Contents lists available at ScienceDirect

Int. J. Production Economics

journal homepage: www.elsevier.com/locate/ijpe

Particle swarm optimization: A study of particle displacement for solving continuous and combinatorial optimization problems

Sylverin Kemmoé Tchomté [a,b,*], Michel Gourgand [c]

[a] Laboratoire d'Informatique (LIMOS, UMR CNRS 6158), Campus des Cézeaux, 63177 Aubière, France
[b] IUP, Université d'Auvergne, 26 Av. Léon Blum, 63008 Clermont Ferrand, Cedex, France
[c] ISIMA, Université Blaise Pascal, Campus des Cézeaux, 63177 Aubière, Cedex, France

ARTICLE INFO

Article history:
Received 17 December 2007
Accepted 24 March 2008
Available online 3 May 2009

Keywords:
Particle swarm optimization
Intensification
Diversification
Particle displacement

ABSTRACT

This article addresses the particle swarm optimization (PSO) method. It is a recent proposed algorithm by Kennedy and Eberhart [1995. Particle swarm optimization. In: Proceedings of the IEEE International Conference on Neural Networks (Perth, Australia), vol. IV, IEEE Service Center, Piscataway, NJ, pp. 1942–1948]. This optimization method is motivated by social behaviour of organisms such as bird flocking and fish schooling. PSO algorithm is not only a tool for optimization, but also a tool for representing socio-cognition of human and artificial agents, based on principles of social behaviour. Some scientists suggest that knowledge is optimized by social interaction and thinking is not only private but also interpersonal. PSO as an optimization tool, provides a population-based search procedure in which individuals called particles change their position (state) with time. In a PSO system, particles fly in a multidimensional search space. During flight, each particle adjusts its position according to its own experience, and according to the experience of neighbours, making use of the best position encountered by itself and its neighbours. In this paper, we propose firstly, an extension of the PSO system that integrates a new displacement of the particles (the balance between the intensification process and the diversification process) and we highlight a relation between the coefficients of update of each dimension velocity between the classical PSO algorithm and the extension. Secondly, we propose an adaptation of this extension of PSO algorithm to solve combinatorial optimization problem with precedence constraints in general and resource-constrained project scheduling problem in particular. The numerical experiments are done on the main continuous functions and on the resource-constrained project scheduling problem (RCPSP) instances provided by the psplib. The results obtained are encouraging and push us into accepting than both PSO algorithm and extensions proposed based on the new particles displacement are a promising direction for research.

1. Introduction

The particle swarm optimization (PSO) model is a new population based optimization strategy introduced by Kennedy and Eberhart (1995a). To find the optimal solution, each particle adjusts its flying according to its own flying experience and its companions flying experience. Shi and Eberhart (1998a) named the former the cognition part and the latter the social part. For the social part, Kennedy and Eberhart (1995b) developed the so-called gbest (the companions flying experience is obtained in the population) and lbest (the companions flying experience is obtained in the local neighborhood) models for the neighborhood structure of particles. This paper overviews a study, which includes a new displacement of the particles. This new displacement takes into account

* Corresponding author at: Laboratoire d'Informatique (LIMOS, UMR CNRS 6158), Campus des Cézeaux, 63177 Aubière, France.
E-mail addresses: kemmoe@isima.fr (S. Kemmoé Tchomté), gourgand@isima.fr (M. Gourgand).

0925-5273/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.ijpe.2008.03.015

more the process of intensification than the process of diversification. It also provides a formal proof that particles converge to a slongtable point with this new displacement. Applications on continuous nonlinear functions and on resource-constrained project scheduling problem (RCPSP) instances are done.

2. The basic PSO algorithm

2.1. Generalities

2.1.1. Principles

The basic PSO algorithm consists of a swarm of particles moving in an n-dimensional, real valued search space of possible problem solutions. For the search space, in general, a certain quality measure, the fitness, is defined making it possible for particles to compare different problem solutions. Every particle i at the time t has the following characteristics:

- $X_{i,t}, V_{i,t}$ are the position and the velocity vectors;
- $P_{i,t}$ is the small memory storing its own best position seen so far;
- $G_{i,t}$ is the global best position (this information flow is obtained by defining a neighborhood topology on the swarm).

At each time step t the velocity is updated and the particle is moved to a new position. This new position is simply calculated as the sum of the previous position and the new velocity:

$$X_{i,t+1} = X_{i,t} + V_{i,t+1} \tag{1}$$

The update of the velocity is determined with the following relation:

$$V_{i,t+1} = \underbrace{c_1.V_{i,t}}_{\text{momentum}} + \underbrace{c_2.r_2.(P_{i,t} - X_{i,t})}_{\text{local information}} + \underbrace{c_3.r_3.(G_{i,t} - X_{i,t})}_{\text{global information}} \tag{2}$$

where the parameter c_1 introduced by Shi and Eberhart (1998b) is the inertia weight, it controls the magnitude of the old velocity $V_{i,t}$. This parameter can be specific to each particle's component as specified in Chi-Yang and Szu-Wei (2008) and also can be decreasing suite in each iteration (Liu et al., 2006). The parameters c_2 and c_3 are real numbers chosen uniformly and at random in a given interval, usually $[0, 1]$. These values determine the significance of $P_{i,t}$ and $G_{i,t}$, respectively. In Wong and Leung (2008), $P_{i,t}$ and $G_{i,t}$ are two states, the current states are chosen according to some parameters values. The three velocities components are defined as follows:

- $c_1.V_{i,t}$ serves as a momentum term to prevent excessive oscillations in search direction;
- $c_2.r_2.(P_{i,t} - X_{i,t})$ refers to as the cognitive component, it represents the natural tendency of individuals to return to environments where they experienced their best performance;
- $c_3.r_3.(G_{i,t} - X_{i,t})$ refers to as the social component, it represents the tendency of individuals to follow the success of other individuals.

2.1.2. Neighborhood system

For the neighborhood system, there are many ways to define it. Kennedy (1999) distinguishes two classes: physical and social neighborhoods.

- Physical neighborhood takes distances into account. In practice, distances are recomputed at each time step, which is quite costly, but some clustering techniques need this information.
- Social neighborhood, which just takes relationships into account. In practice, for each particle, its neighborhood is defined as a list of particles at the very beginning, and does not change.

Note that, when the process converges, a social neighborhood becomes a physical one. Fig. 1 illustrates these neighborhood systems.

2.2. PSO algorithm

The steps of the PSO technique are given in the following pseudo-code (Algorithm 1).

