Scientia Iranica E (2013) 20(6), 2123–2137

Sharif University of Technology
Scientia Iranica
Transactions E: Industrial Engineering
www.scientiairanica.com

Development of a method based on particle swarm optimization to solve resource constrained project scheduling problem

V. Zeighami[a], R. Akbari[b,*] and K. Ziarati[c]

a. *Department of Mathematics, Shiraz University, Shiraz, Iran.*
b. *Department of Computer Engineering and Information Technology, Shiraz University of Technology, Shiraz, Iran.*
c. *Department of Computer Science and Engineering and Information Technology, Shiraz University, Shiraz, Iran.*

Received 9 February 2012; received in revised form 10 December 2012; accepted 9 September 2013

KEYWORDS
Particle swarm optimization;
Termite colony optimization;
Resource constrained project scheduling problem.

Abstract. This work presents an efficient hybrid method based on Particle Swarm Optimization (PSO) and Termite Colony Optimization (TCO) for solving Resource Constrained Project Scheduling Problem (RCPSP). The search process of this hybrid method employs PSO iterations for global search and TCO iterations for local search. The proposed method works by interleaving the PSO and TCO search processes. The PSO method update schedules by considering the best solution found by the TCO approach. Next the TCO approach picks the solutions found by PSO search and perform local search around each solution. Each individual in TCO approach moves randomly but it is biased towards locally best observed solutions. Apart from hybridization, a new constraint handling approach is proposed to convert the infeasible solutions to the feasible ones. The standard benchmark problems of size j30, j60, j90, and j120 from PSPLIB are used to show the efficiency of the proposed method. The results showed that although PSO and TCO methods independently give good solutions, the hybrid of PSO and TCO gives better solution.

1. Introduction

The classical single-mode resource constrained project scheduling problem has been an active research area in recent years. In RCPSP problem, it is assumed that we have a project consisting of a set of activities with fixed durations. Arrangement of the activities is performed subject to the precedence and resource constraints. The resources are considered renewable. Upon these assumptions, the goal is to find a feasible schedule of the activities that minimizes the makespan of the project [1].

In a standard from, the RCPSP is described as follows.

Assume that we have a project which consists of a set of dummy and non-dummy activities $A = \{A_0, A_1, ..., A_n, A_{n+1}\}$ and K renewable resource type. The duration of an activity A_j is denoted as d_j. Each activity A_j requires r_{jk} units of resource R_k during each period of its duration. The dummy activities A_0 and A_{n+1}, respectively, are used as the beginning and end of the project, where $d_0 = d_{n+1} = 0$ and $r_{0k} = r_{n+1k} = 0$. Under these assumptions, the classical RCPSP tries to minimize duration of the project by generating a set of possible schedules [1]. In other

*. *Corresponding author.*
E-mail addresses: vahid.zeighami@gmail.com (V. Zeighami); akbari@sutech.ac.ir (R. Akbari) and ziarati@shirazu.ac.ir, (K. Ziarati)

words, the RCPSP tries to minimize the makespan of schedule F_{n+1} Subject to:

$$F_h \leq F_j - d_j, \qquad j = 1, ..., n+1; \quad h \in P_j, \tag{1}$$

where F_j is the finish time of activity, A_j and P_j is a set of preceding activities (or predecessors) of activity A_j. This equation enforces the precedence constraints between activities which can be given as:

$$\sum_{j \in V(t)} r_{jk} \leq R_k, \quad k \in K; \; t \geq 0, \tag{2}$$

where:

$$V(t) = \{j \in A | F_j - d_j \leq t < F_j\}. \tag{3}$$

This equation enforces the resource limitation constraint, and $F_j \geq 0, \quad j = 1, ..., n+1$ describes the constraints of decision variables. It is apparent from the problem formulation that RCPSP can be considered as a constraint satisfaction problem. Two types of constraints that should be satisfied are the "precedence of each activity" and the "sum of required resources" for each activity in any time period. The precedence constraint allows an activity A_j to start if and only if all its predecessor activities P_j have been completed. The "sum of required resources" constraint shows that the sum of resource requirements for resource k in any time period cannot exceed R_k .

The RCPSP has attracted extensive attentions in recent years and different types of methods have been proposed to cope with its' difficulty. The aim of this work is to design a hybrid method called PSTCO based on the PSO and TCO meta-heuristics. The PSO has shown efficiency in solving RCPSP problem. Due to the performance of PSO, different types of PSO-based approaches have been proposed in literature. Unlike PSO, the TCO meta-heuristic has not been used for resolving RCPSP problems. The PSO has rapid convergence speed and provide appropriate way for performing global search. However, in order to find better solutions in search spaces like RCPSPs we need some kinds of local searches. The TCO method is basically designed as a distributed decision making process that utilizes the potential of local searches around search space. The hybridization helps us to use benefits of two methods. The proposed method solves the RCPSP problem by interleaving PSO and TCO iterations. The interleaving process requires PSO and TCO pass their solutions to each other. The solutions of TCO are updated in PSO iterations by considering the global best solution. Similarly, the solutions of PSO are adjusted by considering the locally observed information. We found that better performance could be obtained when TCO and PSO work together on the RCPSP problem.

In this paper we show how to hybridize the PSO and TCO method and investigate its performance over the RCPSP problem. The rest of this paper is organized as follows. Section 2 presents an overview of recent developments to solve the RCPSP. In this section we specially focus on the previous PSO-based approaches. Section 3 describes the basic concepts of the PSO and TCO methods. Section 4 describes in detail the modified PSO method and its hybridization with TCO method. Section 5 is devoted to experimental study and analysis of the proposed hybrid method compared to the other state-of-art methods. Finally, Section 6 ends this work by presenting some concluding remarks.

2. Related works

In recent years many exact, heuristics, and meta-heuristic methods have been proposed by authors to solve the RCPSP problems. RCPSP is known as a NP-hard problem. Due to its NP-hardness, the exact methods can not solve large size RCPSP problems in a satisfactory manner. Hence, the majority of researches have concentrated on the heuristic and meta-heuristic approaches. A meta-heuristic approach has the ability to produce an optimal or near-optimal solution at each cycle.

Dynamic programming [2], branch and bound [3-6], and zero-one programming [7-9] are among the most competitive exact methods. These exact methods can only solve small problem instances. Several heuristics [10-13] have been proposed to tackle complexities of the RCPSPs. In recent years, many meta-heuristics have been proposed for solving RCPSP. Some of the meta-heuristic approaches such as tabu search [14] and simulated annealing [15] maintain only one solution at each cycle of the algorithm. These methods try to find a new solution with better quality from the current solution iteratively.

Other meta-heuristics containing methods such as Genetic Algorithm (GA) [16-19], Ant Colony Optimization (ACO) [20], Particle Swarm Optimization (PSO) [21-23], and Bee Algorithms (BA) [24-26] maintain a set of solutions at each cycle of the algorithm. These approaches solve the RCPSP by employing an initial population of individuals each of which representing a candidate schedule for the project. Then, they evolve the initial population by successively applying a set of operators on the old solutions to transform them into new solutions. These approaches showed efficiency in solving RCPSP problems.

