

A model for resource-constrained project scheduling using adaptive PSO

Neetesh Kumar · Deo Prakash Vidyarthi

Published online: 5 February 2015
© Springer-Verlag Berlin Heidelberg 2015

Abstract Resource-constrained project scheduling problem (RCPSP) is an important, but computationally hard problem. Particle swarm optimization (PSO) is a well-known and highly used meta-heuristics to solve such problems. In this work, a simple, effective and improved version of PSO i.e. adaptive-PSO (A-PSO) is proposed to solve the RCPSP. Conventional canonical PSO is improved at two points; during the particle's position and velocity updation, due to dependent activities in RCPSP, a high possibility arises for the particle to become invalid. To overcome this, an important operator named valid particle generator (VPG) is proposed and embedded into the PSO which converts an invalid particle into a valid particle effectively with the knowledge of the in-degree and out-degree of the activities depicted by the directed acyclic graph. Second, inertia weight (ω) that plays a significant role in the quick convergence of the PSO is adaptively tuned by considering the effects of fitness value, previous value of ω and iteration counter. Performance of the model is evaluated on the standard benchmark data of the RCPSP problem. Results show the effectiveness of the proposed model in comparison to other existing state of the art model that uses heuristics/meta-heuristics. The proposed model has the potential to be applied to other similar problems.

Keywords Particle swarm optimization · NP-hard · Heuristics · Resource-constrained project scheduling problem · Makespan

1 Introduction

Resource-constrained project scheduling problem (RCPSP) is an important, but NP-hard problem in project planning research [Blazewicz et al. \(1983\)](#). Though, in the past, a number of methods using linear programming, heuristics and meta-heuristics have been proposed to solve the RCPSP, these methods do not often produce good results and take substantial amount of time to converge. In recent times, researchers have intensified their interest towards optimal solution for RCPSP using evolved heuristics and meta-heuristics ([Chen and Huang 2007](#); [Valls et al. 2005](#)) making this an active research area.

RCPSP contains a set of activities with deterministic execution time, precedence relation among activities, accumulative resources availability constraints and its consumption by the activities. The objectives in RCPSP is to find a feasible schedule with some quality characteristics such as optimal makespan and computation time, response time and throughput, such that all the constraints are satisfied.

In literature, many exact scheduling algorithms to solve the RCPSP problem that uses linear programming approach ([Pritsker et al. 1969](#); [Kaplan 1996](#); [Klein 2006](#); [Mingozzi et al. 1998](#); [Kone et al. 2011](#)) are available. Out of these, the algorithms proposed by [Brucker et al. \(1998\)](#) and [Mingozzi et al. \(1998\)](#) seem to be the most effective and comprehensive. The drawback of their algorithms is that it solves the problem of small instances only (60 activities) in a satisfactory manner. Their method succumbs to higher convergence time for large instances as the solution search space increases

Communicated by V. Loia.

N. Kumar · D. P. Vidyarthi (✉)
School of Computer and Systems Sciences, Jawaharlal Nehru
University, New Delhi 110067, India
e-mail: dpv@mail.jnu.ac.in

N. Kumar
e-mail: dgoldneetesh15@gmail.com

drastically, leaving the space for heuristic and meta-heuristic to solve the RCPSP problem in an efficient and satisfactory manner (Bean 1994). Researchers have proposed many scheduling algorithms based on the variants of heuristics and meta-heuristics that include branch and bound (BB) (Dorn-dorf et al. 2000; Brucker et al. 1998; Demeulemeester and Herroelen 1995; Heilmann 2003), Tabu search (TS) (Nonobe and Ibaraki 2002), genetic algorithm (GA) (Hartmann 1998, 2002; Alcaraz et al. 2003; Valls et al. 2005), simulated annealing (SA) (Bouleimen and Lecocq 2003), adaptive search (AS) (Schirmer 2000), Ant colony optimization (ACO) (Merkle et al. 2002; Herbots et al. 2004), artificial bee colony (ABC) (Shi et al. 2010; Akbari et al. 2011; Ziaratia et al. 2011; Qiong and Yoonho 2013), particle swarm optimization (PSO) (Chen et al. 2010; Kolisch and Hartmann 2006; Zhang et al. 2005, 2006; Jarboui et al. 2008; Lu et al. 2008; Qiong and Yoonho 2013) etc., to produce efficient and satisfactory results for RCPSP.

Particle swarm optimization, an emerging meta-heuristic, has been applied to solve a number of optimization problems because of its distinguishing characteristics (Bai 2010; Jones 2005; Singh et al. 2014) such as (1) simple and easy enumeration, (2) being free from derivation, (3) robustness to control parameters, (4) having limited number of parameters, (5) sensitivity towards the objective function and parameters, (6) less dependency on initial particles, (7) relatively quick convergence and (8) good quality solution. Apart from the mentioned benefits, PSO has been found to be extremely efficient for solving the RCPSP applications (Zhang et al. 2005).

Initially, Zhang et al. (2006) proposed PSO to solve the RCPSP problem with good individual conjunction of priority-based and permutation-based representation and observed that the permutation-based representation scheme performed better. In their proposal, a general framework of PSO was used and a direction to successive use of PSO for RCPSP was mentioned as future research work. Later, Jarboui et al. (2008) proposed a combinatorial PSO (CPSO) with multiple execution modes to solve the RCPSP problem. In their work, modes to each activity were assigned and local search optimization was used to better prioritize the sequence of associated activities. Results indicate that CPSO performs better than SA and was close to the PSO proposed by Zhang et al. (2006). Lu et al. (2008) analyzed the resource-constrained critical path and proposed a PSO-based approach in the generation of resource-constrained schedule for the shortest makespan. Chen et al. (2010) proposed an algorithm for solving RCPSP using delay local search (Zhang et al. 2005) and bidirectional scheduling in PSO. Conclusively, the above proposed methods are practically good to solve the RCPSP problem. Recently, Qiong and Yoonho (2013) proposed an improved PSO to solve the RCPSP using rank priority-based representation, double justification operator and move operator along with the greedy search. Results

indicate better performance of the model over other contemporary PSO approaches. The operators used in the above discussed model (e.g. double justification and move operator) are time consuming as these are applied in addition to the normal PSO procedures for the refinement of their results. Further, it uses an obsolete inertia weight selection procedure which requires extra preprocessing time.

Though the above discussed models use PSO and its variants, these often lack efficiency in terms of convergence and computational time. Nevertheless, it gives a direction for further exploration of PSO with introduction of newer and faster operators/parameters and integration of external knowledge to solve the RCPSP problem with improved quality of the solutions and faster convergence.

This paper proposes a PSO variant, an adaptive PSO (A-PSO) to solve the RCPSP problem. The proposed A-PSO is easy and simple (similar to standard PSO), but at the same time effective and faster in producing good results. It is observed that many a times particles become invalid due to the updation of velocity and position in PSO. An operator named valid particle generator (VPG) is proposed and embedded into PSO. By applying VPG, the invalid particles are converted into valid particles in an effective way. Further, an adaptive inertia weight tuner is proposed, which tunes the inertia weight by considering three particles' parameters: fitness value, previous inertia weight and iteration counter. These three parameters are highly responsible for the effective convergence of the PSO. Fitness of the solution is evaluated in terms of makespan by the assignment of the RCPSP activity to the best processing unit (core) at the moment. To test the effectiveness of the model, a number of experiments were designed on standard instance sets of J30, J60 and J120 from well-known project scheduling Problem Library (PSPLIB) (<http://www.om-db.wi.tum.de/psplib/data.html>). The result shows that the model works efficiently over 25, 17 and 23 existing state-of-the-art heuristics/meta-heuristics for the instance sets of J30, J60 and J120, respectively.

The outline of this paper is as follows. After an introduction in Sect. 1, the RCPSP problem is described in Sect. 2. Section 3 briefs the standard PSO, whereas the proposed model is fully described in Sect. 4. Experimental studies for the performance of the proposed model along with the comparative experimental results are given in Sect. 5. Finally, the work is concluded in Sect. 6.