Algorithm 1. The pseudo code of PSO algorithm.

```
Initialize PSO parameters coefficients, swarm size
Initialize positions, velocities of particles
Initialize Personal best position
Determine Best particle of the swarm
Loop
        Update velocities
        Update positions
        Determine personal best position
        Determine Best particle of the swarm
        Local search (optional)
Until criterion
```

The initial positions and velocities are determined randomly. The coefficients are chosen according to the previous mathematical study. The comparison between two particles is done with the fitness value. The stop criterion can be the maximum number of iteration or the computational time.

2.3. Mathematical study

The theory of linear, discrete-time dynamic systems provides a powerful set of tools and results for assessing

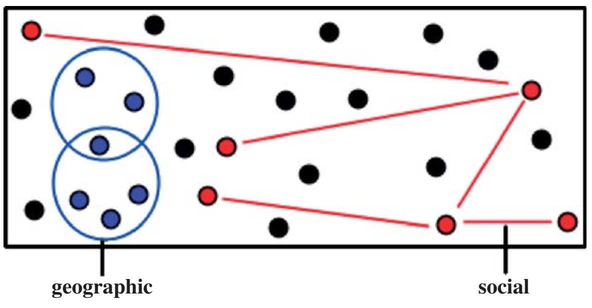


Fig. 1. Two main types of neighborhood system for PSO algorithm.

the dynamic behavior of a particle. The velocity and position updates are given by Eqs. (1) and (2). A simplification consists in considering the deterministic version of the PSO algorithm. In this version, the random numbers r_2 and r_3 are replaced by their mean values 1/2. We consider R the barycentre of the points $P_{i,t}$ and $G_{i,t}$, respectively weighted by c_2 and c_3. With these assumptions, the deterministic velocity update is written as follows:

$$V_{i,t+1} = c_1.V_{i,t} + (c_2 + c_3)/2.(R - X_{i,t}) \tag{3}$$

Let us consider $Y_{i,t} = R - X_{i,t}$ and $\phi = (c_2 + c_3)/2$. We can write

$$V_{i,t+1} = c_1.V_{i,t} + \phi.Y_{i,t} \tag{4}$$

And we know that $X_{i,t+1} = X_{i,t} + V_{i,t+1}$, we obtain

$$Y_{i,t+1} = c_1.V_{i,t} + (1 - \phi).Y_{i,t} \tag{5}$$

In matrix form, let us consider the vector $Q_{i,t} = \binom{V_{i,t}}{Y_{i,t}}$ and the matrix $A = \begin{pmatrix} c_1 & \phi \\ -c_1 & 1-\phi \end{pmatrix}$. The equations can be rewritten as follows:

$$Q_{i,t+1} = A.Q_{i,t} \tag{6}$$

Eq. (6) can be simplified as

$$Q_{i,t+1} = A^{t+1}.Q_{i,0} \tag{7}$$

In the context of the dynamic system theory, $Q_{i,t}$ is the particle state made up of its current position and velocity, A is the dynamic matrix whose properties determine the time behavior of the particle, R is the external input used to drive the particle towards a specified position. An equilibrium point is a state maintained by the dynamic system in the absence of external excitation (i.e., $R =$ constant). If one exists, any equilibrium point must satisfy the obvious condition $Q^{eq}_{i,t+1} = Q^{eq}_{i,t}$ for any i and t. For the particle in a deterministic swarm a straightforward calculation using $Q_{i,t+1} = A.Q_{i,t}$ gives

$$Q^{eq}_{i,t} = [0, 0]^T \Rightarrow V^{eq}_{i,t} = 0 \quad \text{and} \quad X^{eq}_{i,t} = R \tag{8}$$

This is intuitively reasonable. At equilibrium the considered particle must have zero velocity and must be positioned at the attraction point R. Standard results from dynamic system theory say that the time behavior of the particle depends on the eigenvalues of the matrix A. Eigenvalues 1 and 2 (either real or complex) are the solutions of the equation

$$P_A(\lambda) = 0 \tag{9}$$

where $P_A(\lambda)$ is the characteristic polynomial. It is obtained by this relation:

$$P_A(\lambda) = det(A - \lambda.I) \tag{10}$$

where the function *det* returns the determinant of the matrix $A - \lambda.I$ with I as the identical matrix. We have $P_A(\lambda) = 0 \Rightarrow \lambda^2 - (\phi - c_1 - 1) * \lambda + c_1 = 0$. The necessary and sufficient condition for the equilibrium point given by the equation to be slongtable (an attractor) is that both eigenvalues of the matrix A have magnitude less than 1. In this case the particle will eventually settle at the equilibrium and the PSO algorithm will converge. The discriminant of the quadratic equation is

$$\delta = (\phi - c_1 - 1)^2 - 4.c_1 = \phi.(\phi - 2.(c_1 + 1)) + (c_1 - 1)^2 \tag{11}$$

The analysis of the roots given by the discriminant leads to the following set of conditions:

$$\phi > 0, \quad \phi - 2 * (c_1 + 1) < 0 \quad \text{and} \quad c_1 < 1 \tag{12}$$

All the details of this study are given in Trelea (2003a). According to Eq. (8) there are various behaviors of the particles:

- In practice, one chooses the coefficients c_1 and ϕ so that Eq. (8) is verified. For any initial position and velocity, the particle will converge to its equilibrium position if and only if the algorithm parameters are selected according to Eq. (8). Before convergence, the particle exhibits harmonic oscillations around the equilibrium point when the eigenvalues of the matrix A are complex, this is possible if $\delta < 0$.
- The particle may also exhibit zigzagging behavior around the equilibrium point when at least one of the eigenvalues of the matrix A, whether real or complex, has negative real part. This is possible if $c_1 < 0$ or $c_1 - \phi + 1 > 0$. According to the choice of the coefficients and to Eq. (8), Trelea (2003a) shows that, for optimization, the choices which seem to give better results are oscillatory harmonic type, with a more or less fast convergence according to whether one wants to support the intensification or diversification (Trelea, 2003b).

What one would like ideally what for $\phi > 0$, the PSO algorithm converges. For that, the coefficient of constriction is introduced. Let us note by χ this coefficient and suppose that ϕ is the same constant for all the particles and all the dimensions. The velocity update is

$$V_{i,t+1} = \chi.(c_1.V_{i,t} + \phi.Y_{i,t}) \tag{13}$$

It can be shown that the algorithm converges if the values of χ is smaller than 1 for any $\phi > 0$. We obtain

$$\chi = \begin{cases} \dfrac{2\kappa}{\phi - 2 + \sqrt{\phi^2 - 4\phi}} & \phi \geqslant 4 \\ \kappa & \text{otherwise} \end{cases} \tag{14}$$

with $\kappa < 1$.