PSO was first introduced by Kennedy and Eberhart in [27]. This method has been successfully applied on a wide range of engineering fields. The application of PSO for RCPSP was studied by Zhang et al. in [28] and [29]. They used permutation-based representation and

priority-based representation to represent a solution. In permutation-based representation, the location of an activity in the solution vector shows the order the activity scheduled. In priority-based representation, each location in the solution vector shows an activity and the corresponding value shows the priority of activity. Similarly, Tchomte et al. applied an improved PSO on the RCPSP problem [30]. Their study showed that PSO provides efficient way to solve RCPSP. It seems that performance of PSO over RCPSP may be improved through different approaches such as enhancing its exploration ability, providing appropriate balance between exploration and exploitation, balancing local and global search or increasing diversity of its populations. For example, the critical path method was used by Chen et al. to enhance exploration ability of PSO when solving RCPSP [31]. Also they used delay local search and bidirectional scheduling to further improve the performance of the standard PSO.

Apart from RCPSP, other scheduling problems may be solved successfully using PSO. The PSO has been used in [32] to solve flowshop scheduling problem. Also, PSO has been used by authors to schedule tasks in a grid [33].

TCO is another meta-heuristic method which was used in [34] for routing in adhoc networks. After that, TCO has been used by Hedayatzadeh et al. in [35] to optimize numerical functions. TCO has been successfully applied to optimize real-valued parameters. The TCO can be used as a local search method that helps us to increase diversity of search. The search diversification enhances exploration ability of an algorithm. Hence, the algorithm has greater chance to find better solution. However, there is no application of TCO on RCPSP.

Although PSO and its improvements applied on RCPSP, its hybridization with other methods has been seldom applied to solve RCPSP problems. Hence, this work tries to further improve the efficiency of PSO in solving RCPSP problems by introducing new constraint handling method and combining it with the TCO meta-heuristic.

3. Basic concepts

This section presents the required basic concepts for the description of the proposed method. Here, the concepts about PSO, TCO, scheduling scheme, and forward-backward improvement are described.

3.1. Particle swarm optimization

PSO is a swarm intelligence technique inspired from social behavior of bird flocking and fish schooling. PSO has been applied successfully to optimize a wide range of engineering problems. The standard PSO algorithm is an iterative process in which a set of particles are characterized by their position and the velocity with which they move in the solution space of a cost function. Each individual in PSO flies in the parameter space with a velocity which is dynamically adjusted according to the flying experiences of its own and those of its companions. Therefore, every individual is gravitated toward a stochastically weighted average of the previous best point of its own and that of its neighborhood companions [35]. Mathematically, given a swarm of particles, each particle i is associated with a position vector $\vec{X}_i = \{X_{i1}, X_{i2}, ..., X_{iD}\}$, which is a feasible solution for an optimal problem in the D-dimensional search space S. let the best previous position (the position giving the best objective function value called pbest) that particle i has found in the parameter space be denoted by $\vec{X}_{pi}$; the best position that the particles have ever found called gbest is denoted using $\vec{X}_G$. At the start time all of the positions and velocities are initialized randomly. At each iteration, the position vector of each particle i is updated by adding an increment vector $\vec{V}_i = \{V_{i1}, V_{i2}, ..., V_{iD}\}$ so called velocity. In the original PSO algorithm, the particles' positions are updated according to the following equations:

$$\vec{V}_i\{k+1\} = \vec{V}_i\{k\} + c_1 r_1 \left\{\vec{X}_{pi} - \vec{X}_i\{k\}\right\} + c_2 r_2 \left(\vec{X}_G - \vec{X}_i\{k\}\right), \quad (4)$$

$$\vec{X}_i\{k+1\} = \vec{X}_i\{k\} + \vec{V}_i\{k+1\}, \quad (5)$$

where $\vec{V}_i k$ and $\vec{V}_i\{k+1\}$ present velocity vectors for particle i in previous and current iterations, c_1 and c_2 are two positive constants, r_1 and r_2 are two random parameters of uniform distribution in range of [0, 1], which limit the velocity of the particle in the coordinate direction.

This iterative process will continue swam by swarm until a stop criterion is satisfied. In the right-hand side of Eq.(4), the second term represents the cognitive part of the PSO algorithm at which a particle changes its velocity based on its own thinking and memory, while the third term is the social part of the PSO algorithm at which the particle modifies its velocity based on the adaptation of the social-psychological knowledge. In PSO algorithms, exploitation is obtained through selecting the particle with best fitness and moving toward that particle. Each particle memorizes its previous best position to explore the space between its previous best position and the global best position found by swarm [36].

3.2. Termite colony optimization

TCO is an optimization method inspired from intelligent behaviors of termites. A colony of termites

has the ability to perform complex tasks by applying simple rules between its individuals [35]. TCO employs stochastic process which is used by termites to find the optimal solution. Each termite i in the population is associated with a position vector $\vec{X}_i = \{X_{i1}, X_{i2}, ..., X_{iD}\}$ which represents a feasible solution for an optimal problem in the D-dimensional search space S. In TCO, each position vector $\vec{X}_i$ represents a hill with an associated quality which is represented as $fit(\vec{X}_i)$. The fitness value models amount of pheromones which are deposited on the hill. At each cycle of the algorithm, the fitness value of each termite is evaluated. The pheromone content at the location j is computed based on the following equation:

$$\tau_i\{k+1\} = (1-\rho)\,\tau_i\{k\} + 1 \Big/ \left(\text{fit}\left(\vec{X}_i\right) + 1\right), \quad (6)$$

where ρ is the evaporation rate that is taken in range of [0..1], $\tau_i\{k\}$ and $\tau_i\{k+1\}$ are respectively the pheromone level at the current and previous locations of ith termite.

After computing the pheromone levels, each termite adjusts its trajectory based on local information and moves to new location. The termite movement is a function of pheromone level at the visited location and the distance between a termite location and the visited locations. Based on these parameters, two different movement patterns may be introduced for termites. A local region around each termite is considered, and the number of visited position in the neighborhood of the termite is computed. If there is no visited position in the neighborhood of a termite, it moves randomly in its own nearby regions. Termites with one or more visited positions in their neighborhood may select a more profitable position and move toward that position.

A part of termites employ a random movement pattern in order to find more profitable regions. The random walk is performed by the termite in a region with radius . The search region is centered at current position of the termite. So the next position of a termite is updated using the following equation [35]:

$$\vec{X}_i\{k+1\} = \vec{X}_i\{k\} + Rw\left(\tau, \vec{X}_i\{k\}\right), \quad (7)$$

where $\vec{X}_i(k)$ represents the previous position of the termite which is replaced by the new position of that termite (i.e. $\vec{X}_i(k)$), and Rw is a random walk function that depends on the current position of the termite and the radius search τ . The initial value of radius τ is defined as a percentage of $|Ub - Lb|$, where Ub and Lb, respectively, represent the upper bound and lower bound of the search space along each dimension.