2 The RCPSP problem

A classical RCPSP problem with specified availabilities of multiple renewable resources (Kolisch and Hartmann 2006, 1999) is as follows. A project includes a set of activities $V = 0, 1, 2, \dots, N, N + 1$, each with K renewable resource

types. The duration of an activity i in V is denoted by d_i , and R_k is the availability (depicted in time period) of each resource type $k \in K$ in each time period. An activity i requires r_{ik} units of k type of resources during each time period in its total duration. The activities 0 and $N + 1$ i.e. the beginning and the end of the project are dummy activities, where $d_0 = d_{N+1} = 0$ and $r_{0k} = r_{N+1k} = 0$. The parameters d_i, R_k and r_{ik} are assumed to have non-negative integer values. Activities are interconnected with two types of constraints: one is the precedence constraint of an activity which prevents the start of the execution of that activity while its parent activities parent_i are yet to finish, and the other is that sum of all the required resources of a resource type k at any time period that cannot exceed R_k . The objective of the classical RCPS problem is to find a feasible schedule with the given precedence constraint among the activities, so that the makespan is minimized (Schirmer 2000). Figure 1a shows an example of the classical RCPS problem with six ($N = 6$) activities to be scheduled on $K = 1$ renewable resource of $R_k = 6$ units. One of the feasible schedules for 1(a) is shown in Fig. 1b. t on X -axis and R on Y -axis of Fig. 1b represent the time and resource instance, respectively. The example has six maximum available instances of resource type R_1 .

Let us assume that the completion time of an activity i is denoted as C_i and the completion times for the schedule S , consisting of various activities, is denoted as $C_1; C_2; \dots; C_N$. The mathematical formulation for the classical RCPS is as follows (Alba and Chicano 2007; Christofides et al. 1987):

$$\text{Minimize } C_{N+1}, \tag{1}$$

$$\begin{aligned} \text{Subject to } & C_i \leq C_j - d_j \text{ for } i = 1, 2, \dots, N + 1; \\ & \text{and } j \in \text{parent}_i, \end{aligned} \tag{2}$$

$$\sum_{i \in A(t)} r_{i,k} \leq R_k, \quad k \in K; t \geq 0 \tag{3}$$

$$\text{and } C_i \geq 0, \quad i = 1, 2, \dots, N + 1. \tag{4}$$

Equation 1 shows the objective function which is makespan minimization of the schedule. Equation 2 represents the precedence relationship between activities and Eq. 3 depicts the resource limitation constraint. Finally, the constraint of the decision variables is described in Eq. 4.

3 Overview of PSO

Kennedy et al. (2001) proposed PSO as an optimization technique that mimics the behaviour of social creatures (particles) in food searching (Badawi and Shatnawi 2013; Tasgetiren et al. 2004). In this technique, all the particles search for the food in multidimensional search space based on their two

important characteristics: position (referred to as the suggested solution) and velocity (rate of change of particle position). If any particle finds a better path to the food location, it attracts other particles to follow its path. The optimal path is evaluated based on its fitness. All particles move slowly towards the obtained solution updating their personal best and the global best solution. At the end, all particles reach the same position following the most probable optimal path. The standard canonical PSO is shown below as mentioned in (Mendes et al. 2004; Sevkli et al. 2004).

Canonical PSO

```

{
  Initialize number of particles by generating random velocity and position vector
  For each particle do
  {
    Evaluate the fitness
    Update the personal best
  }
  Update global best
  While termination conditions are not met do
  {
    For each particle do
      Update velocity and position
      Evaluate the fitness
      Update personal best if it is better than previous
    }
    Update global best if it is better than previous
  }
}
    
```

Canonical PSO starts with a set of particles with their corresponding random position and velocity vector and is referred to as initial population. To know the relevance of the particle towards the solution, a fitness function is used. Personal best and global best solutions are updated. Initially, each particle’s personal best is considered as its initial solution and the solution with optimum fitness value of all the particles is considered as global best. With each iteration, the position and the velocity vector of each particle is updated using some rules by evaluating the fitness of each particle. The particle with the best fitness value is compared with the global best solution; if it is better, the global best is updated. This procedure is repeated until the predefined termination condition is satisfied.

4 The proposed A-PSO for RCPS

This section presents the detailed description of the proposed A-PSO to solve the RCPS problem.

4.1 The proposed model

The model is initiated by random initial population of the particles. The random particles are in the form of random

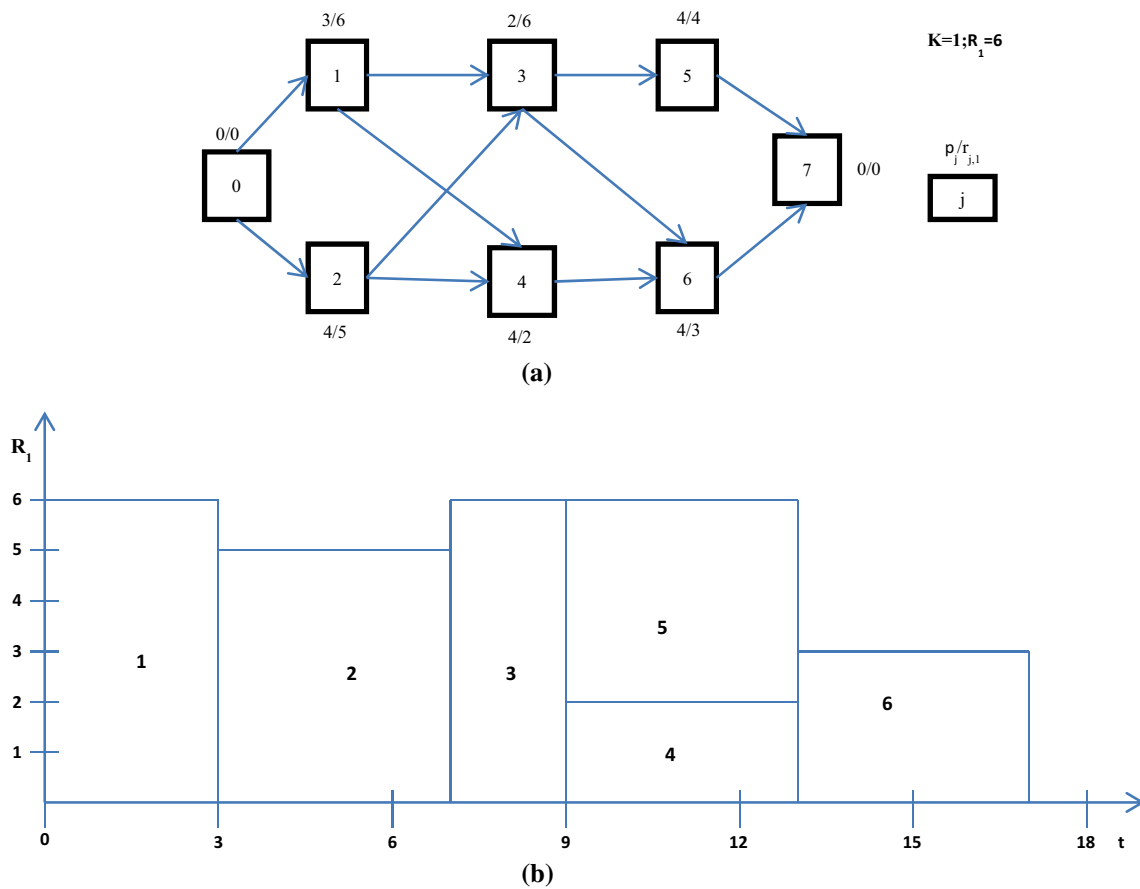


Fig. 1 a A project example. b A feasible schedule corresponding to the problem 1(a)

position and velocity values which are evaluated using the fitness function. Now, the termination conditions are verified; if the termination criteria are met, then exit; otherwise, the inertia weight is adaptively tuned to update velocities and positions of each particle. Smallest position value (SPV) rule (sec. 4.3) is applied to convert a continuous value vector into a discrete vector. Some of the produced discrete sequence vectors, corresponding to the particles, may be an invalid

sequence. A VPG operator is applied to convert them into a valid sequence. Now, the fitness of each particle is evaluated and based on the fitness value $pbest$ and $gbest$ solutions are initialized. The same procedure is repeated until a termination criterion is met. The flowchart of the proposed model is given in Fig. 2.

The pseudo-code of the proposed A-PSO model for RCPSP is as follows.