3. Proposal of a new particle's displacement

3.1. Ideas

The displacement of the particles in the search space is governed by the equations of updates velocity and position. As that appears in Eqs. (1) and (2), the particles fly around the positions $P_{i,t}$ and $G_{i,t}$ and we think not enough near these two good positions. PSO algorithm approach should be designed with the aim of effectively and efficiently exploring a search space. This search performed by PSO algorithm should be clever enough to both intensively explore areas of the search space with

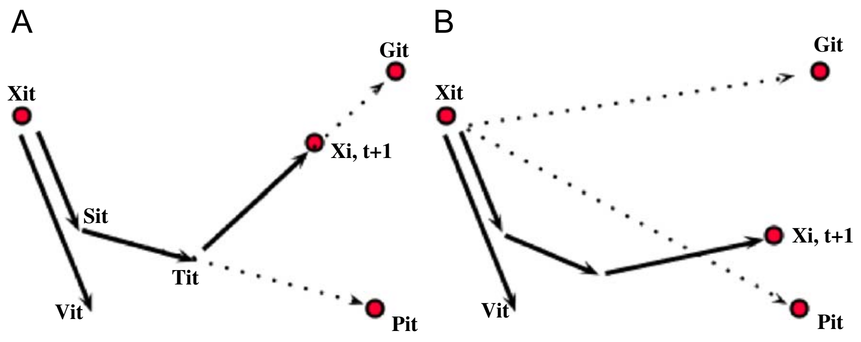


Fig. 2. Particle displacement in the swarm space.

high quality solutions, and to move to unexplored areas of the search space when necessary. The concepts for reaching these goals are called intensification (or exploitation) and diversification (or exploration). We suggest that for the classical PSO algorithm, the process of diversification is more taken into account than the process of intensification in the displacement of the particles. These concepts of intensification and diversification coming from Tabu search (Glover and Laguna, 1997) are two powerful forces driving optimization methods to high performance. In the PSO algorithm, the classical particle displacement does not benefit sufficiently the advantages of memory $P_{i,t}$ and $G_{i,t}$ in the search space, because the particles move in an area rather far away from the positions $P_{i,t}$ and $G_{i,t}$. We propose a new displacement of particles in which the strategy of intensification will be more taken into account. Other authors were interested in the compromise diversification–intensification while acting on the coefficients of the velocity update. Let us quote, for example (Shi and Eberhart, 1998b, 2000) which show thanks to several simulations that a great value of the coefficient c_1 facilitates a global diversification while a small value of the same coefficient ensures a better intensification of the areas located around the solutions of good quality.

3.2. *Mathematical formulation*

The intuition behind this new particle displacement is that, by letting information about two good solutions $(P_{i,t}, G_{i,t})$ spread out through the swarm, the particles will tend to move around these solutions in the search space. Let us note by $S_{i,t}$ and $T_{i,t}$ the intermediate positions of a particle i at the step t from the position $X_{i,t}$ to the position $X_i, t+1$. Fig. 2 illustrates on the left side (A) this new displacement and the classical one is given in the right side (B).

At each time step t the position is updated, the position is determined as follows in the new displacement:

$$S_{i,t} = X_{i,t} + c_1.V_{i,t} \quad \text{and} \quad T_{i,t} = S_{i,t} + c_2.r_2.(P_{i,t} - S_{i,t})$$

$$X_{i,t+1} = T_{i,t} + c_3.r_3.(G_{i,t} - T_{i,t})$$

$$X_{i,t+1} = X_{i,t} + c_1.(1-c_2.r_2).(1-c_3.r_3).V_{i,t} \\ + c_2.r_2.(1-c_3.r_3).(P_{i,t} - X_{i,t}) + c_3.r_3.(G_{i,t} - X_{i,t})$$

At each time step t the velocity is updated:

$$V_{i,t+1} = \alpha.V_{i,t} + \beta.(P_{i,t} - X_{i,t}) + \gamma.(G_{i,t} - X_{i,t}) \tag{15}$$

with

$$\alpha = c_1.(1 - r - 2.c_2).(1 - r_3.c_3), \quad \beta = c_2.(1 - r_3.c_3), \quad \gamma = c_3 \tag{16}$$

3.3. *Remarks*

The velocity update given by Eqs. (11) and (12) for this new displacement has the similar form than the classical case. We note that the three components of PSO weighted by coefficients are also present. The philosophy of the PSO system is preserved in the new displacement of the particles. The only change is a relation between classical coefficients and coefficients of the extension proposed. For the theoretical analysis of the PSO, the deterministic version will also be considered. In a similar way, the mathematic study carried out in the classical PSO system (Section 2.3) can be used in this section. Let us consider $\psi = (\beta + \gamma)/2$, and the velocity update for the new displacement of the particles is $V_{i,t+1} = \alpha.V_{i,t} + \psi.(R_1 - X_{i,t})$ with R_1 the barycenter of the points P_{it} and G_{it}, respectively, weighted by α and ψ. Taking into account the fact that the expression of the velocity update is the same structure in the two types of displacements, the conditions of convergence of PSO algorithm with the coefficients (α, ψ) can be determined in the similar way as classical PSO. We obtain

$$\psi > 0, \quad \psi - 2.(\alpha + 1) < 0 \quad \text{and} \quad \alpha < 1. \tag{17}$$

3.4. *Additional conditions of convergence*

Which are the additional conditions on the coefficients c_1, c_2 and c_3 satisfying (12) so that a particle converging in the classical PSO system also converges in the system based on the new displacement? To determine the additional conditions, we use a numerical simulation present with this following algorithm:

Algorithm 2. The pseudo code of PSO algorithm.

```
$c_1 = 0.1$
While $c_1 < 1$
  $c_2 = 0.1$
  While $c_2 < 1$
    $c_3 = 0.1$
    While $c_3 < 4$
      If $\phi - 2.(c_1 + 1) < 0$
        Determine $\phi, \phi - 2.(\alpha + 1), \alpha$
        If $\phi > 0, \phi - 2.(\alpha + 1) < 0, 0 < a < 1$
          Write $c_1, c_2, c_3$
        EndIf
      EndIf
      $c_3 = c_3 + 0.1$
    EndWhile
    $c_2 = c_2 + 0.1$
  EndWhile
  $c_1 = c_1 + 0.1$
EndWhile
```

We obtain two sets of values for the coefficients:

$$0 < c_1 < 0.9; \quad 0 < c_2 < 2 \quad \text{and} \quad 0 < c_3 < 2 \tag{18}$$

$$0 < c_1 < 0.9; \quad 2 <= c_2 < 4 \quad \text{and} \quad 2 <= c_3 < 4 \tag{19}$$

It is mean that the coefficients must be chosen according to Eqs. (12) and (18) or according to (12) and (19) to ensure the convergence with the new displacement of particles.