The locally observed pheromones provide the ability for a termite to use a selection process in order to adjust its trajectories towards one of these pheromone gradients. In other words, the termite evaluates the provided pheromone information, and adjusts its trajectory towards a position with the highest level of pheromone. A termite i considers the local best position (denoted as $\vec{B}_i$) as its own promising position and moves towards that if its current position has smaller level of pheromone compared to the best local position. The movement trajectory of the termite i is controlled using the following equation [35]:

$$\vec{X}_i\{k+1\} = \vec{X}_i\{k\} + \omega_b r_b\left\{\vec{B}_i - \vec{X}_i\{k\}\right\}, \quad \text{if } \{\tau_i\{k\} < \tau_{b_i}\{k\}\}, \quad (8)$$

where $1 < w_b \leq 2$ and $0 < r_b < 1$ probabilistically controls the attraction of the termite towards local best position.

3.3. Serial versus parallel scheduling scheme

To evaluate the performance of an individual in a solving RCPSP problem we need to build a schedule from a list of activities. Hence, we should use a Schedule Generation Scheme (SGS). Given a list of activities, we can build a schedule using one of the SGSs approaches so-called parallel-SGS and serial-SGS. The serial-SGS performs time-incrementation to construct active schedules. The serial-SGS is an activity oriented scheme that generates a schedule from the activity list in n stages. In each stage, one activity is selected and it is scheduled at the earliest precedence and resources feasible time.

The parallel-SGS performs activity incrementation. In parallel-SGS, the activities with the feasible precedence and resource are scheduled at each point of time. In this way, it is possible to assign more than one activity with different priorities at a certain time. If no activity can be assigned at that time, the time is set to the end time of the shortest in-process activity [37].

3.4. Forward backward improvement

The serial-SGS or parallel-SGS is applied on the solutions provided by the individuals in a meta-heuristic approach in order to build a feasible schedule. Both of these scheduling schemes may consider the activity list in a forward or backward order. In the backward order, the last activity A_{n+1} is considered at first and all the other activities are viewed in reverse. Usually, using the same SGS, the forward and backward approaches return different solutions. The best of these solutions can be used as the result of SGS. To obtain more efficiency, one can use both of the forward and backward approaches to solve the problem in two times. Normally, the better solution is considered.

After generating a schedule using a SGS, it is some times possible to make the schedule shorter by shifting its activities to the right using a backward scan. The newly obtained result can be shifted to the left using a forward scan. This process which reduces the makespan is called forward backward improvement.

4. Hybrid of PSO and TCO

This section describes the hybrid of PSO and TCO called PSTCO method in details. The pseudocode of the PSTCO is given in Figure 1. The PSTCO interleaves search iterations of the PSO with the search iterations of the TCO. The main idea is that search iterations of the PSO being followed by search iterations of TCO in order to utilize potentials of global and local searches.

In one hand, the PSO iterations provide a global search over the solution space. Hence, these iterations produce good solutions which are distributed globally in the solution space. However, in PSO, the local landscape around the individuals is not considered. The local landscape around an individual may provide valuable additional information. Local exploration provides the ability to gravitate an individual towards better position. Upon these considerations, the obtained solutions by the PSO iterations are feed to the TCO iterations.

In other hand, the TCO provides efficient way to perform local search. However, we can improve its performance by providing good starting point in the solution space. Hence, the solutions provided by the PSO iterations are fed to the TCO iterations in order to accelerate termites' movements towards more profitable regions.

Inputs:
max_iter : maximum number of iterations.
n_1: number of PSO iterations.
n_2: number of TCO iterations.
s : population size.
η : number of solutions which are feed from PSO to the TCO and vice versa.

Initialization:
Distribute randomly PSO particles over the solution space.
Update:
Do for each iteration *iter*: ($1 \le iter \le max\_iter$)
If current iteration is a particle iteration
If the time is to switch from TCO to PSO
Feed η solution from TCO to the PSO
Do for each new solution *i*
Replace corresponding particle with solution *i*
Compute fitness of particle *i* using serial-SGS
Update previous best position of particle *i*
Update global best position of the population
End Do
End If
Do for each particle *i* in the particle swarm
Compute fitness of particle *i* using serial-SGS
Update previous best position of particle *i*
Update global best position of the population
Update particle *i* using equations (9) and (5)
End Do
Else
If the time is to switch from TCO to PSO
Feed η solution from PSO to the TCO
Replace corresponding termite with the new solution *i*
Compute fitness of the termites using serial-SGS
Update pheromone level at each position
End If
Do for each termite *i*
Find the neighbor positions for termite *i*
If termite *i* has neighbors **Then**
Select best neighbor position $\vec{b}_i$
Update position of termite *i* using equation (8)
Else
Update position of termite *i* using equation (7)
End If
End Do
Adjust radius τ and step size s
End If
End Do
Terminate:
Return Best Solution

Figure 1. Pseudocode of the proposed method.

The PSTCO method receives the following parameters as inputs: maximum number of iterations (max_iter), population size (s), number of PSO iterations (n_1), number of TCO iterations (n_2), and number of solutions which are fed from PSO (TCO) to the TCO (PSO) at the switching time (η). The parameters n1 (n_2) respectively shows how many times PSO iterations (TCO iterations) should be executed before a switching time. For example, if we set $n_1 = 2$ and $n_2 = 3$, it means that PSTCO executes 2 iterations of PSO, switches from PSO to TCO, executes 3 iteration of TCO, and then switches from TCO to PSO. More precisely, n_1 iterations of PSO are followed by n_2 iterations of TCO.

The PSTCO has seven phases: 1) initialization, 2) global search, 3) switch from global search to local search, 4) local search, 5) switch from local to global search, 6) constraint handling, and 7) termination. The following sections describe the PSTCO phases in details.

4.1. Initialization

The PSTCO method starts with PSO iterations. Hence, we need only to initialize the PSO. The algorithm starts with n particles being placed randomly in the solution space. The proposed hybrid algorithm uses the priority-based representation for their individuals. Each particle presents a position in the search space. If the problem has N activities, the particles will fly in the search space with N dimensions. A position is a candidate for the priority list $\vec{P} = \langle p_1, p_2, ..., p_n \rangle$. Each element of the list fixedly represents an activity and its corresponding value shows the priority of that activity. Hence, the position vector $\vec{X}_i = \{X_{i1}, X_{i2}, ..., X_{iD}\}$ of each individual i represents the priority values of n activities.

Under this configuration, a solution space of priorities will be created where the lower and upper bounds of each dimension is defined as $Lb = 0.0$ and $Ub = 1.0$. Hence the value of each element must be limited to $[Lb, Ub]$. The elements with values larger (smaller) than upper bound (lower bound) are set to $Ub(Lb)$.