Procedure A-PSO

{ K=0 // iteration counter

Randomly initialize the position $X_i^d(k)$ and velocity $V_i^d(k)$ of i^{th} particle in d^{th} dimension for k^{th} iteration

$$S_i(k) = \text{SPV}(X_i(k)) \quad // \text{Apply SPV rule}$$

$[S'_i(k), X'_i(k), V'_i(k)] = \text{VPG}(S_i(k), X_i(k), V_i(k))$ // Apply VPG operator

$$F(S'_i(k)) = \text{Fitness}(S'_i(k), P, R_A, R_{Rd}) \quad // \text{Evaluate the fitness}$$

for all i particles in swarm do

$$pbest_i(k) = X'_i(k)$$

End for

$gbest = pbest_i(k)$ for all particles $j, j \neq i$ and $F(S'_i(k)) < F(S'_j(k))$

While termination conditions not met **do**

$$k = k + 1$$

Adaptive tuning of inertia weight (ω)

for all i particles in swarm do

$$V_i^d(k) = \omega \times V_i^d(k) + c_1 \times r_1 \times (pbest_i^d(k) - X_i^d(k)) + c_2 \times r_2 \times (gbest^d - X_i^d(k))$$

// update the velocity

$$X_i^d(k) = X_i^d(k) + V_i^d(k) \quad // \text{Update the position}$$

$$S_i(k) = \text{SPV}(X_i(k)) \quad // \text{Apply SPV rule}$$

$[S'_i(k), X'_i(k), V'_i(k)] = \text{VPG}(S_i(k), X_i(k), V_i(k))$ // Apply VPG operator

$$F(S'_i(k)) = \text{Fitness}(S'_i(k), P, R_A, R_{Rd}) \quad // \text{Evaluate the fitness}$$

If $F(S'_i(k)) < F(S'_i(k-1))$ **then**

$$pbest_i(k) = X'_i(k)$$

End if

End for

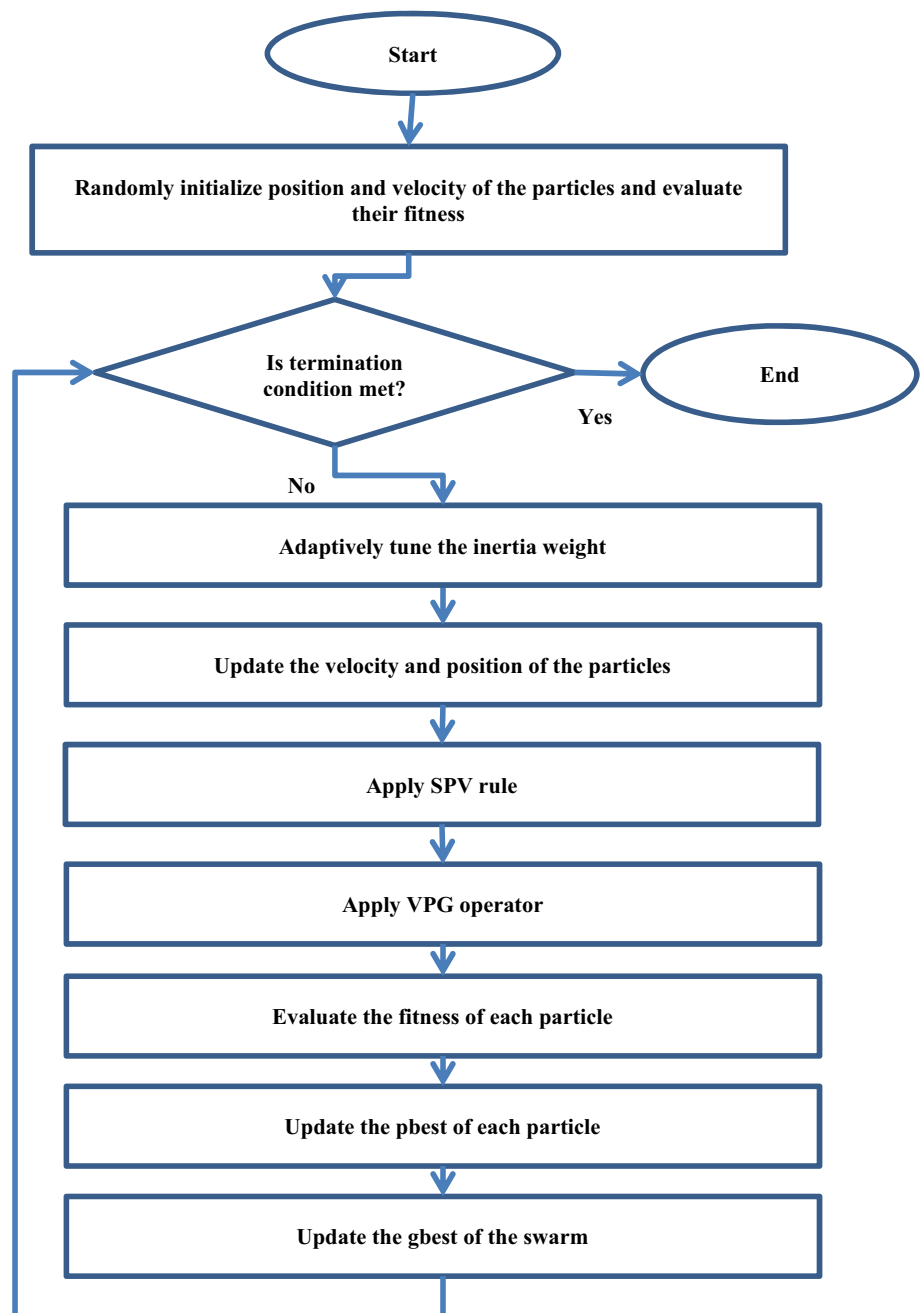
$gbest = pbest_i(k)$ for all particles $j, j \neq i$ and $F(S'_i(k)) < F(S'_j(k))$

End while }

The proposed A-PSO begins with the predefined number of random particles where $X_i^d(k)$ and $V_i^d(k)$ are the position and velocity of the i^{th} particle at the k^{th} iteration in the d^{th} dimension. To convert a continuous value vector of PSO particles into a discrete value vector, the SPV rule is applied to the $X_i(k)$ and a sequence vector $S_i(k)$ is produced.

The sequence vector may have an invalid sequence; to verify and convert it into a valid sequence, a valid particle generator, i.e. $\text{VPG}(S_i(k), X_i(k), V_i(k))$, is applied which changes an invalid sequence $S_i(k)$ to a valid sequence $S'_i(k)$. This sequence is evaluated using a fitness function $F(S'_i(k))$. On the basis of the fitness values, the $pbest_i(0)$ and $gbest$ are

Fig. 2 The flowchart of the proposed model



initialized. This procedure is iterated until the termination condition is met. In each next iteration, the values corresponding to ω , $V_i^d(k)$ and $X_i^d(k)$ are updated using Eqs. 10, 11 and 12, respectively. Using the same procedure the values of $pbest_i(k)$ and $gbest$ are updated. Finally, $gbest$ is obtained.

4.2 Particle initialization

Initially, a predefined number of random particles are generated by assigning an initial position and velocity for each particle. For this, the following equations have been used.

Position vector \vec{X}_i^d depicts the position vector for the i th particle corresponding to the d th dimension at the 0th iteration (initially) and is generated by Eq. 5.

$$X_i^d(o) = X_{\min} + (X_{\max} - X_{\min}) * r, \quad (5)$$

where X_{\min} and X_{\max} have values 0.0 and 4.0, respectively, to make the procedure random and r takes uniform random values between 0 and 1.

The velocity vector \vec{V}_i^d is for the i th particle corresponding to the d th dimension at the 0th iteration (initially) and is generated by Eq. 6.

Table 1 Particle encoding representation

D	1	6	4	3	5	2	7
$X_i^d(0)$	2.12	0.54	1.56	2.42	0.94	0.34	3.3
$V_i^d(0)$	0.34	-0.84	-1.83	1.34	3.83	-3.2	0.92

$$V_i^d(0) = V_{\min} + (V_{\max} - V_{\min}) * r, \tag{6}$$

where V_{\min} and V_{\max} have values -4.0 and 4.0 , respectively, to make the procedure random and r takes uniform random value between 0 and 1 .

The input range of values for the experiment corresponding to the random position and velocity of the particles have been taken from [Badawi and Shatnawi \(2013\)](#). By taking the modulus of the maximum value after each update, the scope of the particle’s position and velocities are controlled.

The encoding of the i th particle with seven dimensions (D) (i.e. activities) is shown in [Table 1](#).

where an activity of the problem represents a value in the corresponding dimension. In [Table 1](#), values corresponding to the first row represent activities of the problem and the values in the second and third rows, i.e. $X_i^d(0)$ and $V_i^d(0)$ represent the random position and velocity of the i th particle at the 0 th iteration of the d th dimension, respectively.