The PSO algorithm has been introduced for solving the nonlinear continuous functions. Clerc (2004) has provided a general framework to apply PSO algorithm to combinatorial optimization problem.

4. PSO algorithm for combinatorial optimization method

4.1. General framework

A general framework for the PSO algorithm for combinatorial optimization problems has been provided by Clerc (2004). To use PSO algorithm in the discrete case, the search space, the position, the velocity and operations were redefined:

- The search space of positions or states $\Omega = \{s_i\}$.
- A cost or objective function Φ on Ω, into a set of values $\Phi : \Omega = \{s_i\}$.
- $C = \{c_i\}$, whose minimums are on the solution states.
- An order on C, or, more generally, a semi-order, so that for every pair of elements of C, (c_i, c_j), we have either $c_i < c_j$ or $c_i >= c_j$.
- Particle position: A position of a particle can be considered as a sequence of activities or jobs. For travel salesman problem (TSP) a position is defined as an N-cycle: $s_i = (s_{i1}, s_{i2}, \ldots, s_{iN})$ because we are looking for Hamiltonian cycles. The search space is defined as the finite set of all N-cycles. For others combinatorial optimization problems, a sequence can be used as position.
- Particle velocity of particle. This operator when applied to a position during one time step gives another position. For TSP, it is a permutation of N elements called list of transpositions. The velocity is represented as $v = ((x_k, y_k)_{k=1,\ldots,|v|})$ where $|v|$ is the length of the list. The velocity with an empty list is represented by $v = \emptyset$.

Some operations were redefined:

- The opposite of a velocity v. The velocity result is the same list of transpositions as in v, but in reverse order. It is easy to verify that if we apply two times the operator opposite to a velocity, the velocity does not change.
- The addition between a position and a velocity. This operation permits to move a particle in the search space. For example, if we consider $s = (1, 2, 3, 4, 1)$ and $v = ((12)(34))$, then $s \bar{\oplus} v = s' = (2, 1, 4, 3, 2)$.
- The addition between two velocities. If we consider the velocities v_1 and v_2, the addition $v_1 \oplus v_2$ is the list of transpositions which contains first the ones of v_1, followed by the ones of v_2.
- The external multiplication of a velocity v by a scalar c:
  1. If $c = 0$ then $c \otimes v = \emptyset$.
  2. If $0 < c < 1$ then we *truncate* the list of transpositions of the velocity v to $E(c.\|v\|)$ where $E(x) <= x < E(x) + 1$.
  3. If $c >= 1$ then we *duplicate* the list of transpositions of the velocity v. Let $c = k + c'$ with k nonzero integer value and $0 <= c' < 1$, $c \otimes v = \underbrace{v \oplus v \oplus v \ldots v \oplus v}_{ktimes} \oplus c' \otimes v$.
- The subtraction between two positions. If s_1 and s_2 are two positions, $s_1 \ominus s_2$ defines a velocity obtained by a given algorithm.
- The distance between two positions s_1 and s_2 is the length of the velocity v obtained by calculating $s_1 \ominus s_2$ or $s_2 \ominus s_1$.

4.2. Adaptation for project scheduling problem

4.2.1. RCPSP presentation

The resource-constrained project scheduling problem represents a challenging research problem of great interest that has been widely studied over the past decades. The RCPSP can be stated as follows: a project consists of activities $0, \ldots, n+1$ where each activity has to be processed to complete the project. For the sake of simplicity, in general a unique source activity 0 and a sink activity $n+1$ are included in the project. These activities 0 and $n+1$ correspond to the "project start" and to the "project end", respectively. The activities are interrelated by two kinds of constraints. First, precedence constraints state that one activity j cannot start before all its immediate predecessors have been achieved. Second, resource constraints state that activity requires resources to be achieved. There are K renewable resource types. The availability of each resource type k is R_k. Each activity i requires r_{ik} units of resource k to be processed. The duration of one activity i is denoted p_i. Commonly, the structure of the project is depicted by a so-called activity-on-node (AON) network where the nodes and the arcs represent the activities and the precedence constraints, respectively. $G = (E, V)$ denotes the precedence graph. Because, the RCPSP is an extension of the job-shop, it is an NP-hard problem (see Blazewicz et al. (1983) and Brucker et al. (1999) for details on complexity). Several surveys are available including but not limited to the survey of Herroelen et al. (1998), Brucker et al. (2000), Kolisch and Padman (2001), Weglarz (1999) and Demeulemeester and Herroelen (2002). Note that Brucker et al. (1999) provided a classification scheme for the relationships between RCPSP and machine scheduling. Kolisch and Padman (2001) surveyed some heuristic methods for classes of project scheduling problems.

4.2.2. *Valid sequence*

The project scheduling problem has the precedence constrained between some couple of activities. And then the general framework proposed by Clerc (1999) doesn't work because if we permute two activities, we can obtain a sequence of activities for which the precedence relations are not satisfied. We propose to model a position in the search space, only with the sequence that respects the precedence relation. We call it the valid sequence. The operation of sum of a position with a velocity in the general framework is redefined to avoid nonvalid sequence.

4.2.3. *Clone particle detection*

Trying to avoid premature converge to local minima, all population algorithm based approaches try to avoid clones. Since exact clone detection can be time consuming, hashing techniques have been promoted, for example, by Cormen et al. (1990). We consider for our study that

$$mark(C) = \sum_i t_i^2 (moduloH) \tag{20}$$

with t_i is the starting time of activity i and where H is a great integer value.

4.2.4. *Local search*

An efficient local search is an iterative search process based which exploits the special structure of the problem as much as possible. We apply at each iteration a local search on the best particle obtained by using left shifts and right shifts activities operations.