4.2. Global search phase

The PSO tries to find better solutions by considering the social knowledge which is obtained by explorations of the particles over the solution space. Each particle presents a possible schedule for the RCPSP problem. The PSO phase follows the standard scenario which is given in Section 3.1. However, to improve the convergence speed of the standard PSO, the velocity equation is changed as follows:

$$\vec{V}_i\{k+1\} = \gamma \times \{\vec{V}_i\{k\} + c_1 r_1\{\vec{X}_{pi} - \vec{X}_i\{k\}\} + c_2 r_2(\vec{X}_G - \vec{X}_i\{k\})\}, \quad (9)$$

where $0 < \gamma < 1$ is a constriction factor that speed up the convergence of the proposed method. Under this configuration, the PSO phase updates the position of each particle by considering its current position, its personal best and the global best solution. Hence, we need to calculate the fitness of the proposed schedules by the particles in order to determine their personal bests as well as the global best solution. Using the serial-SGS, the makespan of each schedule presented by a particle j is calculated using the following equation:

$$\vec{\text{fit}}(x_j) = \frac{1}{m\_\text{span}_j}, \quad (10)$$

where $m - \text{span}_j$ is the makespan of the schedule proposed by the particle j.

The RCPSP is known as a constraint satisfaction problem. The provided solutions by the particles may be feasible or infeasible. Hence, the infeasible solutions should be handled by the PSTCO method. For this purpose we use a new constraint satisfaction approach which is given in Section 4.6. This module investigates the infeasible solutions and replaces it with the new feasible solution which is constructed based on the infeasible ones.

4.3. Switching from global to local search

The PSTCO method simply switches from PSO to TCO. At the switching time, a part of solutions found by the PSO is passed to the TCO. Each solution determines the start position of a termite in the next iteration of TCO. More precisely, at the switching time, each particle switches its type as termite.

4.4. Local search phase

The TCO uses the solutions which are passed from PSO as the start positions of its termites. Next, TCO tries to find improved solutions in the local neighborhoods of those solutions. We follow the standard way as presented in Section 3.2 in order to perform local search in the neighborhoods of the termites. However, the neighborhood for each termite should be defined. In this work, the Euclidian distances of termites from the candidate termite are calculated. If a Euclidian distance is smaller than a threshold, the corresponding termite is considered as a neighbor of the candidate termite. This thresholds are adaptively adjusted by considering the lower bound and upper bound of the search space. The thresholds is defined using following equation:

$$R = 1 - \frac{Q}{\max\_iter} \times iter, \quad (11)$$

where $iter$ is the current iteration, max_$iter$ is the maximum number of iterations, and the parameter Q is manually adjusted based on the maximum number of iteration. Normally, at the first iteration, the threshold R has large size whereas its size decreases as the algorithm proceeds. At each iteration the

neighbors of a candidate termite are considered. The termite with no neighbor moves randomly using Eq.(7). The termite with one or more neighbors selects one of them randomly as its neighbor and updates its position using Eq. (8).

After the termites have been updated, their provided solutions may be infeasible. Similar to the PSO phase, the infeasible solutions are passed to the constraint handling module. This module will try to convert them to feasible solutions.

4.5. Switching from local to global search

Switching from TCO to PSO is quite straightforward. At the switching time, each termite switches its type as particle. The solution found by termites updates the positions of the corresponding particles in the PSO. Hence we need to update the previous best position of each particle as well as the global best position of the swarm. The fitness of the new solution for a particle is compared to its old fitness and if the new solution has better fitness, it will be considered as the personal best position. Also, the global best position is compared with this new personal best position and it will be updated if the current personal best position has better fitness compared to the global best position.

4.6. Constraint handling

RCPSP is a constraint satisfaction problem. In each cycle of the algorithms, after construction of the new solutions, a new population of feasible and infeasible solutions is generated. A solution is called infeasible if it violates the constraints of the problem. The constraint handling process has two steps: detection of infeasible solutions and recovering them to feasible solutions. Assume that a priority list and its corresponding activity list are given. At the first step, we try to construct schedule from activity list using serial SGS method. At each stage of serial SGS method, the next activity with the highest priority is selected to be added to the schedule. If the activity could not be added to the schedule due to a constraint, the solution is considered as infeasible. If this situation occurs, the second step starts. At the second step, the activity which violates the constraints is changed with the next activity with smaller priority and the constraint handling process is applied on the new activity list. This process is iterated until the infeasible solution is converted to a feasible solution.

It should be noted that when two activities are exchanged in the activity list, their corresponding priorities remain in their positions. It means that the activity that violates the constraints, obtains the lower priority and the activity that recovers infeasible solution obtains higher priority. Under this way, the activities which violate constraints are shifted to the right side of the list.

4.7. Termination

The algorithm terminates after the termination condition is satisfied. After termination, the solution with the minimum makespan is returned as the final result.

5. Experiments

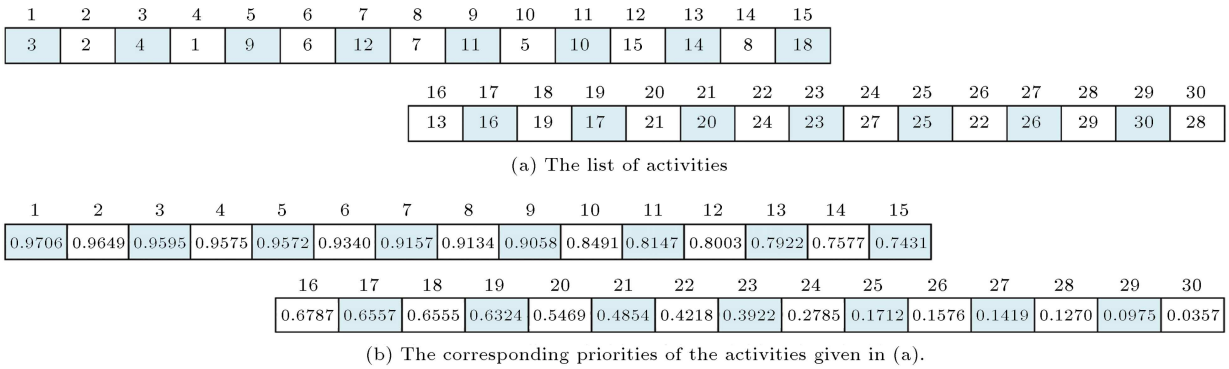
This section presents the experiments conducted for investigating the performance of the proposed algorithms. The investigated algorithms have been applied on the Single Mode Data Sets cases in PSPLIB library [38]. The proposed method has been coded in Matlab. First, the representation of solutions in the proposed method is described. Next, tuning of the parameters is presented. Finally, the proposed method is compared with the other methods.

5.1. Solution representation

The priority-based representation [24] is used by the PSTCO method. Each individual in PSTCO represents a point in a N-dimensional search space. More precisely, each point is a N-dimensional vector in which an element represents an activity and its corresponding value represents the priority. The priority values can vary in range of [0,1]. The solution representation by PSTCO is shown using a project from the j30 case studies. This example has 30 activities and four resources. The minimum makespan of this case study is 53 and the available resources for R1, R2, R3 and R4 are 13, 13, 12, and 15, respectively. The details of this example are given in Figure 2.

