it at the time slot as it exists in pbest/gbest: ***insert procedure***. *ii*) Furthermore, if a meeting exists in $X_{i,t}$ and did not in the *pbest/gbest* position, we check whether it has never been scheduled. In the hope to emphasize diversifying, we preserve it for the $present_{i,t+1}$ position, otherwise we do not consider it in the schedule. We have been inspired by the *aspiration criteria* of the tabu search where we aim to preserve rare solutions.
We have also to remind that the swap and insert procedures could not be applied to all particles' positions. In fact, in many cases a new meeting's time slot/room insertion will break some hard constraints (such as participants/room availability). In the case of a broken room availability constraint, the algorithm looks for, at first, a meeting room that has not been used yet as an attempt to diversify the search, otherwise it merely looks for an available room. In the same way, when the new meeting time slot breaks a participant availability constraint, the algorithm tries to schedule the meeting at a new time slot in the same day.

6.2.4. The main algorithm

To set the trend we depict a flowchart describing the main algorithm in Fig.1. After the generation of the first population as explained in sec.(6.2.1), we evaluate particles according to their objective function (OF) then we initialize *pbest* and compute *gbest*. Particles in the swarm have to perform movements. We apply the velocity to each particle position as explained in sec.(6.2.2). We evaluate particles in the swarm and we update *pbest* and *gbest*. This treatment is repeated until a stopping criterion.

6.2.5. The stopping criterion

We have to remember that the convergence towards an optimal solution is not guarantee. Hence, the algorithm stops generating neighborhood when for the k last iterations the algorithm still not improving the values of the *gbest* objective function, or the fixed iteration number is achieved. We notice that the iteration number and k are fixed in experiments.

7. Experimental evaluation

A first native version of the proposed MSSI was implemented in an object oriented programming language (Java) using Eclipse environment. The parameters used are: N, the neighborhood size fixed to 150, n meetings, p participants in each meeting, r meeting rooms, c_i velocity coefficients fixed to $c_1 = c_2 = 2$, an inertia coefficient w initially set to 0.4. If the objective function remains stable or deteriorates during more than 15 iterations, then this coefficient is incremented by 0.2. Also, in these experiments all coefficients which weights the objective function are set to one as follows:

$$F(Sol_i) = NbPlanned_{Meeting} + Global_{Pref} + Importance - \sum_{i=1}^{n} Cost(S_i) \tag{17}$$

In order to figure out the efficiency of our proposed approach we assessed it through a comparison with an existing approach [1] MSS (for *meeting scheduling solver*). We intended to conduct fair experiments, consequently, we implemented a centralized version of MSS. Also, both approaches were performed on the same set of random problems. However, we have to notice that MSSI is subject to a higher number of constraints compared to MSS. In fact, the

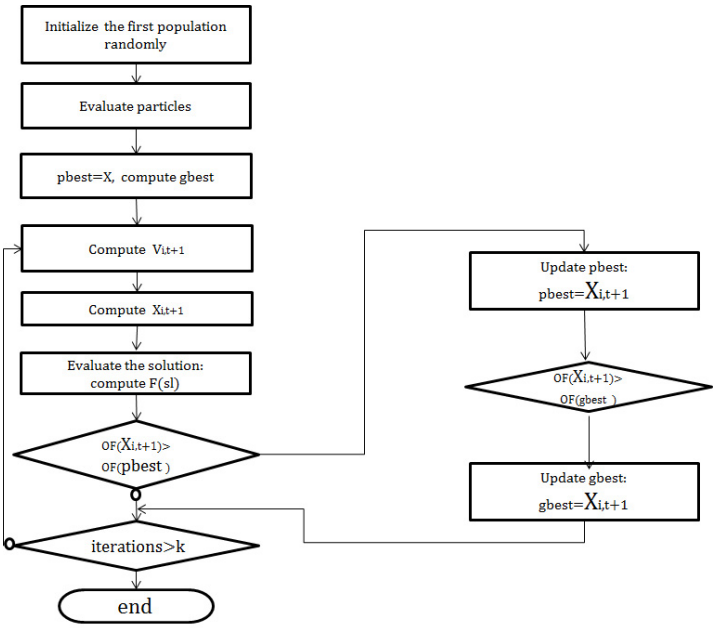

Fig. 1. The flowchart of the proposed MSSI algorithm

approach proposed in[1] does not consider resources constraints. The main goal of these experiments is to highlight the usefulness of using particle swarm optimization especially for most hard problems. Random problems were generated while considering an important number of participants $p \in \{40, 60\}$, a varied number of meetings $n \in \{10, 30, 50, 70\}$ and a fixed number of rooms $r = 5$ with different capacities. Meeting subjects, rooms capacities and participants calendars were generated and assigned randomly while trying to be as much as possible realistic. In summary for each triplet (n, p, r) we generated 30 random MS problems and then we considered the average of the obtained results. Hence, each curve reflects the behavior of MSSI or MSS approach when we fix the number of participants and we vary the number of meetings.