4.3 Smallest position value (SPV)

Smallest position value (SPV) is a heuristic proposed by [Tasgetiren et al. \(2004\)](#) to convert a continuous value vector of PSO into a discrete value vector so that it can be applied to all sequencing problems. This concept is similar to the random key concept proposed by [Bean \(1994\)](#) for genetic algorithm. With this heuristic, a continuous position value vector of wandering particles is easily converted into discrete activity vector. Conclusively, this heuristic produces the discrete value sequence vector \vec{S} by sorting the particle’s continuous value position vector X'' in ascending order. The detailed description is given in [Badawi and Shatnawi \(2013\)](#), [Tasgetiren et al. \(2004\)](#), whereas the pseudo-code for SPV is given as follows.

SPV($X_i(k)$)

```
{
Sort  $X_i(k)$  in ascending order
Enumerate  $S_i(k)$  with discrete values where  $S_i(k) = \text{dimention}(X_i(k))$ 
}
```

Table 2 Particle encoding representation

D	1	6	4	3	5	2	7
$X_i^d(0)$	2.12	0.54	1.56	2.42	0.94	0.34	3.3
$V_i^d(0)$	0.34	-0.84	-1.83	1.34	3.83	-3.2	0.92
$S_i^d(0)$	6	2	5	3	1	4	7

A demonstration of the SPV rule is given in [Table 2](#).

In [Table 2](#), the values corresponding to $S_i^d(0)$ represent the ascending order of the activities of the i th particle at 0 th iteration in d th dimension corresponding to their position values $X_i^d(0)$.

4.4 The proposed valid particle generator

PSO is one of the best population-based optimization techniques that operates in multidimensional search space ([Tasgetiren et al. 2004](#)). The initial population in PSO is the collection of randomly generated particles ([Tasgetiren et al. 2004](#)) covering multidimensional search space. Particles are the set of activities along with their associated position and velocity values updated with each iteration. It works smoothly if all the activities of the optimization problem are independent. When the activities of the optimization problem are constrained or dependent in some manner (e.g. activities in RCPS), a possibility exists that updated particle may become invalid (violating precedence constraint). Normally, a huge amount of computation (in updating the position and velocity vector) and number of iteration are involved in dealing with an invalid particle. Further, in an optimization problem of the order of $1,000$ or higher activities, there is a possibility of huge computational energy wastage to the tune of many hours or days in reaching to a valid particle. Since RCPS is a precedence-constrained optimization problem, it suffers the same. To overcome this problem of PSO, a valid particle generator is proposed. This checks only those suspected activities of the particle that results in the creation of an invalid particle and converts it into a valid particle by swapping the suspected activities. In other words, VPG changes the direction of the particle if it is going in the wrong direction. The pseudo-code for VPG is as follows.

```

VP $G(S_i(k), X_i(k), V_i(k))$ 
{
  Calculate in-degree of  $i^{\text{th}}$  particle corresponding to all  $d \in D$  dimensions
   $\varphi_i^d(k) = \text{indegree}(S_i^d(k))$ 
   $\vartheta = \{ \}$  // set of verified activities

  For  $d=1$  to  $D$  do //  $D$  is the number of dimension in the particle
     $\lambda = \{ \alpha \in S_i(k) | \text{indegree}(\alpha) = 0 \ \& \ \alpha \notin \vartheta \}$  // Find all the activities of zero in-degree in the  $i^{\text{th}}$  particle

    If  $(S_i^d(k) \notin \lambda)$ 
       $y_1 = S_i^d(k)$ 
       $\tau = \{ \alpha \in \lambda | \text{outdegree}(\alpha) \}$  // calculate out-degree of all the activities belongs to  $\lambda$ 
      if  $((\alpha_1 == \alpha_2) \ \& \ (\alpha_1, \alpha_2 \in \tau))$ 
         $y_2 = \text{random}(\lambda)$ 
      else
         $y_2 = \max(\tau)$ 
      End if
      swap( $y_1, y_2$ )
      swap( $X_i^{y_1}(k), X_i^{y_2}(k)$ )
      swap( $V_i^{y_1}(k), V_i^{y_2}(k)$ )
       $\lambda = \lambda - y_2$ 
       $\vartheta = \vartheta \cup y_2$ 
       $\mathbb{S} = \{ \alpha \in S_i(k) | \alpha = \text{successor}(y_2) \}$ 

    For  $j=1$  to length( $\mathbb{S}$ ) do
      If  $(\mathbb{S}_j \in S_i(k))$ 
         $\varphi_i^j(k) = \varphi_i^j(k) - 1$ 
      End if
    End for
  End if
  End for

   $S'_i(k) = S_i(k)$ 
   $X'_i(k) = X_i(k)$ 
   $V'_i(k) = V_i(k)$ 
}

```

The sequence vector $S_i(k)$ along with its $X_i(k)$ and $V_i(k)$ passes through the VPG operator to verify its correctness and making it a valid particle. The VPG operator begins with the calculation of in-degree of each activity (using $indegree(\alpha)$) in the particle and stores in $\phi_i^d(k)$. The activities with the zero in-degree are contained in λ , which represents the activities ready to be scheduled without violating the dependency constraint (i.e. their parents have already been scheduled). Further, the out-degree of each activity (using $outdegree(\alpha)$) which belongs to λ is calculated and stored in τ . Now, each activity in the sequence vector $S_i^d(k)$ is checked for its belonging to λ . If it belongs, it does nothing. Otherwise, if the out-degree of all the activities in τ is different, an activity is selected with the maximum out-degree (a maximum out-degree activity is preferred to increase the degree of parallelism) and swap it with the activity. In case the out-degrees of the activities in τ are the same, a random activity is selected from λ and swapped with the activity. Random activity is selected using the $random()$ function and swapping is done using the $swap(y_1, y_2)$ function. All activities are verified in a similar manner and finally a valid sequence of activities along with the corresponding position and velocity, i.e. $S_i'(k), X_i'(k), V_i'(k)$, is returned.

4.5 Fitness function

Once a particle passes through the valid particle generator, it is confirmed to be a valid particle. This confirmation allows us to use a simple fitness function to evaluate the makespan of the schedule. Hence, the fitness function evaluates particles with input such as sequence vector (discrete sequence corresponding to continuous position values), number of cores, available resource instances and required resource instances of the respective activity represented as $S'(k), P, R_A$ and R_R , respectively. Fitness function, shown in Eq. 7, is the makespan on P homogeneous cores such that the resource constraint (Eq. 8) is satisfied.

$$F(S(k)) = \max_{i=1,2,\dots,D} \{ \max_{j=1,2,\dots,P} \{ FT(S_i(k), j) \} \} \quad \text{Such that } R_R \leq R_A \tag{7}$$

$$\begin{aligned} & \{r_1 \leq r_{1a}\}, \{r_2 \leq r_{2a}\}, \dots \\ & \{r_m \leq r_{ma}\}, \text{ where } (r_1, r_2 \dots r_m) \in R_R \text{ and} \\ & (r_{1a}, r_{2a} \dots r_{ma}) \in R_A, \end{aligned} \tag{8}$$

where $FT(\alpha, j)$ represents the finish time of activity α on the best suitable core $j \in P$ at the time and the resource constraint for each activity, i.e. $R_R \leq R_A$ is represented in Eq. 8.

4.6 Tuning of inertia weight (ω)

Before discussing the proposed adaptive tuning of ω , it is necessary to understand the impact of ω on the particle movement and its role in fast convergence of the PSO algorithm. Initially, all randomly generated particles are widely spread over the search space of the potential solution. It is less likely that particles, in the initial iteration itself, visit optimal search space. After a number of iterations, a possibility arises that at least some particles have visited the optimal search space orbit. At this point, attention is required for the tuning of ω as it plays a vital role in the convergence. For large ω values between 0.9 and 1, the algorithm works as a global search algorithm. With the decrease in ω value, it slightly moves towards the optimal/local search space. To handle ω , the following two rules may be considered.

1. During initial iterations of the algorithm, the value of ω should be kept large (between 0.9 and 1), so that particles can navigate the search space globally in multidimensional space.
2. As particles approach the optimal multidimensional search space orbit, the value of ω should be tuned in a way that it provides effective and optimal local search.

To achieve effective convergence in the A-PSO, ω is adaptively tuned. Three important factors are taken into account to tune the value of ω used in the A-PSO: first, a normalized fitness value of the swarm, as it provides a good direction to the particles towards the optimal solution; second, the iteration counter and finally the previous value of ω (ω^{K-1}) to keep track of the movement of the particles from their previous position. Normalized fitness value is calculated using Eq. 9.

Normalized Fitness Value (NFV)

$$= \frac{Fitness_{Current} - Fitness_{min}}{Fitness_{max} - Fitness_{min}}, \tag{9}$$

where $Fitness_{Current}$ represents the global best and $Fitness_{max}$ represents the maximum personal best of the particle at the current iteration. $Fitness_{min}$ is taken as optimal makespan or critical path lower bound value (as per availability).