4.2.5. *Sequence evaluation model*

As previously stressed, the makespan is obtained after have executed the activities in a strict order provided by the sequence. We obtain a fitness (or makespan) by using this following algorithm:

Algorithm 3. Sequence evaluation model

```
Input parameter s: valid sequence
Output parameter fitness
Variables: i, job, jobsuc, t, k , Start, End, Ress(T,K)
fitness = 0,   Ress(t,k) = RessDispo(k),   t = 1,T; k = 1,K
For   i = 1   to   N
  job :=s(i);   t :=EarliestDate(job);   Bool :=false;
  While Bool  = =  false
    Start :=t;
    End :=t + ProcessingTime(job) 1;
    If   ResourceAvailable(job,start,end) = = true
      Bool :=true;
    Else
      t :=t + 1;
    EndIf
  EndWhile
  StartTime(job) :=start - 1
  EndTime(job):=StartTime(job)+ProcessingTime(job)
  If   fitness < EndTime(job)
    fitness:=EndTime(job);
  EndIf
  ResourceConsumption(job, Ress)
  For jobsuc in Succ(job)
    EarliestDate(jobsuc) :=EndTime (job);
  EndFor
EndFor
```

The following parameters and procedures are used in the algorithm:

- RessDispo(k): gives the amount of resource k available.
- EarliestDate(job): gives the earliest starting time of job.
- ProcessingTime(job): gives the processing time of job.
- ResourceAvailable(job,start,end): returns true if resources are available or false otherwise.
- StartTime(job): contains the starting time of job.
- EndTme(job): contains the ending time of job.
- ResourceConsumption(job,Ress): update the amount of resources in the variable Ress.

5. Computational experiments

5.1. *Continuous functions*

5.1.1. *Details of implementation*

The proposed extension of PSO algorithm is tested, respectively, on five benchmark functions: Sphere, Rosenbrock, Rastrigin, Griewank and Schaffer. For each function, Table 1 summarizes the following parameters: the number of dimensions (n), the admissible range of the variable (x), the optimum, the goal values and the sketch in two dimensions.

For implementation, we use six sets of coefficients as summarized in Table 2. The sets of coefficients 4–6 are, respectively, obtained by applying Eq. (16) to coefficients sets 1–3.

5.1.2. *Results for continuous functions*

For each set of parameter, we compute, respectively, the average (Av.), the minimum (Min.), the maximum (Max.) iteration number obtained by this extension of PSO algorithm. We also determine the success rate (Succ.Rate) and the expected functions evaluations number (E.F.E.N.) equal to the product of the number of particles and average number of iterations divided by the success rate, which is the main algorithm performance criterion. Tables 3–5 give, respectively, the results obtained with the coefficient sets 1 and 4, coefficient sets 2 and 5, coefficients sets 3 and 6. The three groups of five lines of each table have been obtained, respectively, with a population size of 15, 30 and 60. The five lines of each group correspond to the five functions defined above.

Table 1
Optimization test functions.

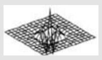
Formula $f_1(x)$, $f_2(x)$, $f_3(x)$, $f_4(x)$, $f_5(x)$	Range $[X_{min}, X_{max}]^n$	Optimal f	Goal for f	Sketch in 2D.
$\sum_{i=1}^{n} x_i^2$	$[-100, 100]^{30}$	0	0.01	
$\sum_{i=1}^{n-1}(100(x_{i+1}-x_i^2)^2-(x_i-1)^2)$	$[-30, 30]^{30}$	0	100	
$\sum_{i=1}^{n}(x_i^2-10cos(2*\pi*x_i)+10)$	$[-5.12, 5.12]^{30}$	0	100	
$(1/4000)\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}cos(x_i/\sqrt{i})+1$	$[-600, 600]^{30}$	0	0.1	
$0.5-(\sin(\sqrt{x_1^2+x_2^2}))^2-0.5/(1+0.001(x_1^2+x_2^2))^2$	$[-100, 100]^2$	0	10^{-5}	

Table 2
Coefficients sets tested.

Iter.	= 1000	Number of	Runs = 20	Number of particles = 15,30,60
Set1	$c_1 = 0.6$	$\phi = 1.7$		Proposed by Trelea (2003a)
Set2	$c_1 = 0.729$	$\phi = 1.494$		Proposed by Clerc (1999)
Set3	$c_1 = -0.5$	$\phi = 1.0$		Randomly generated
Set4	$\alpha = 0.014$	$\beta = 0.255$	$\gamma = 1.7$	Deduced from set1 and (8)
Set5	$\alpha = 0.047$	$\beta = 0.378$	$\gamma = 1.494$	Deduced from set2 and (8)
Set6	$\alpha = -0.125$	$\beta = 0.5$	$\gamma = 1.0$	Deduced from set3 and (8)

Fig. 3 illustrates the convergence of the PSO algorithm with the coefficient sets 1 and 4. The figure on the left side is obtained by doing the average of the number of iteration necessary to reach the goal of functions 1–4. For the coefficients set 1, the number of iterations is increasing, and for our proposed coefficients set 4, we obtain the goal of the function more quickly if we use high population size. The figure on the right side gives the results for function 5. The coefficients set 1 provided by Herroelen et al. (1998) gives the results better than our results obtained by set 4.

The line with a star $*$ are the best results reported by Shi and Eberhart (1998a), obtained with a population size of 30 particles, the coefficient set is set 2 and the velocity of particles limited to $[X_{min}, X_{max}]$.

Fig. 4 illustrates the convergence of the PSO algorithm with the coefficient sets 2 and 5.

Fig. 5 illustrates the convergence of the PSO algorithm with the coefficient sets 3 and 6 by using the results from Table 5.

Figs. 4 and 5 show that the coefficient sets 2 and 5, sets 3 and 6 have the opposite behavior. Our proposed coefficients sets 5 and 6 permit to reach more quickly to the optimum value.

5.1.3. *Analysis and discussion*

The rigorous form of velocity update for the PSO algorithm contains the random numbers; in this paper the theoretical analysis is done with the assumption of fixed value of the PSO coefficients. The results obtained by this assumption remain valid, and the proof is given by Trelea (2003a). It is showed in Trelea (2003a) that the presence of random numbers enhances the zigzagging tendency and slows down convergence, thus improving the state space exploration and preventing premature convergence to non-optimal points. This is especially true when the particle's own attraction point P is situated far from the population attraction point G. The equivalent attraction point R, is, in the case of the random algorithm, given by

$$R = \frac{r_2.c_2}{r_2.c_2 + r_3.c_3}P + \frac{r_3.c_3}{r_2.c_2 + r_3.c_3}G \tag{21}$$

If $P <> G$, it changes from iteration to iteration even if no better solutions are discovered, i.e., P and G remain constant. In the long run, however, it is expected that $P = G$ as all the particles in the population agree upon a single best point which becomes the unique attractor. In this case, (21) says that $R = P = G$ irrespective of the generated random numbers. The results obtained with the set of coefficients 4–6 are better than those obtained with the corresponding set of coefficients 1–3. For a fixed coefficient set, the corresponding one obtained by Eq. (16) according to these five functions tests give quick convergence. Our results are better than those reported by Shi and Eberhart (1998a), obtained with the classical PSO algorithm. We suggest that these good results can be explained by a better balance of the exploration of state space and the exploitation of the current optimum. The results of -1 obtained in the column E.F.E.N. means that

Table 3
Results obtained with coefficient sets 1 and 4.