We carried out two kinds of experiments. The main goal of the first experiments was to evaluate the *quality of the obtained solution*. For this purpose we computed the number of planned meetings. The obtained results express that the number of scheduled meetings by the MSSI approach is greater than of MSS as shown in Fig.(2.a). The difference achieves 25% in the case of $(n = 10, p = 60)$, and 20% in the case of $(n = 30, p = 60)$ and 15% in the case of $(n = 30, p = 40)$. These results can be vindicated by the fact that the use of velocity ensures a balance between intensifying and diversifying yielding to an exhaustive scan of the search space for better solutions. However, it resulted in a little deterioration around 15% of the objective function, as shown in Fig.(2.b) and can be elucidated on the one hand by the fact that MSSI approach is subject to more constraints, and on the other hand the decrease of the objective function is caused by the increase of the planned meetings number(case of $p = 40$ and $n = 70$).

In the case of over-constrained problems, MSSI approach succeeds to plan a greater number of meetings while keeping a satisfactory objective function. However, it is not oriented for small instances of meeting scheduling problems. In fact, for these instances it is inadequate to use a meta-heuristic.

In the second kind of experiments, we are interested in the evaluation of the *scalability* of the proposed approach. For this purpose we tested the CPU time consumed by our approach compared to that consumed by MSS to find optimal solutions. Results are depicted in Fig.(3). The two curves that depict MSSI behaviour have the same shade. The participant number does not influence the CPU time consumed. However, it is proportional among meetings and is caused by the number of alterations that the MSSI performs during the solving process. The obtained results show that the MSSI approach requires, in most cases, more CPU time than the MSS approach (the difference noticed between MSSI and MSS in CPU time is about 0.6 *seconds*). This is insignificant since the average of CPU time consumed

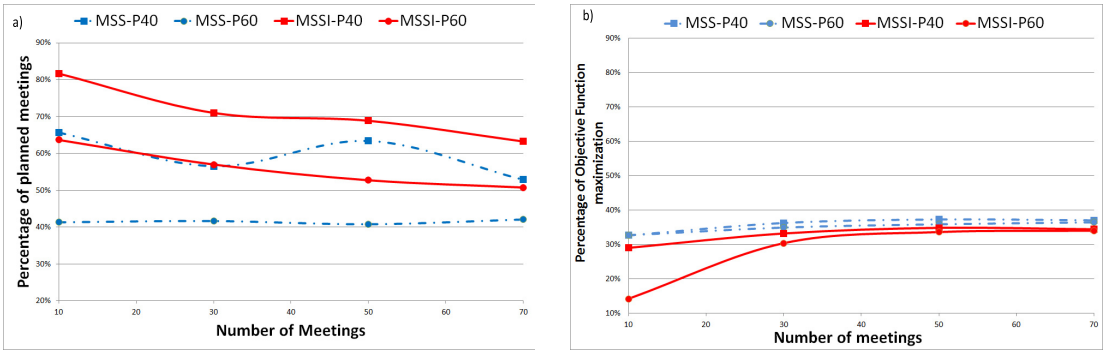

Fig. 2. a) The percentage of planned meetings; b) The percentage of Objective function maximization

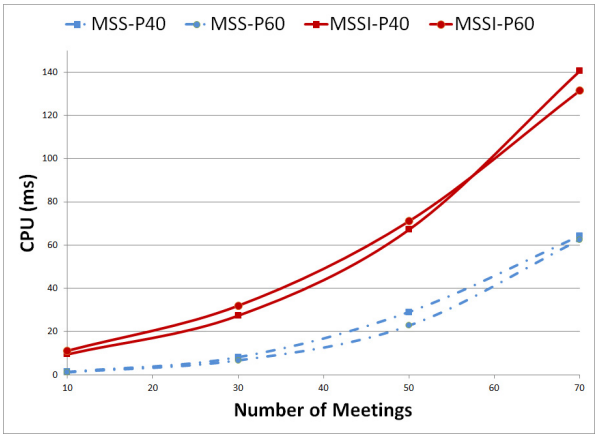

Fig. 3. The CPU time consumed by MSS and MSSI approaches

does not exceed one second.