ω is calculated using Eq. 10:

$$\omega^k = \omega_{max} - \left(\frac{\omega_{max} - \left(\frac{NFV + \omega^{K-1}}{2} \right)}{K_{max}} \right) \times K. \tag{10}$$

Generally, when the value of ω is high it works as global search, and when it slowly becomes low it turns in to local search. Further, it is observed that if the global best solution of the current iteration is better than the previous one, it indicates that the particles are moving in a good direction, i.e.

the value of ω should be low for the next iteration. On the other side, if the global best solution of the current iteration is poorer than the previous one, it indicates that the particles are not moving in a good direction, i.e. a global search is further required and the value of ω should be high relatively to the previous iteration value. Hence, the proposed tuner (Eq. 10) works by considering the above-mentioned facts with a good combination of ω^{K-1} , NfV and iteration counter. The effectiveness of the tuner is proved in Sect. 5.

4.7 Rules for velocity and position update

Two important rules for updating velocity and position of the particles are as follows:

Velocity vector updating rule The velocity at the k th iteration is updated using Eq. 11.

$$\vec{V}_k = \omega * \vec{V}_{k-1} + c_1 r_1 (\vec{pbest}_{k-1} - \vec{X}_{k-1}) + c_2 r_2 (\vec{gbest} - \vec{X}_{k-1}), \quad (11)$$

Where c_1 and c_2 are self-recognition and social constants, respectively, and r_1, r_2 are uniform random numbers between 0 and 1.

Position vector updating rule The position of the particle is updated at the k th iteration using Eq. 12.

$$\vec{X}_k = \vec{X}_{k-1} + \vec{V}_k. \quad (12)$$

4.8 Critical path (CP) length

CP length is the longest path from the source to the sink node (Badawi and Shatnawi 2013; Tasgetiren et al. 2004) as represented by Eq. 13. The motive of CP length is to provide a bound to the optimal solution (Badawi and Shatnawi 2013):

$$\text{CP length} = \sum W_j. \quad (13)$$

where W_j is the processing time of task j belonging to the critical path, $j \in N$ and N is the number of tasks in the directed acyclic graph (DAG). To parallelize this DAG, minimum M number of cores is required, which is obtained using Eq. 14.

$$M = \frac{\sum W_i}{\text{CP length}} \quad 1 \leq i \leq N. \quad (14)$$

According to Eqs. 13 and 14, CP length is equal to the optimal schedule if there is minimum M number of cores available and the communication cost is negligible.

4.9 Termination condition

Three termination conditions have been used in the proposed PSO model: first, the specified number of iterations passed;

second, the length of $gbest$ equal to CP length; third, when all the particles reach the same position (i.e. $pbest$ of all the particles become same).

5 Experimental analysis

The proposed A-PSO model is applied to the standard data sets of J30, J60 and J120 generated by a standard problem generator as given in Kolisch and Sprecher (Kolisch and Sprecher 1997). J30 and J60 have a set of 480 instances each with 30 and 60 activities, respectively, and J120 has a set of 600 instances each with 120 activities. These sets of instances are publicly available at the well-known PSPLIB (<http://www.om-db.wi.tum.de/psplib/data.html>) with known optimum or best-known solutions as obtained by various researchers over the years. The optimum solutions are known only for J30 instances and the upper bounds (current best value of the solution) and lower bounds are provided for J60 and J120 set of instances. In this experimental work, two lower bounds are considered best lower bound (LB_f) and critical path lower bound (LB_o) with resource relaxation in RCPSP, same as in Qiong and Yoonho (2013). For fair comparison of the proposed model with other state-of-the-art models, 1,000 and 5,000 schedules are generated. The solution quality of the model is measured by three performance parameters, i.e. average deviation (Avg_dev) (Eq. 15), optimal rate (Optimal) (Eq. 16) and average computational time (Hartmann 1998; Alcaraz et al. 2003; Hartmann 2002; Valls et al. 2005; Bouleimen and Lecocq 2003; Schirmer 2000; Chen et al. 2010; Merkle et al. 2002; Herbots et al. 2004; Shi et al. 2010; Akbari et al. 2011; Ziaratia et al. 2011; Qiong and Yoonho 2013; Zhang et al. 2005; Kolisch and Hartmann 2006; Mendes et al. 2004; Sevкли et al. 2004; Zhang et al. 2006). Avg_dev shows the deviation between the solution given by the proposed model and the best solution, Optimal accounts for the percentage of solutions reaching the optimal value, and the average computation time represents the average time taken by an instance while computing in its execution unit.

$$\text{Avg_dev} = \frac{\sum \text{instances} \left(\frac{\text{Obtained-Best}}{\text{Best}} \times 100\% \right)}{\text{instances}}, \quad (15)$$

$$\text{Optimal} = \frac{\text{Optimal instances}}{\text{instances}} \times 100\%. \quad (16)$$

5.1 Parameter setting

The model starts by random initialization of the particles in which the position vector of all the particles are initialized with random continuous values between 0 and 4 as given in

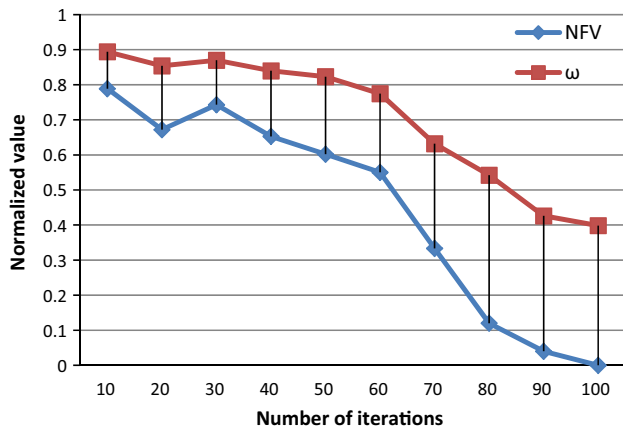


Fig. 3 Relation among NFV, ω and the iterations

Badawi and Shatnawi (2013) and Tasgetiren et al. (2004). The velocity vectors of all the particles are initialized by random continuous values between -4 and 4 as in Badawi and Shatnawi (2013) and Tasgetiren et al. (2004). The self-learning factor C_1 and social learning factor C_2 are set to 2 (suggested value 2.05 in “standard” PSO) (Bratton and Kennedy 2007). The inertia weight ω is adaptively tuned using Eq. 10 with its initial (maximum) value as 0.9 (good enough for global search in general). The iteration-wise tuning behaviour of values of ω , generated by the proposed tuner with respect to NFV, are shown in Fig. 3. It is observed that the tuned value of ω slightly increases when the value of NFV moves away

from the optimal value of the solution, which means it needs more global search. The tuned value of ω goes down when the number of iterations is increased, and when the value of NFV slightly moves towards the optimal value, it means that the particles move globally to local search. Further, to test the effectiveness of the adaptive tuner (in Eq. 10), a comparison with F-race approach (Birattari et al. 2010; Montes de Oca et al. 2011) is done. Totally, 16 treatments are done and for each treatment 48 instances out of 480 from J30 and J60, and 60 instances out of 600 from J120 are randomly selected. For each experiment, 1,000 schedules with 100 maximum numbers of iterations are run. The results corresponding to each treatment in the form of Avg_Dev._{opt} (average deviation from optimal makespan for J30) and Avg_Dev._{LB0} (average deviation from the lower bound of the critical path for J60 and J120) are shown in Table 3.

Figure 4 shows the comparative study of the average of average deviation resulting from the F-race approach and the proposed adaptive tuner of inertia weight (ω) corresponding to 16 treatments done in Table 3. From the results, it is easy to analyze that the proposed adaptive tuner performs effectively over the F-race approach. Hence, the adaptive tuner is used for further experimentation to perform effective global to local search.

In all the experiments, A-PSO is evaluated for 480 instances of J30 and J60, and 600 instances of J120, and the average deviation is calculated. The maximum number of iterations used is 100.