Activity	Successors	Duration	Required resources			
			R1	R2	R3	R4
1	11 22	10	5	0	0	8
2	4 7 16	4	10	0	5	0
3	5 18	1	5	0	1	0
4	23	3	0	0	10	5
5	6 9 25	5	3	0	0	0
6	15 21	10	0	0	5	0
7	27	1	0	4	0	0
8	8	4	5	0	0	7
9	10	6	1	0	4	8
10	23	8	2	0	0	1
11	12 21	7	0	10	7	6
12	14 24	7	10	0	0	5
13	25	4	0	6	0	0
14	13	3	9	0	3	10
15	19 27	10	3	5	0	0
16	23	3	10	4	3	10
17	20	4	0	1	0	1
18	17	3	6	6	0	1
19	26	7	0	6	0	0
20	19 20	5	0	0	5	0
21	29	1	2	8	2	1
22	28	10	0	0	0	1
23	22	8	1	0	8	2
24	24 29	1	1	0	0	8
25	26	6	0	3	0	8
26	6	4	0	0	4	0
27	30	7	0	9	0	0
28		6	0	7	0	0
29		9	0	0	4	4
30		4	9	0	1	7
Available resources			13	13	12	15

Figure 2. The details of the example from the j305-1 case studies.



(a) The list of activities

(b) The corresponding priorities of the activities given in (a).

Figure 3. The schedule proposed by PSTCO.

The best solution found by PSTCO method for this case is shown in Figure 3. The figure shows the associated priorities to each of the activities. The activities are sorted in descending order based on their priorities.

The schematic diagram of the proposed schedule by the PSTCO for this case is given in Figure 4. This schedule is produced based on the solution given in Figure 3. The serial schedule generation scheme or parallel schedule generation scheme has been used by researches to construct the schedules. In this example, the serial-SGS is used to construct the schedule of activities. The serial-SGS sequentially adds the activities selected from the sorted list of activities to the schedule until a solution is obtained. At each step of the serial-SGS, the next activity is selected and the first possible position is assigned to that activity such that the precedence and resource constraints are satisfied. As can be seen from Figure 4, the proposed method produces the minimum makespan for this case study.

5.2. Tuning parameters

The proposed method has some parameters. These parameters should be tuned in order to use the proposed method efficiently. The suggested settings of the parameters to evaluate performance of the proposed PSO, TCO, and PSTCO methods are given in Table 1. As can be seen from Table 1, the PSTCO method has eight parameters. It should be noted that a large number of values could be assigned to some of the parameters such as c_1 and it is hard to examine all the combination of the parameters. Hence, we have tried to reduce the number of experiments conducted for tuning of the parameters. For better representation, tunings of the parameters are given in the following categories:

Table 1. Settings of the PSTCO parameters.

Parameter	Value
c_1: acceleration coefficient	1.0
c_2: acceleration coefficient	1.0
γ: constriction factor	0.7
w_b: weight	1.0
n_1: number of PSO iterations	1
n_2: number of TCO iterations	1
Q: radius parameter	0.4
s: population size	35

A) Coefficients and interleaving parameters;

B) Radius parameter;

C) Population size.

A) Coefficients and interleaving parameters. The proposed method interleaves the PSO and TCO phases. Hence the number of PSO and TCO iterations (i.e. n_1 and n_2) in each episode is set at 1. Our empirical study showed that the changes on the coefficient factors c_1, c_2, and coefficient w_b result small changes on the performance of the proposed method. Hence, in this work, their values are set at 1.0. The effect of the constriction factor has been studied by Yin and Wang in [36]. Based on their study, the value of constriction factor is set at 0.7. In addition to these parameters, the PSTCO method has two other parameters that have positive effect on its performance. Tuning of these parameters are given in the next section.

B) Radius parameter. In this experiment, all the parameters except parameter Q have fixed values given in Table 2. The values of parameter Q are varied from 0.1 to 0.8 in step of 0.1. The effect of this parameter on the performance of PSTCO in terms of success rate is given in Table 2. For the j30 case studies, the best result is obtained for $Q = 0.5$, while for the j60, j90, and j120, the best results are obtained when the radius is set at 0.4. Hence, the range of [0.4,0.5] is recommended for parameter Q . In our experiments, the value of parameter Q is set at 0.4 for the comparative study.

C) Population size. It seems that performance of the PSTCO method is affected by the population size.

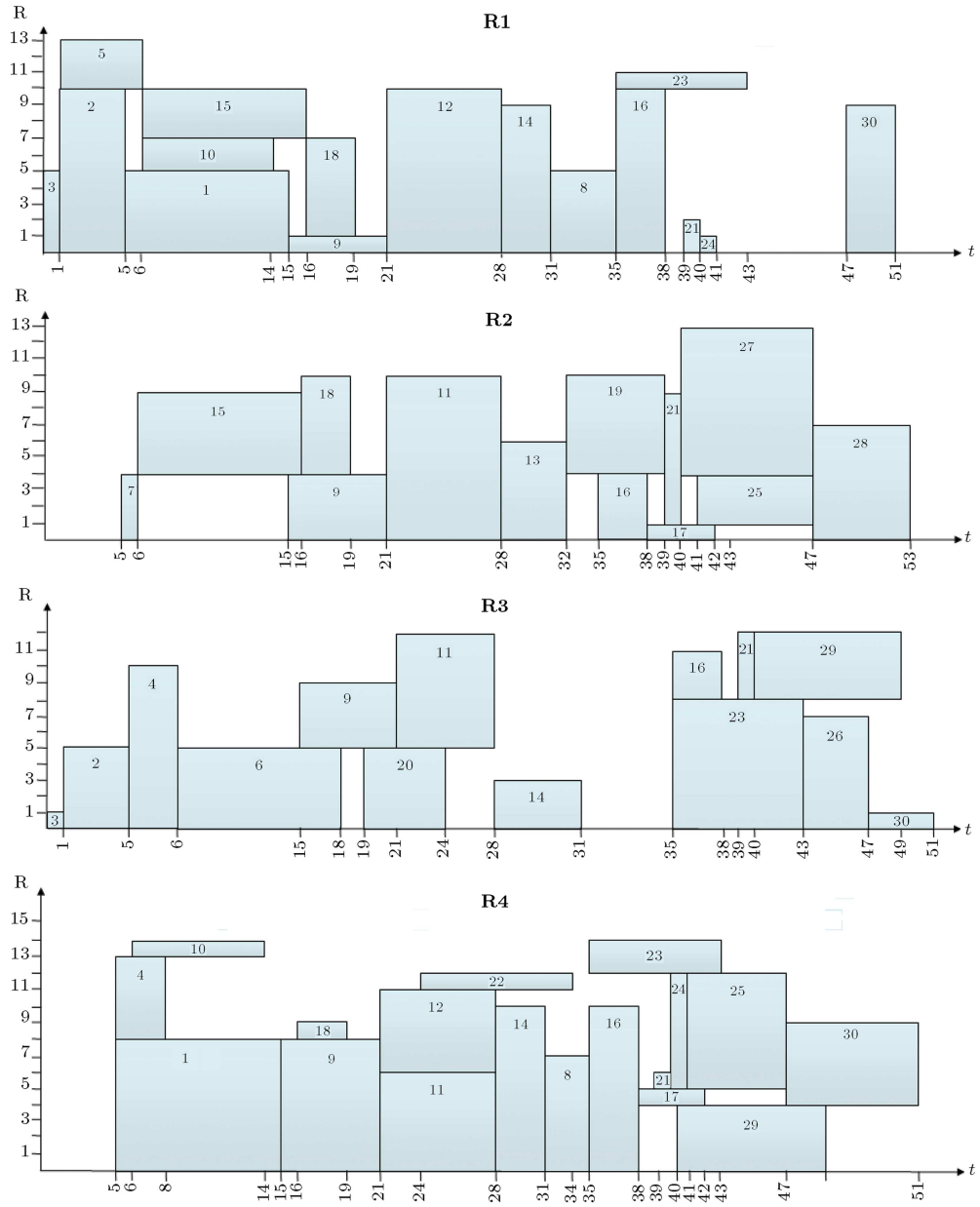


Figure 4. The best solution found by PSTCO.