Set 1					Set 4				
Av.	Min.	Max.	Succ. Rate	E.F.E.N.	Av.	Min.	Max.	Succ. Rate	E.F.E.N.
63.4	49	91	1.0	951	40.8	39	43	1.0	611
47.5	32	82	1.0	713	21.0	19	23	1.0	315
123.8	46	283	1.0	1857	19.9	17	24	1.0	297
51.0	29	112	1.0	765	31.0	22	78	1.0	465
28.7	9	51	1.0	430	477.4	2	1000	0.6	11015
71.4					28.2				
64.8	47	99	1.0	1942	37.8	35	39	1.0	1132
58.3	41	80	1.0	1750	18.9	17	20	1.0	566
289.3	43	830	1.0	8677	20.4	19	23	1.0	611
54.3	30	152	1.0	1627	28.8	21	91	1.0	862
19.4	2	35	1.0	581	303.6	3	1000	0.7	13013
116.7					26.5				
66.6	48	95	1.0	3995	35.0	33	36	1.0	2097
56.2	31	108	1.0	3372	18.1	16	20	1.0	1088
814.3	367	1000	0.4	122137	20.5	18	23	1.0	1227
74.3	27	241	1.0	4455	21.5	19	25	1.0	1290
11.3	4	24	1.0	675	188.1	2	1000	0.9	13277
252.8					23.7				

Table 4
Results obtained with coefficient sets 2 and 5.

Set 2					Set 5				
Av.	Min.	Max.	Succ. Rate	E.F.E.N.	Av.	Min.	Max.	Succ. Rate	E.F.E.N.
124.3	92	210	1.0	1865	26.6	26	28	1.0	399
155.4	59	353	1.0	2331	13.9	13	15	1.0	207
805.1	76	1000	0.3	40254	12.9	10	15	1.0	192
144.8	53	452	1.0	2172	16.6	15	21	1.0	249
41.7	11	93	1.0	624	21.5	2	60	0.1	322
254.3					18.3				
138.4	92	218	1.0	4153	24.9	23	26	1.0	745
530*	495*	573*	1.0*	15900*					
178	66	326	1.0	5340	13.1	12	15	1.0	393
669*	402*	1394*	1.0*	20070*					
1000	1000	1000	0.0	−1	12.6	11	14	1.0	379
213*	161*	336*	1.0*	6390*					
204.6	68	446	1.0	6139	18.5	14	44	1.0	553
313*	282*	366*	1.0*	9390*					
21.7	1	50	1.0	4455	21.5	19	25	1.0	1290
532*	94*	2046*	1.0*	15960*					
308.5					18.1				
149.7	96	244	1.0	8981	23.8	22	25	1.0	1427
318.7	49	660	1.0	19122	12.3	11	13	1.0	735
1000	1000	1000	0.0	−1	12.8	12	15	1.0	768
188.1	43	519	1.0	11288	16.4	14	23	1.0	983
19.2	7	31	1.0	1152	5.6	1	44	1.0	335
335.1					14.2				

Table 5
Results obtained with coefficient sets 3 and 6.

Set 3					Set 6				
Av.	Min.	Max.	Succ. Rate	E.F.E.N.	Av.	Min.	Max.	Succ. Rate	E.F.E.N.
104.9	73	142	1.0	1574	23.6	23	25	1.0	354
76.1	37	123	1.0	1141	12.9	12	14	1.0	193
140.7	73	244	1.0	2110	11.5	9	13	1.0	172
102.6	47	239	1.0	1538	15.6	14	18	1.0	233
28.5	7	43	1.0	428	9.4	2	22	1.0	141
90.6					14.6				
94.8	69	126	1.0	2842	22	21	23	1.0	660
91.7	44	180	1.0	2749	11.6	11	13	1.0	348
335.1	118	679	1.0	10053	11.6	10	13	1.0	348
125.5	50	362	1.0	3765	15.2	13	28	1.0	455
20.8	5	33	1.0	623	6.1	2	17	1.0	182
133.6					13.3				
108.1	76	182	1.0	6483	21.3	20	22	1.0	1277
101.7	45	166	1.0	6099	11.2	10	12	1.0	671
611.7	347	1000	0.9	40780	11.1	10	14	1.0	668
118	43	430	1.0	7080	14.9	12	40	1.0	893
16.5	6	29	1.0	987	4.1	2	8	1.0	245
191.2					12.5				

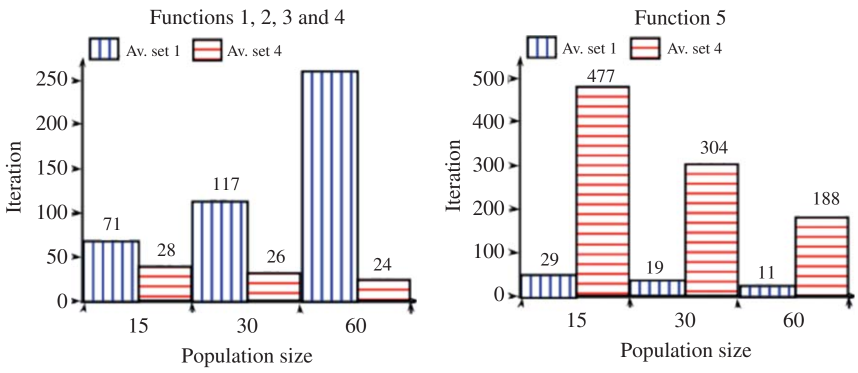


Fig. 3. Iteration versus population size with the coefficient sets 1 and 4.

there are no optimal solutions found during the 1000 iterations.