Table 3 Average deviation on training instances by F-race (Birattari et al. 2010) and the proposed adaptive tuner

Exp. No.	Average deviation (Avg_Dev)					
	J30 (Avg_Dev. _{opt})		J60(Avg_Dev. _{LB0})		J120(Avg_Dev. _{LB0})	
	F-race (ω)	Adaptive (ω)	F-race (ω)	Adaptive (ω)	F-race (ω)	Adaptive (ω)
1	0.75	0.4	3.44	3.63	8.76	8.68
2	0.93	0.84	4.28	4.08	8.32	7.93
3	0.58	0.45	3.98	3.22	7.89	8.48
4	0.62	0.34	4.20	3.76	8.45	7.86
5	0.22	0.56	3.87	3.27	8.49	7.38
6	0.31	0.44	3.82	3.53	8.24	7.88
7	0.58	0.18	4.48	4.24	8.39	7.95
8	0.42	0.94	3.48	3.65	8.95	8.06
9	1.35	0.55	3.39	3.37	8.95	8.46
10	0.66	0.23	3.76	3.67	8.45	7.98
11	0.63	0.42	3.46	3.98	8.36	8.58
12	0.25	0.36	4.38	3.55	7.98	7.90
13	1.23	1.15	4.22	3.86	8.47	8.57
14	0.73	0.28	3.74	3.24	8.54	8.58
15	0.3	0.38	4.38	3.74	8.93	8.12
16	0.23	0.24	3.46	3.86	8.64	7.78

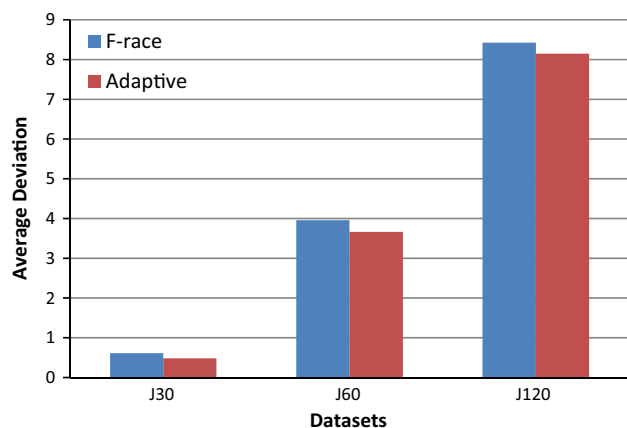


Fig. 4 The comparison of average of average deviation for training instances corresponding to Table 3

5.2 Experimental results

A comparative study of the proposed model with other heuristics, discussed in literature (Hartmann 1998; Alcaraz et al.

2003; Hartmann 2002; Valls et al. 2005; Bouleimen and Lecocq 2003; Schirmer 2000; Chen et al. 2010; Merkle et al. 2002; Herbots et al. 2004; Shi et al. 2010; Akbari et al. 2011; Ziaratia et al. 2011), is done on 1,000 and 5,000 randomly generated schedules. The compared heuristics/meta-heuristics are branch and bound (BB), Tabu search (TS), genetic algorithm (GA), simulated annealing (SA), adaptive search (AS), ant colony optimization (ACO), artificial bee colony (ABC), particle swarm optimization (PSO), etc.

Table 4 shows the percentage of average deviation and optimally solved instances with respect to optimal makespan for J30 instances. The model is evaluated for 1,000 and 5,000 number of schedules. The average deviation of the model is calculated as 0.28 and 0.06 % for 1,000 and 5,000 schedules, respectively, and the proposed model solved 88.92 and 97.48 % instances optimally for both the set of schedules, respectively. As obvious from Table 4, the model performs better over 25 other heuristics and meta-heuristics.

Tables 5 and 6 show the comparative results for the instances of J60 and J120 sets, respectively. For these

Table 4 Results of algorithms corresponding to the J30 case study (%)

Algorithm	References	Avg_Dev. _{opt}		Optimal	
		1,000	5,000	1,000	5,000
A-PSO	Present work	0.28	0.06	88.92	97.48
PSO	Qiong and Yoonho (2013)	0.49	–	–	–
PSO+	Chen et al. (2010)	0.54	–	–	–
PSO	Ziaratia et al. (2011)	0.69	0.42	–	–
PSO-Bidirectional	Chen et al. (2010)	0.84	–	–	–
PSO	Zhang et al. (2005)	0.98	–	–	–
PSO-Delay	Chen et al. (2010)	1.03	–	–	–
PSO	Chen et al. (2010)	1.33	–	–	–
GA	Alcaraz et al. (2003)	0.33	0.12	–	–
PABC	Qiong and Yoonho (2013)	0.34	0.17	86.60	91.74
ABC	Shi et al. (2010)	0.35	0.12	–	–
GA-DJ	Valls et al. (2005)	0.34	0.20	–	–
GA	Hartmann (2002)	0.38	0.22	–	–
SA	Bouleimen and Lecocq (2003)	0.38	0.23	–	–
BA-DJ	Ziaratia et al. (2011)	0.42	0.19	83.96	91.05
BSO-DJ	Ziaratia et al. (2011)	0.45	0.22	83.55	90.21
TS	Nonobe and Ibaraki (2002)	0.46	0.16	–	–
ABC-DJ	Ziaratia et al. (2011)	0.47	0.28	82.50	90.00
GA	Hartmann (1998)	0.54	0.25	81.50	–
BA	Ziaratia et al. (2011)	0.63	0.33	78.54	86.25
BSO	Ziaratia et al. (2011)	0.65	0.36	77.30	85.63
AS	Schirmer (2000)	0.65	0.44	–	–
ABC	Ziaratia et al. (2011)	0.98	0.57	72.71	83.84
GA	Hartmann (1998)	1.38	1.22	70.60	–
AS	Kolisch and Hartmann (2006)	1.44	–	–	–
ACO	Chen et al. (2010)	1.57	–	–	–

Table 5 Results of algorithms corresponding to the J60 case study (%)

Algorithm	References	Avg_Dev.LB _f		Avg_Dev.LB ₀		Optimal	
		1,000	5,000	1,000	5,000	1,000	50,000
A-PSO	Present work	3.02	2.76	11.94	11.12	73.02	75.58
PSO	Qiong and Yoonho (2013)	–	3.11	12.12	–	–	–
PSO+	Chen et al. (2010)	–	3.71	–	–	–	–
PSO-Bidirectional	Chen et al. (2010)	–	3.80	–	–	–	–
PSO	Chen et al. (2010)	–	4.23	–	–	–	–
PSO-Delay	Chen et al. (2010)	–	4.40	–	–	–	–
GA	Hartmann (2002)	3.26	2.88	12.21	11.70	–	–
PABC	Qiong and Yoonho (2013)	3.35	3.02	12.35	11.96	72.50	74.03
B&B	Dorndorf et al. (2000)	3.40	–	12.50	–	–	76.20
GA	Alcaraz et al. (2003)	–	–	12.57	11.86	–	–
BSO-DJ	Ziaratia et al. (2011)	–	–	12.58	12.29	72.08	73.34
ABC-DJ	Ziaratia et al. (2011)	–	–	12.61	12.24	71.67	73.34
GA	Hartmann (1998)	4.16	–	12.68	11.89	–	–
ABC	Shi et al. (2010)	–	–	12.75	11.48	–	–
SA	Bouleimen and Lecocq (2003)	–	–	12.75	11.90	–	–
AS	Schirmer (2000)	–	–	12.94	12.58	–	–
TS	Nonobe and Ibaraki (2002)	–	–	12.97	12.18	–	–
GA	Hartmann (1998)	4.96	–	13.30	12.74	–	–
BA	Ziaratia et al. (2011)	–	–	13.35	12.83	66.25	68.34
BSO	Ziaratia et al. (2011)	–	–	13.67	12.70	64.34	70.63
ABC	Ziaratia et al. (2011)	–	–	14.57	13.12	61.88	67.09
AS	Kolisch and Hartmann (2006)	–	–	15.94	–	–	–

instances, the proposed model is compared on percentage average deviation from the best lower bound and critical path lower bound of makespan, because optimal solutions for J60 and J120 sets of instances are not available. Again, the model is evaluated for two sets of numbers of schedules, i.e. 1,000 and 5,000. The average deviations of the model with respect to the best lower bound and critical path lower bound are obtained as 11.94 and 11.12 % and the A-PSO solves 73.02 and 75.58 % optimal instances of the J60 set of instances for both the set of schedules, respectively. The average deviations on the J120 set of instances with respect to the best lower bound and critical path lower bound are obtained as 34.935 and 32.49 % by solving 30.75 and 32.96 % optimal instances for both the set of schedules, respectively. So, with the same number of schedules, it gives better results over 21 other heuristics/meta-heuristics on J60 set of instances and over 27 other heuristics/meta-heuristics on J120 set of instances, respectively.