Table 2. The effect of parameter Q on the performance of the PSTCO method.

	$Q = 0.1$	$Q = 0.2$	$Q = 0.3$	$Q = 0.4$	$Q = 0.5$	$Q = 0.6$	$Q = 0.7$	$Q = 0.8$
j30	0.8604	0.8667	0.8750	0.8958	**0.9000**	0.8708	0.8667	0.8667
j60	0.7292	0.7333	0.7375	**0.7417**	0.7396	0.7354	0.7354	0.7333
j90	0.7312	0.7312	0.7354	0.7438	0.7375	0.7375	0.7292	0.7271
j120	0.2967	0.3000	0.3033	**0.3083**	0.3067	0.3067	0.2983	0.3000

In this experiment the effect of population size of the performance of the PSTCO is studied. For this experiment, the number of produced schedules is set at 5000. The population size is varied from 15 to 40 in step of 5. The effect of population size on the success rate of the PSTCO method is given in Table 3. For the j30 case study, the best result is obtained when the population size is set at 35. For the j60 case study, the best results are obtained when the population size is set at 30 and 35. Finally, the best results for j90 and j120 case studies are obtained by a population of size 35. From the results, it can be seen that the population size

Table 3. The effect of population size on the performance of the PSTCO method.

	Size					
Case study	**15**	**20**	**25**	**30**	**35**	**40**
j30	0.8583	0.8583	0.8729	**0.8979**	0.8938	0.8875
j60	0.7271	0.7292	0.7333	**0.7417**	**0.7417**	0.7354
j90	0.7292	0.7292	0.7354	0.7417	**0.7438**	0.7333
j120	0.2967	0.3000	0.3000	0.3033	**0.3083**	0.3050

has positive effect on the performance of the PSTCO algorithm. In general we recommend the use of the population size in range of [30, 35]. In this paper, the population size is set at 35.

5.3. Comparative study

The best configuration obtained under tuning of parameters is used here to compare the proposed method against other methods. Here, we conduct the following metrics to investigate the performance of the proposed method:

A) Success rate after predefined number of schedules;

B) Average deviation from critical path;

C) Convergence behavior throughout iterations.

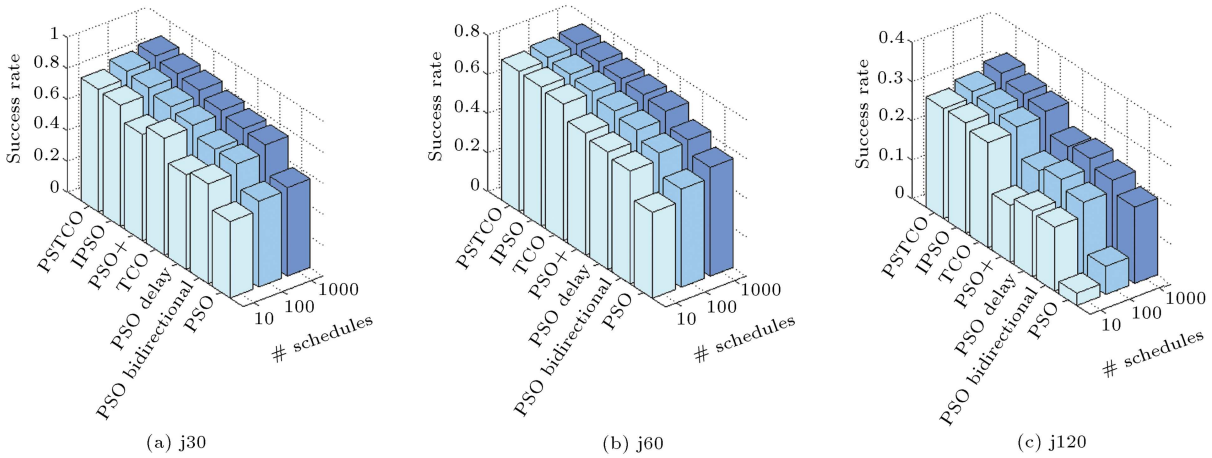
A) Success rate. In the first experiment, the success rate of the proposed method compared to the other variants of PSO is considered. Here we used the following variants of PSO for comparison: standard PSO [28], PSO Delay [31], PSO Bidirectional [31], and PSO+ [31]. This experiment shows how many cases of PSPLIB library [38] can be solved by the proposed algorithm. We say that a case study problem can be solved if the algorithm finds optimal solution or lower bound solution for that problem. Figure 5 presents the experimental results for the j30, j60, and j120 case studies. The vertical axis of each figure indicates the percentage of the problem instances which are successfully solved by an algorithm.

From the results we can see that the modified PSO (called IPSO) method independently outperforms other variants of the PSO method presented in literature. Also, the proposed TCO method has the ability to outperform other PSO-based methods when applied independently over the case studies. However, our proposed PSO dominates the TCO method.

Although PSO and TCO independently provide good solutions, we can obtain better performance with their hybrid variant. In general, the best results were obtained by the PSTCO method over the j30, j60, j90, and j120 case studies.

B) Statistical test. Usually, the statistical tests such as t-test, Wilcoxon test, etc. are used to determine if a proposed method has statistically better performance in comparison with the other methods or not. Such tests help the users to select a method for their purposes. Here, the Wilcoxon Rank Sum Test is used in order to show if the proposed method has significantly better performance compared to other variants of PSO or not. The proposed method statistically has better performance if the p-value between its results and the best results obtained by the other algorithms is smaller than the significance level $\alpha = 0.005$.

In this experiment, the number of produced



(a) j30 (b) j60 (c) j120

Figure 5. Average success rates of the proposed methods in comparison with other variants of PSO method.

Table 4. The results of Wilcoxon rank sum test.

Case study	h
j30	1
j60	1
j90	1
j120	1

schedules is set at 1000. The results of the statistical test are given in Table 4. The one value for the h shows that the proposed method has better performance while the zero value shows the proposed method is not statistically better than the other methods. From the results it can be seen that the proposed method significantly has better performance compared to other PSO-based methods.

C) Average deviation. The objective of this experiment is to compare the ability of the methods in finding the schedule with minimal makespan. The experiments were conducted on the j30 and j60 case studies with 480 instances and for the j120 case study with 600 instances. The complexity of the problem increases as the number of activities increases.