- Effects of the population size: For the sets of coefficient sets 1–3, increasing the number of particles does not give results envisaged. The iteration required oscillates and always does not decrease. The results obtained by the corresponding set of coefficient sets 4–6 show that the number of required algorithm iterations decreased. The average number of iteration to reach the optimum solution decreases when the number of particles increases.
- Effects of the coefficient a: Parameters sets 4–6 have a higher convergence rate than sets 1–3. For sets 4–6, the exploitation is thus favored compared to the exploration of the state space. The average number of algorithm iterations for sets 4–6 was generally smaller and the risk of premature convergence to non-optimal points was also smaller, as indicated in Tables 4 and 6 by the higher success rate (except for success rates of sets 1 and 4 which are comparable, Table 3).
- Effects of the objective function: In all cases, the global minimum (the only that achieved the goal) was situated at or near the center of the search domain. The Sphere and the Rosenbrock functions have a single minimum, while the other functions have multiple local minima (Table 1). Rastrigin and Griewank functions have a large scale curvature which guides the search toward the global minimum, while the Schaffer

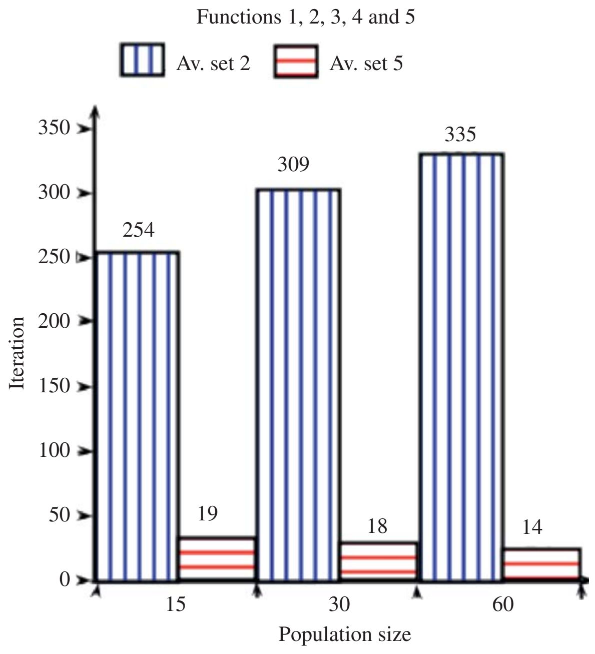


Fig. 4. Iteration versus population size with the coefficient sets 2 and 5.

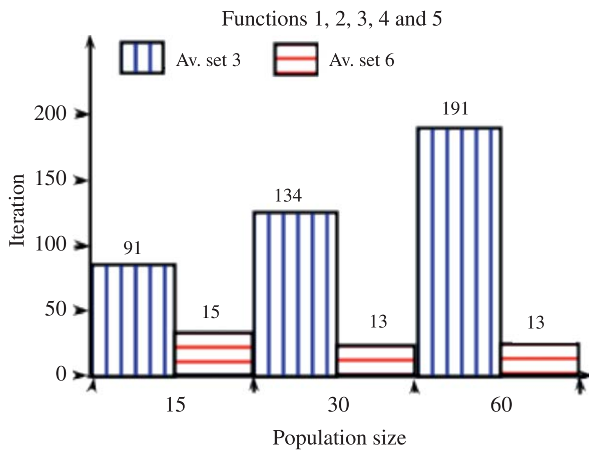


Fig. 5. Iteration versus population size with the coefficient sets 3 and 6.

function is essentially flat except near the global minimum. Therefore, it is not surprising that the largest success rates were achieved for the Sphere and Rosenbrock functions and the smallest one for the Schaffer function (cf Table 3 and coefficient set 4). Most of the times, failure to achieve the goal was due to a premature convergence to a local minimum.

5.2. Combinatorial optimization problem

5.2.1. Details of implementation

We have used the standard sets $j30$ and $j60$, the projects in each test set consist of 480 instances. Each instance has 10 runs. The number of generated and evaluated schedules is fixed to 1000 and 5000. According to the coefficients provided by Clerc (1999) ($c_1 = 0.729, c_2 = 1.494, c_3 = 1.494$) for continuous functions, we propose empirically to adapt these coefficients to RCPSP, we consider these coefficients: ($c_1 = 0.047$, $c_2 = 0.378, c_3 = 1.494$). According to Glover and Laguna (1997), the best swarm and neighborhood sizes are depending on the number of local minimums of the objective function. Usually, we simply do not have this information, except the fact that there are at most $N-1$ such minimums. In this study, we fix the swarm size at N particles where N is the number of dimension of the search space.

5.2.2. Results

Tables 6 and 7 present a comparison of PSO to state-of-the-art heuristic algorithms.

5.2.3. Analysis and discussion

Results of Table 6 indicate that PSO algorithm is capable of providing near optimal solutions, and it is comparable to others heuristics methods. The average CPU time of our PSO algorithm is 0.2 and 2.18 s for 1000 and 5000 evaluations, respectively, what are very competitive. Table 7 summarizes the average deviations from critical path based lower bound in $j60$. The critical path makespan lower bound is obtained by computing the length of a critical path in the resource relaxation of the

Table 6
Average deviation from optimal solution $-J = 30$.

Algorithm	Authors	Iter. 1000	Iter. 5000
PSO	Our proposal	0.26	0.21
PSO	Classical	0.31	0.29
GA, TS, path relink.	Kochetov and Stolyar (2003)	0.1	0.04
Hybrid GA	Valls et al. (2007)	0.27	0.06
GA	Alcaraz and Maroto (2001)	0.33	0.12
DJGA	Valls et al. (2005)	0.34	0.20
Sampling + BF/FB	Tormos and Lova (2003b)	0.25	0.13
Tabu search	Nonobe and Ibaraki (2002)	0.46	0.16
Sampling + BF	Tormos and Lova (2001)	0.30	0.16
Self-adapting GA	Hartmann (2002)	0.38	0.22
Activity list GA	Hartmann (1998)	0.54	0.25
Sampling + BF	Tormos and Lova (2003a)	0.3	0.17

Table 7
Average deviation from critical path $-J = 60$.

Algorithm	Authors	Iter. 1000	Iter. 5000
PSO	Our proposal	9.52	9.01
PSO	Classical	11.89	11.29
GA, TS, path relink.	Kochetov and Stolyar (2003)	11.71	11.17
Hybrid GA	Valls et al. (2007)	11.56	11.10
GA	Alcaraz and Maroto (2001)	12.57	11.86
DJGA	Valls et al. (2005)	12.21	11.27
Sampling + BF/FB	Tormos and Lova (2003b)	11.88	11.62
SA	Bouleimen and Lecocq (2003)	12.75	11.90
Sampling + BF/FB	Tormos and Lova (2003a)	11.88	11.62
Sampling + BF	Tormos and Lova (2001)	12.18	11.87

problem. The algorithm is capable of providing near optimal solutions. The average CPU time for PSO algorithm is 1.77 and 6.18 s for 1000 and 5000 evaluations, respectively.