Table 7 shows the average computational time and used CPU core's frequency by A-PSO to evaluate the experimental results for the respective instance sets. The average CPU time taken by A-PSO is very low, which is possible with the significant contribution of VPG and adaptive tuning of inertia weight.

Furthermore, Table 8 presents the comparative study with other heuristics for average computational time. The average commutation time, average deviation and the used clock cycles of cores are shown in the third, fourth and fifth columns of Table 8, respectively. The performance of the reported heuristics/meta-heuristics was obtained from survey and original papers (Kolisch and Hartmann 2006; Chen et al. 2010; Birattari et al. 2010). The performance of A-PSO, shown in Table 8, is limited to 5,000 evaluated schedules. From Table 8, it is easy to see that A-PSO has significant lower average computation time for J30, J60 and J120 over other heuristics along with economic and acceptable average deviation.

Analysing the computational results, it is observed that the proposed A-PSO algorithm gives much better results for the same number of RCPSP schedules in comparison to other algorithms/heuristics. The reasons for this are: the VPG operator which continuously changes invalid particles into valid particles effectively (saving huge amount of computation) and inertia weight tuner which controls particle's movement effectively from global to local (or vice versa) search in large multidimensional search space. In other words, the direction of the particles changes smoothly from the infeasible region towards the feasible region. Hence, in each iteration of

Table 6 Results of algorithms corresponding to the J120 case study (%)

Algorithm	References	Avg_Dev.LB _f		Avg_Dev.LB ₀		Optimal	
		1,000	5,000	1,000	5,000	1,000	50,000
A-PSO	Present work	8.02	7.24	34.93	32.49	30.75	32.96
PSO	Qiong and Yoonho (2013)	–	8.40	37.22	–	–	–
PSO+	Chen et al. (2010)	–	9.38	–	–	–	–
PSO-Bidirectional	Chen et al. (2010)	–	9.38	–	–	–	–
PSO-Delay	Chen et al. (2010)	–	11.39	–	–	–	–
PSO	Chen et al. (2010)	–	12.17	–	–	–	–
GA-DJ	Valls et al. (2005)	–	–	35.39	33.24	–	–
PABC	Qiong and Yoonho (2013)	8.18	7.42	36.84	35.79	29.50	31.20
ABC	Shi et al. (2010)	–	–	36.29	34.18	–	–
GA	Hartmann (2002)	–	–	37.19	35.39	–	–
p-ACO	Herbots et al. (2004)	–	–	–	36.01	–	19.00
BA-DJ	Ziaratia et al. (2011)	–	–	37.72	36.76	29.84	31.17
BSO-DJ	Ziaratia et al. (2011)	–	–	37.84	36.51	29.17	30.84
ABC-DJ	Ziaratia et al. (2011)	–	–	37.85	36.82	29.34	30.34
GA	Alcaraz et al. (2003)	–	–	39.36	36.57	–	–
GA	Hartmann (1998)	–	–	39.37	36.74	–	–
B&B(a)	Dorndorf et al. (2000)	9.20	–	–	37.10	33.30	–
ACO	Herbots et al. (2004)	–	9.92	–	37.85	–	29.33
B&B(b)	Dorndorf et al. (2000)	9.50	–	–	38.0	32.20	–
AS	Schirmer (2000)	–	–	39.85	38.70	–	–
GA	Ziaratia et al. (2011)	–	–	39.93	38.49	–	–
BA	Ziaratia et al. (2011)	–	–	40.38	38.12	17.84	20.84
TS	Nonobe and Ibaraki (2002)	–	–	40.86	37.88	–	–
BSO	Ziaratia et al. (2011)	–	–	41.18	37.86	17.00	22.50
SA	Bouleimen and Lecocq (2003)	–	–	42.81	37.68	–	–
ACO	Merkle et al. (2002)	–	10.16	–	38.02	–	26.50
s-ACO	Merkle et al. (2002)	–	11.38	–	39.82	–	26.70
ABC	Bai (2010)	–	–	43.24	39.87	15.34	18.17

Table 7 A-PSO average computation time for various schedules of J3, J60 and J120

Instance set	Number of schedules	Average computation time (s)	Core
J30	1,000	0.031	Intel(R) Core(TM) i-5@ 3.20GHz, 4-gb RAM
J30	5,000	0.092	
J60	1,000	0.067	
J60	5,000	0.35	
J120	1,000	0.57	
J120	5,000	1.97	

the procedure, all the particles survive with effective fitness. Both VPG and adaptive tuning of inertia weight contribute substantially to the PSO to get a better and time-efficient solution.

6 Conclusion

This work proposes a variant of the PSO algorithm called adaptive PSO or A-PSO. A-PSO has been applied and tested for the RCPSP problem. A new operator, called a valid particle generator (VPG), has been proposed and inertia weight ω is adaptively tuned. The proposed VPG operator and ω play a significant role in the quick convergence of the PSO, saving a huge amount of computation. The proposed A-PSO offers low average deviation and its computation time for the experimentation is also low in comparison to other similar models. The VPG operator may prove to be important for many other variants of PSO to solve similar types of optimization problems.

The model has been simulated to study its performance on different sizes of RCPSP problem instances, e.g. J30, J60 and J120. A comparative study is done in terms of standard deviation and computation time. The results show that the model

Table 8 Comparative study of average computation time for J3, J60 and J120

Algorithm	References	Average computation time (s)	Avg_dev	Core
(a) $J = 30$				
Present work		0.092	0.06	3.2 GHz
ANGEL	Tseng and Chen (2006)	0.11	0.09	1 GHz
ACOSS	Chen et al. (2010)	0.13	0.06	1.86 GHz
PSO-HH	Koulinas et al. (2014)	1.15	0.04	3.0 GHz
Population-based	Valls et al. (2004)	1.16	0.10	400 MHz
Local search-critical	Valls et al. (2003)	1.61	0.6	400 MHz
Decompos. and local opt.	Palpant et al. (2004)	10.26	0.00	2.3 GHz
(b) $J = 60$				
Present work		0.35	11.12	3.2 GHz
ACOSS	Chen et al. (2010)	0.72	10.98	1.86 GHz
ANGEL	Tseng and Chen (2006)	0.76	11.27	1 GHz
PSO-HH	Koulinas et al. (2014)	2.0	11.13	3.0 GHz
Local search-critical	Valls et al. (2003)	2.8	11.45	400 MHz
Tabu search	Artigues et al. (2003)	3.2	12.45	450 MHz
Population-based	Valls et al. (2004)	3.7	10.89	400 MHz
Decompos. and local opt.	Palpant et al. (2004)	38.81	10.8	2.3 GHz
(c) $J = 120$				
Present work		1.97	32.49	3.2 GHz
ACOSS	Chen et al. (2010)	3.8	32.48	1.86 GHz
ANGEL	Tseng and Chen (2006)	4.79	34.49	1 GHz
PSO-HH	Koulinas et al. (2014)	7.99	32.59	3.0 GHz
Local search-critical	Valls et al. (2003)	17	34.53	400 MHz
Tabu search	Artigues et al. (2003)	67	36.16	450 MHz
Population-based	Valls et al. (2004)	59.4	31.58	400 MHz
Decompos. and local opt.	Palpant et al. (2004)	207.9	32.41	2.3 GHz

Bold results show the best result obtained by proposed work

offers competitive results, as the average deviations are 0.06, 11.12 and 32.49 % for J30, J60 and J120, respectively, and it takes 0.091, 0.32 and 1.9 % s average computation time for the schedule of J30, J60 and J120, respectively. Furthermore, the model performs better over 25, 21 and 27 other existing heuristics/meta-heuristics for a set of instances corresponding to J30, J60 and J120, respectively. The advantage of the model is faster convergence in comparison to other models for the same purpose. The proposed model can be used for solving other combinatorial optimization problems efficiently.