8. Conclusion

We proposed in this work *i*) a novel formalization of any generic MS problem and *ii*) a new centralized and scalable approach MSSI (for *complex meeting scheduling problem based on swarm intelligence*). Our aim is to reflect the real world requirements such as: user preferences, user non-availability, room availability, room capacities, importance of the meeting, etc. For this purpose, in our new MS proposed formalization, we used hard constraints to prohibit, *i*) the scheduling of meetings having common participants at overlapping times, *ii*) the scheduling of different meetings at the same rooms and at overlapping time slot, *iii*) the scheduling of meetings in an inappropriate meeting room and *iv*) the violation of the precedence order between meetings. We used, as well, soft constraints to express user preferences and the proper and optimum use of resources. As for the MSSI approach, the proposed objective function focuses on maximizing the number of planned meetings and the participant preferences, while ensuring fairness between them. Also, this function attempts to improve the resources allocation while reenforcing meetings importance order. Experiments were conducted in terms of number of scheduled meetings, CPU time and objective function and the obtained results show that our approach succeeds to improve the quality of the obtained solution within a reasonable CPU time.

As for our future work, we will try to bring some enhancements to the approach by proposing an intelligent procedure to generate the first population and then increase the number of particles.

References

1. BenHassine, A., and T. B. Ho, 2007, An agent-based approach to solve dynamic meeting scheduling problems with preferences: Engineering Applications of Artificial Intelligence, v. 20, p. 857-873.
2. Bratton, D., J. Kennedy, and Ieee, 2007, Defining a standard for particle swarm optimization: 2007 Ieee Swarm Intelligence Symposium, p. 120-127.
3. Chu, S.-C., Y.-T. Chen, and J.-H. Ho, 2006, Timetable scheduling using particle swarm optimization: ICICIC 2006: First International Conference on Innovative Computing, Information and Control, Vol 3, Proceedings, p. 324-327.
4. Chun, A., H. Wai, and R. Y. M. Wong, 2003, Optimizing agent-based meeting scheduling through preference estimation: Engineering Applications of Artificial Intelligence, v. 16, p. 727-743.
5. Clerc, M., 2004, Discrete particle swarm optimization, illustrated by the Traveling Salesman Problem: New Optimization Techniques in Engineering, v. 141, p. 219-239.
6. Dahmani, N., S. Krichen, and Ieee, 2013, On solving the bi-objective aircraft cargo loading problem: 2013 5th International Conference on Modeling, Simulation and Applied Optimization (Icmsao), p. 6.
7. Irene, H. S. F., D. Safaai, M. Hashim, S. Zaiton, and Acm, 2009, University Course Timetable Planning using Hybrid Particle Swarm Optimization: World Summit on Genetic and Evolutionary Computation (Gec 09), p. 239-245.
8. Jarboui, B., N. Damak, P. Siarry, and A. Rebai, 2008, A combinatorial particle swarm optimization for solving multi-mode resource-constrained project scheduling problems: Applied Mathematics and Computation, v. 195, p. 299-308.
9. Kanaga, E. G. M., and M. L. Valarmathi, 2012, Multi-agent based Patient Scheduling Using Particle Swarm Optimization: International Conference on Communication Technology and System Design 2011, v. 30, p. 386-393.
10. Kennedy, J., R. Eberhart, and Ieee, 1995, Particle swarm optimization: 1995 Ieee International Conference on Neural Networks Proceedings, Vols 1-6, p. 1942-1948.
11. Kennedy, J., R. C. Eberhart, and Ieee, 1997, A discrete binary version of the particle swarm algorithm: 1997 IEEE International Conference on Systems, Man, and Cybernetics - Computational Cybernetics and Simulation (SMC 97), p. 4104-4108.
12. Miguel, I., and Q. Shen, 1999, Hard, flexible and dynamic constraint satisfaction: Knowledge Engineering Review, v. 14, p. 199-220.
13. Qarouni-Fard, D., A. Najafi-Ardabili, M. H. Moeinzadeh, S. Sharifian-R, E. Asgarian, J. Mohammadzadeh, and Ieee, 2007, Finding feasible timetables with particle swarm optimization: 2007 Innovations in Information Technologies, Vols 1 and 2, p. 294-298.
14. Tassopoulos, I. X., and G. N. Beligiannis, 2012, A hybrid particle swarm optimization based algorithm for high school timetabling problems: Applied Soft Computing, v. 12, p. 3472-3489.
15. Wang, K. P., L. Huang, C. G. Zhou, W. Pang, and Ieee, 2003, Particle swarm optimization for Traveling Salesman Problem: 2003 International Conference on Machine Learning and Cybernetics, Vols 1-5, Proceedings, p. 1583-1585.