6. Conclusion

We have proposed in this paper a new particle displacement. We suggest that this PSO extension balance better the process of exploitation and exploration. This extension consists in the redefinition of a concept as position (valid sequence) and some operations as the addition between a position and a velocity and the difference between two positions. With this new version, for a fixed set of coefficient we can obtain a set of coefficients that allows to accelerate convergence. The results on continuous functions are encouraging. We have also applied this new version to solve combinatorial optimization problems (RCPSP), the results obtained are comparable with those obtained with other approach methods. The computational results show that PSO algorithm is a fast and high quality algorithm. For the instance set *j*60, our algorithm outperforms all state-of-the-art algorithms for the RCPSP.

References

Alcaraz, J., Maroto, C., 2001. A robust genetic algorithm for resource allocation in project scheduling. Annals of Operations Research 102, 83–109.

Blazewicz, J., Lenstra, J.K., Rinooy Kan, A.H.G., 1983. Scheduling subject to resource constraints: classification and complexity. Discrete Applied Mathematics 5, 11–24.

Bouleimen, K., Lecocq, H., 2003. A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version. European Journal of Operational Research 149 (2), 268–281.

Brucker, P., Drexl, A., Mohring, R., Neumann, K., Pesch, E., 1999. Resource-constrained project scheduling: notation, classification, models, and methods. European Journal of Operational Research 1121, 3–41.

Brucker, P., Knust, S., Roper, D., Zinde, Y., 2000. Scheduling UET task system with concurrency on two parallel identical processors. Mathematical Methods of Operation Research 53, 369–387.

Chi-Yang, T., Szu-Wei, Y., 2008. A multiple objective particle swarm optimization approach for inventory classification. International Journal of Production Economics 114, 656–666.

Clerc, M., 1999. The swarm and the queen: towards a deterministic and adaptive particle swarm optimization. In: Proceeding of the IEEE Congress on Evolutionary Computation (1951–1957), IEEE Service Center, Washington, DC, Piscataway, NJ.

Clerc, M., 2004. Discrete particle swarm optimization. In: Onwubolu, G.C., Babu, B.V. (Eds.), New Optimization Techniques in Engineering. Springer, Berlin, pp. 204–219.

Cormen, T.H., Leiserson, C.L., Rivest, M.L., 1990. Introduction to Algorithms. MIT Press, Cambridge, MA.

Demeulemeester, E., Herroelen, W., 2002. Project Scheduling: A Research and Book International Series, Operations Research and Management Science. Kluwer Academic Publishers, Boston.

Glover, F., Laguna, M., 1997. Tabu Search. Kluwer Academic Publishers, Dordrecht.

Hartmann, S., 1998. A competitive genetic algorithm for resource-constrained project scheduling. Naval Research Logistics 45, 733–750.

Hartmann, S., 2002. A self-adapting genetic algorithm for project scheduling under resource constraints. Naval Research Logistics 49, 433–448.

Herroelen, W., De Reyck, B., Demeulemeester, E., 1998. Resource-constrained project scheduling: a survey of recent developments. Computers and Operations Research 25 (4), 279–302.

Liu, S., Tang, J., Song, J., 2006. Order-planning model and algorithm for manufacturing steel sheets. International Journal of Production Economics 100 (1), 30–43.

Kennedy, J., Eberhart, R., 1995. Particle swarm optimization. In: Proceedings of the IEEE International Conference on Neural Networks (Perth, Australia), vol. IV, IEEE Service Center, Piscataway, NJ, pp. 1942–1948.

Kennedy, J., Eberhart, R., 1995. A new optimizer using particle swarm theory. In: Proceedings of the Sixth International Symposium on Micromachine and Human Science, vol. 3, Nagoya, Japan, pp. 39–43.

Kennedy, J., 1999. Small worlds and megaminds: effects of neighborhood topology on particle swarm performance. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 1931–1938.

Kochetov, Y., Stolyar, A., 2003. Evolutionary local search with variable neighborhood for the resource constrained project scheduling problem. In: Proceedings of the Third International Workshop of Computer Science and Information Technologies, Russia.

Kolisch, R., Padman, R., 2001. A integrated survey of deterministic project scheduling. OMEGA 29, 249–272.

Nonobe, K., Ibaraki, T., 2002. Formulation and tabu search algorithm for the resource constrained project scheduling problem (RCPSP). In: Ribeiro, C.C., Hansen, P. (Eds.), Essays and Surveys in Metaheuristics. Kluwer Academic Publishers, Dordrecht, pp. 557–588.

Shi, Y., Eberhart, R., 1998. A modified particle swarm optimizer. In: Proceedings of the IEEE Congress on Evolutionary Computation, vol. 29, Piscataway, NJ, pp. 69–173.

Shi, Y., Eberhart, R., 1998. Parameter selection in particle swarm optimization. In: Annual Conference Evolutionary Programming, San Diego.

Shi, Y., Eberhart, R., 2000. Comparing inertia weights and constriction factors in particle swarm optimization. Congress on Evolutionary Computation, San Diego, CA, pp. 84–88.

Tormos, P., Lova, A., 2001. A competitive heuristic solution technique for resource-constrained project scheduling. Annals of Operations Research 102, 65–81.

Tormos, P., Lova, A., 2003a. An efficient multi-pass heuristic for project scheduling with constrained resources. International Journal of Production Research 41, 1071–1086.

Tormos, P., Lova, A., 2003. Integrating heuristics for resource constrained project scheduling: one step forward. Technical Report, Department of Statistics and Operations Research, Universidad Politecnica de Valencia.

Trelea, I.C., 2003a. The particle swarm optimization algorithm: convergence analysis and parameter selection. Information Processing Letters 85, 317–325.

Trelea, I.C., 2003. L'essaim particulaire vue comme un systèeme dynamique. Séminaire de L'optimisation par essaim particulaire, Paris Carré des sciences.

Valls, V., Ballestin, F., Quintanilla, S., 2005. Justification and RCPSP: a technique that pays. European Journal of Operational Research 165, 375–386.

Valls, V., Ballestin, F., Quintanilla, S., 2007. A hybrid genetic algorithm for the resource-constrained project scheduling problem. European Journal of Operational Research 185, 495–508.

Weglarz, J., 1999. Project Scheduling: Recent Models, Algorithms and Applications. Kluwer Academic Publishers, Dordrecht.

Wong, W.K., Leung, S.Y.S., 2008. Genetic optimization of fabric utilization in apparel manufacturing. International Journal of Production Economics 114, 376–387.