References

- Akbari R, Zeighami V, Ziarati K (2011) Artificial bee colony for resource constrained project scheduling problem. *Int J Ind Eng Comput* 2:45–60
- Al Badawi A, Shatnawi A (2013) Static scheduling of directed acyclic data flow graphs onto multicores using particle swarm optimization. *Comput Oper Res* 40:2322–2328
- Alba E, Chicano JF (2007) Software project management with GAs. *Inf Sci* 177:2380–2401
- Alcaraz J, Maroto C, Ruiz R (2003) Solving the multi-mode resource-constrained project scheduling problem with genetic algorithms. *J Oper Res Soc* 54(6):614–626
- Artigues C, Michelon P, Reusser S (2003) Insertion techniques for static and dynamic resource-constrained project scheduling. *Eur J Oper Res* 149:249–267
- Bai Q (2010) Analysis of particle swarm optimization algorithm. *Comput Inf Sci* 3:180–184
- Bean JC (1994) Genetic algorithms and random keys for sequencing and optimization. *ORSA J Comput (Springer)* 6(2):154–180
- Birattari M, Stützle T, Paquete L (2010) Varrentrapp K A racing algorithm for configuring metaheuristics. In: Langdon WB et al. (eds) GECCO 2002: Proceedings of the genetic and evolutionary computation conference. Morgan Kaufmann, San Francisco, vol 2, pp 11–18
- Blazewicz J, Lenstra JK (1983) Scheduling subject to resource constraints: classification and complexity. *Discret Appl Math* 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. *Eur J Oper Res* 149(2):268–281
- Bratton D, Kennedy J (2007) Defining a standard for particle swarm optimization. In: IEEE swarm intelligence symposium, SIS, vol 2007, pp 120–127
- Brucker P, Knust S, Schoo A (1998) A branch and bound algorithm for the resource- on strained project scheduling problem1. *Eur J Oper Res* 107(2):272–288

- Chen C-T, Huang S-F (2007) Applying fuzzy method for measuring criticality in project network. *Inf Sci* 177:2448–2458
- Chen RM, Wu CL, Wang CM, Lo ST (2010) Using novel particle swarm optimization scheme to solve resource-constrained scheduling problem in PSPLIB. *Expert Syst Appl* 37(3):1899–1910
- Chen W, Shi Y, Tenga H, Lan X (2010) An efficient hybrid algorithm for resource-constrained project scheduling. *Inf Sci* 180:1031–1039
- Christofides N, Alvarez-Valdes R, Tamarit JM (1987) Project scheduling with resource constraints: a branch and bound approach. *Eur J Oper Res* 29:262–273
- de Oca MAM, Aydın D, Stützle T (2011) An incremental particle swarm for large-scale continuous optimization problems: an example of tuning-in-the-loop (re)design of optimization algorithms. *Soft Computing* 15:2233–2255
- Demeulemeester E, Herroelen W (1995) New benchmark results for the resource-constrained project scheduling problem. In: Proceedings of the INFORMS Singapore international meeting, Singapore
- Dorndorf U, Pesch E, Huy TP (2000) A time-oriented branch-and-bound algorithm for resource constrained project scheduling with generalized precedence constraints. *Manag Sci* 46:1365–1384
- Hartmann S (1998) A competitive genetic algorithm for resource-constrained project scheduling. *Navig Res Logist* 45(7):733–750
- Hartmann S (2002) A self-adapting genetic algorithm for project scheduling under resource constraints. *Navig Res Logist* 49:433–448
- Heilmann RA (2003) Branch-and-bound procedure for the multimode resource-constrained project scheduling problem with minimum and maximum time lags. *Eur J Oper Res* 144(2):348–365
- Herbots J, Herroelen W, Leus R (2004) Experimental investigation of the applicability of ant colony optimization algorithms for project scheduling. Research Report, KU, Leuven
- Jarboui B, Damak N, Siarry P, Rebai A (2008) A combinatorial particle swarm optimization for solving multi-mode resource constrained project scheduling. *Appl Math Comput* 105(1):299–308
- Jia Qiong, Seo Yoonho (2013) Solving resource-constrained project scheduling problems: conceptual validation of FLP formulation and efficient permutation-based ABC computation. *Comput Oper Res* 40:2037–2050
- Jia Q, Seo Y (2013) An improved particle swarm optimization for the resource-constrained project scheduling problem. *Int J Adv Manuf Technol* 67:2627–2638
- Jones KO (2005) Comparison of genetic algorithm and particle swarm optimization. In: Proceedings in International Conference on Computer Systems and Technologies, pp 1–6
- Kaplan L (1996) Resource-constrained project scheduling with preemption of jobs. Dissertation, University of Michigan
- Kennedy J, Eberhart RC, Shi Y (2001) *Swarm intelligence*, 1st edn. Morgan Kaufmann, San Francisco
- Klein R (2000) *Scheduling of resource-constrained projects*. Kluwer, Boston
- Kolisch R, Sprecher A (1997) PSPLIB—a project scheduling problem library: OR Software—ORSEP Operations Research Software Exchange Program. *Eur J Oper Res* 96:205–216
- Kolisch R, Hartmann S (2006) Experimental investigation of heuristics for resource-constrained project scheduling: an update. *Eur J Oper Res* 174(1):23–37
- Kolisch R, Hartmann S (1999) *Heuristic Algorithms for the Resource-Constrained Project Scheduling Problem: Classification and Computational Analysis—Project Scheduling International Series in Operations Research & Management Science*, vol 14, pp 147–178
- Kone O, Artigues C, Lopez P, Mongeau M (2011) Event-based MILP models for resource-constrained project scheduling problems. *Comput Oper Res* 38(1):3–13
- Koulinas G, Kotsikas L, Anagnostopoulos K (2014) A Particle Swarm Optimization based Hyper-heuristic Algorithm for the Classic Resource Constrained Project Scheduling Problem, *Information Sciences*. doi:10.1016/j.ins.2014.02.155
- Lu M, Lam HC, Dai F (2008) Resource-constrained critical path analysis based on discrete event simulation and particle swarm optimization. *Autom Constr* 17(6):670–681
- Mendes R, Kennedy J, Neves J (2004) The fully informed particle swarm: simpler, maybe better. *IEEE Trans Evolut Comput* 8:204–210
- Merkle D, Middendorf M, Schmeck H (2002) Ant colony optimization for resource-constrained project scheduling. *IEEE Trans Evolut Comput* 6(4):333–346
- Mingozzi A, Maniezzo V, Ricciardelli S, Bianco L (1998) An exact algorithm for the resource-constrained project scheduling problem based on a new mathematical formulation. *Manag Sci* 44(5):714–729
- Nonobe K, Ibaraki T (2002) Formulation and tabu search algorithm for the resource constrained project scheduling problem. In: Ribeiro CC, Hansen P (eds) *Essays and surveys in meta-heuristics*. Kluwer Academic Publishers, Dordrecht, pp 557–588
- Palpanat M, Artigues C, Michelon P (2004) LSSPER: solving the resource-constrained project scheduling problem with large neighborhood search. *Ann Oper Res* 31:237–257
- Pritsker AAB, Watters LJ (1969) Wolfe PM Multi-project scheduling with limited resources: a zero-one programming approach. *Manag Sci* 16(1):93–108
- Schirmer A (2000) Case-based reasoning and improved adaptive search for project scheduling. *Navig Res Logist* 47:201–222
- Sevklı Z, Sevilgen FE (2004) Keleş, Particle swarm optimization for the orienteering problem. *Algorithmica* 35(1):30–39
- Shi YJ, Qu FZ, Chen W, Li B (2010) An artificial bee colony with random key for resource-constrained project scheduling. In: *International conference on life system modeling and simulation and international conference on intelligent computing for sustainable energy and environment (LSMS/ICSEE)*, WuXi, China, pp 148–157
- Singh Navjot, Arya Rinki (2014) A novel approach to combine features for salient object detection using constrained particle swarm optimization. *Pattern Recognit* 47:1731–1739
- Tasgetiren MF, Sevklı M, Liang YC (2004) Gencyilmaz G. Particle swarm optimization algorithm for single machine total weighted tardiness problem. In: *IEEE Congress on Evolutionary Computation* vol 2, pp 1412–1419
- Tseng L-Y, Chen S-C (2006) A hybrid metaheuristic for the resource-constrained project scheduling problem. *Eur J Oper Res* 175:707–721
- Valls V, Quintanilla S, Ballestín F (2003) Resource-constrained project scheduling: a critical activity reordering heuristic. *Eur J Oper Res* 149:282–301
- Valls V, Ballestín F, Quintanilla S (2004) A population-based approach to the resource-constrained project scheduling problem. *Ann Oper Res* 131:305–324
- Valls V, Ballestín F, Quintanilla MS (2005) Justification and RCPSP: a technique that pays. *Eur J Oper Res* 165(2):375–386
- Zhang H, Li XD, Li H, Huang FL (2005) Particle swarm optimization-based schemes for resource-constrained project scheduling. *Autom Constr* 14(3):393–404
- Zhang H, Li H, Tam CM (2006) Particle swarm optimization for resource-constrained project scheduling. *Int J Proj Manag* 24:83–92
- Ziaratia K, Akbaria R, Zeighami V (2011) On the performance of bee algorithms for resource-constrained project scheduling problem. *Appl Soft Comput* 11(4):3720–3733