The comparative study investigates the performance of the bee algorithm against other state-of-art algorithms on a set of well-known benchmarks. The proposed method is compared with other heuristic and meta-heuristic approaches, including Genetic Algorithm (GA), Simulated Annealing (SA), Tabu Search (TS), Adaptive Sampling (AS), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO) and their hybrids in view of the average derivation (Av Dev) from the optimal solution (i.e., project duration). The performances of reported methods were obtained from the original papers.

Table 5 shows the average deviations from the optimal makespan of the proposed algorithms compared to other heuristic and meta-heuristic methods. For the j30 case study, the proposed PSTCO algorithm obtains rank 5 between 27 investigated methods after 5000 schedule generations.

For the j60 case study, the proposed PSTCO obtains ranks 2 between 25 investigated methods after 5000 schedule generations.

For the j120 case study the proposed PSTCO algorithm obtains fourth rank between 24 investigated methods after 5000 schedule generations.

5.4. Convergence behavior

The comparative study showed that PSTCO method provides competitive results compared to other investigated algorithm. In this experiment, the effect of local search on the convergence behavior of the proposed method is considered. For this purpose, a problem instance from the j120 case study is considered, and the number of produced schedules is set at 5000. The experiment is iterated for 10 times and the average result is given. Figure 6 shows the average convergence speed of the proposed PSO, TCO, and PSTCO algorithms. The results show that the TCO independently has the lowest convergence speed. Although the TCO algorithm, which only uses the local search, successfully solves this problem instance, it needs more times in order to converge to the optimum solution. The PSO method which provides appropriate level of global search has faster convergence speed compared to the TCO method. The fastest convergence speed is obtained by incorporating the local search potential of the TCO method into the PSO method.

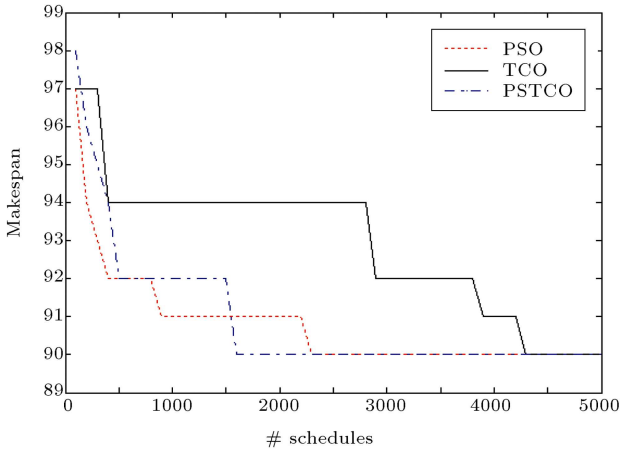


Figure 6. The effect of hybridization on the convergence speed.

In general, hybridization of the PSO method with an efficient local search method not only produces better results compared to other variants of the PSO, but also improves the convergence speed of the proposed algorithm.

6. Conclusions

A hybrid of the PSO and TCO along with a new constraint handling approach for solving resource-constrained project scheduling problem was considered. The proposed method works by interleaving the PSO and TCO iterations. The PSTCO method utilizes the potentials of the PSO and TCO approaches in global and local searches to provide a proper balance between exploration and exploitation.

The performance of the PSTCO algorithm has been investigated on a set of well known benchmarks. The comparative study showed that although the PSO and TCO independently provide good performance in solving the RCPSP problem, more efficiency can be obtained by hybridization between these two methods. The overall performance showed that the proposed algorithm provide efficiency in solving the RCPSP prob-

Table 5. Average deviation (%) from optimal makespan for j30 and critical path lower bound for j60, and j120 case studies.

			j30			j60			j120		
Method	**SGS**	**Ref.**	**1000**	**5000**	**Rank**	**1000**	**5000**	**Rank**	**1000**	**5000**	**Rank**
GAPS	Par.act.	[39]	0.06	0.02	1	11.72	11.04	3	35.87	33.03	2
ACOSS	ser./par.	[40]	0.14	0.06	2	11.75	10.98	1	35.19	32.48	1
ANGEL	Serial	[41]	0.22	0.09	3	11.94	11.27	4	36.39	34.49	7
Neurogenetic	Serial	[37]	0.13	0.10	4	11.51	11.29	5	34.65	34.15	5
PSTCO	Serial	This study	0.27	0.14	5	12.53	11.03	2	34.87	34.01	4
HEDA	Serial	[42]	0.38	0.14	6	11.97	11.43	6	35.44	33.61	3
TS-activity list	Serial	[43]	0.46	0.16	7	12.97	12.18	11	40.86	37.88	11
Sampling-LFT, FBI	Both	[44]	0.30	0.17	8	11.88	11.62	7	35.01	34.41	6
GA-self-adapting	Both	[45]	0.38	0.22	9	12.21	11.70	8	37.19	35.39	8
SA-activity list	Serial	[46]	0.38	0.23	10	12.75	11.90	10	42.81	37.68	10
GA-activity list	Serial	[47]	0.54	0.25	11	12.68	11.89	9	39.37	36.74	9
GA – late join	Serial	[48]	0.74	0.33	12	13.28	12.63	13	39.97	38.41	12
PSO	Per.	[28]	0.69	0.42	13	-	-	-	-	-	-
Sampling- adaptative	ser./par.	[49]	0.65	0.44	14	12.94	12.58	12	39.85	38.70	14
TS-schedule scheme	spec.	[47]	0.86	0.44	15	13.80	13.48	21	-	-	-
Sampling-adaptative	ser./par.	[42]	0.74	0.52	16	13.51	13.06	15	41.37	40.45	18
Single pass/ sampling-LFT	Serial	[44]	0.83	0.53	17	13.96	13.53	23	42.84	41.84	21
Sampling – global	Serial	[48]	0.81	0.54	18	13.80	13.31	19	41.36	40.46	19
GA-random key	Serial	[16]	1.03	0.56	19	14.68	13.32	20	45.82	42.25	22
ABC	Serial	[24]	0.98	0.57	20	14.57	13.12	16	43.24	39.87	17
PSO	P. of A	[28]	0.92	0.61	21	-	-	-	-	-	-
Sampling-rand.	Serial	[50]	1.44	1.00	22	15.94	15.17	25	49.25	47.61	24
GA-priority rule	Serial	[16]	1.38	1.12	23	13.30	12.74	14	39.93	38.49	13
Single pass/ sampling-WCS	par.	[51]	1.40	1.28	24	13.66	13.21	17	39.65	38.77	16
Single pass/ sampling-LFT	par.	[44]	1.40	1.29	25	13.59	13.23	18	39.60	38.75	15
Sampling-random	par.	[50]	1.77	1.48	26	14.89	14.30	24	44.46	43.05	23
GA-problem space	ext.par.	[52]	2.08	1.59	27	14.33	13.49	22	42.91	40.69	20

lem. The PSTCO method provides better performance compared to other PSO-based methods presented in literature.

References

1. Akbari, R., Zeighami, V. and Ziarati, K. "Artificial bee colony for resource constrained project scheduling problem", *International Journal of Industrial Engineering Computations*, **2**(1), pp. 45-60 (2011).
2. Carruthers, J.A. and Battersby, A. "Advances in critical path methods", *Operational Research Quarterly*, **17**, pp. 359-380 (1966).
3. Demeulemeester, E. and Herroelen, W. "New benchmark results for the resource-constrained project scheduling problem", *Management Science*, **43**(11), pp. 1485-1492 (1997).
4. Brucker, P., Knust, S., Schoo, A. and Thiele, O. "A

branch & bound algorithm for the resource-constrained project scheduling problem", *European Journal of Operational Research*, **107**(2), pp. 272-288 (1998).

5. Mingozzi, A., Maniezzo, V., Ricciardelli, S. and Bianco, L. "An exact algorithm for project scheduling with resource constraints based on new mathematical formulation", *Management Science*, **44**(5), pp. 714-729 (1998).
6. Dorndorf, U., Pesch, E. and Phan-Huy, T. "A branch-and-bound algorithm for the resource-constrained project scheduling problem", *Mathematical Methods of Operations Research*, **52**, pp. 413-439 (2000).
7. Pritsker, A.A.B., Watters, L.J. and Wolfe, P.M. "Multiproject scheduling with limited resources: a zero-one programming approach", *Management Science*, **16**, pp. 93-107 (1969).
8. Patterson, J.H. and Huber, W.D. "A horizon-varying zero-one approach to project scheduling", *Management Science*, **20**, pp. 990-998 (1974).
9. Patterson, JH. and Roth GW. "Scheduling a project under multiple resource constraints: A zero-one programming approach", *AIIE Transactions*, **8**, pp. 449-455 (1976).
10. Hartmann, S. and Kolisch, R. "Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem", *European Journal of Operational Research*, **127**, pp. 394—407 (2000).
11. Kolisch, R. and Hartmann, S. "Experimental investigation of heuristics for resource-constrained project scheduling: An update", *European Journal of Operational Research*, **174**, pp. 23-37 (2006).
12. Kolisch, R. and Hartmann, S. "Heuristic algorithms for solving the resource-constrained project scheduling problem: Classification and computational analysis", J. Weglarz, Ed., *Project Scheduling: Recent Models, Algorithms and Applications*, Kluwer Academic Publishers, pp. 147-178 (1999).
13. Kolisch, R. and Padman, R. "An integrated survey of deterministic project scheduling", *Omega*, **29**, pp. 249-272 (2001).
14. Thomas, P. and Salhi, R. "SA tabu search approach for the resource constrained project scheduling problem", *Journal of Heuristics*, **4**, pp. 123-139 (1998).
15. Boctor, F.F. "An adaptation of the simulated annealing algorithm for solving resource-constrained project scheduling problems", *International Journal of Production Research*, **34**, pp. 2335-2351 (1996).
16. Hartmann, S. "A competitive genetic algorithm for resource-constrained project scheduling", *Naval Research Logistics*, **45**, pp. 279-302 (1998).
17. Hartmann, S. "A self-adapting genetic algorithm for project scheduling under resource constraints", *Naval Research Logistics*, **49**, pp. 433-448 (2002).
18. Mendes, J.J.M., Goncalves, J.F. and Resende, M.G.C."A random key based genetic algorithm for the resource constrained project scheduling problem", *Computers and Operations Research*, **36**, pp. 92-109 (2009).
19. Ranjbar, M., Kianfar, F. and Shadrokh, S. "Solving the resource availability cost problem in project scheduling by path relinking and genetic algorithm", *Applied Mathematics and Computation*, **196**, pp. 879-888 (2008).
20. Merkle, D., Middendorf, M. and Schmeck, H. "Ant colony optimization for resource-constrained project scheduling", *IEEE Transactions on Evolutionary Computation*, **6**, pp. 333-346 (2002).
21. Jarboui, B., Damak, N., Siarry, P. and Rebai, A. "A combinatorial particle swarm optimization for solving multi-mode resource-constrained project scheduling problems", *Applied Mathematics and Computation*, **195**, pp. 299-308 (2008).
22. Luo, X., Wang, D., Tang, J. and Tu, Y. "An improved PSO algorithm for resource constrained project scheduling problem", *In the Sixth World Congress on WCICA Intelligent Control and Automation*, **1**, pp. 3514-3518 (2006).
23. Zhang, C., Sun, J., Zhu, X. and Yang, Q. "An improved particle swarm optimization algorithm for flowshop scheduling problem", *Information Processing Letters*, **108**(4), pp. 204-209 (2008).
24. Ziarati, K., Akbari, R. and Zeighami, V. "On the performance of bee algorithms for resource constrained project scheduling problem", *Journal of Applied Soft Computing*, Elsevier, **11**(4), pp. 3720-3733 (2011).
25. Akbari, R., Mohammadi, M. and Ziarati, K. "A novel bee swarm optimization algorithm for numerical function optimization", *Journal of Communications in Nonlinear Science and Numerical Simulation*, **15**, pp. 3142-3155 (2010).
26. Zeighami, V., Akbari, R., Akbari, I. and Biletskiy Y. "An ABC-genetic method to solve resource constrained project scheduling problem", *Artificial Intelligence Research (AIR) Journal*, SCIEDU, Canada, **1**(2), pp. 185-197 (2012).
27. Kennedy, J. and Eberhart, R. "Particle swarm optimization", *Proceeding of IEEE Int. Conf. Neural Networks*, **4**, pp. 1942-1947 (1995).
28. Zhang, H., Li, X., Li, H. and Huang, F. "Particle swarm optimization-based schemes for resource-constrained project scheduling", *Journal of Automation in Construction*, **14**, pp. 393-404 (2005).
29. Hong, Z., Li, H. and Tam, C.M. "Particle swarm optimization for resource constrained project scheduling", *International Journal of Project Management*, **24**, pp. 83-92 (2006).
30. Tchomte, S.K., Gourgand, M. and Quilliot, A. "Solving resource-constrained project scheduling problem with particle swarm optimization", *Proceeding of 3rd*

Multidsciplinary Int. Scheduling Conference (MISTA 2007), pp. 251-258 (2007).

31. Chen, R.M., Wub, C.L., Wang, C.M. and Lo, S.T. "Using novel particle swarm optimization scheme to solve resource-constrained scheduling problem in PSPLIB", *Expert Systems with Applications*, **37**, pp. 1899-1910 (2010).

32. Zhang, C., Sun, J., Zhu, X. and Yang, Q. "An improved particle swarm optimization algorithm for flowshop scheduling problem", *Information Processing Letters*, **108**(4), pp. 204-209 (2008).

33. Chen, T., Zhang, B., Hao, X. and Dai, Y. "Task scheduling in grid based on particle swarm optimization", *The Fifth International Symposium on Parallel and Distributed Computing*, pp. 238-245 (2006).

34. Roth M. and Wicker, S. "Termite: Ad-Hoc networking with stigmergy", *Proceeding of IEEE International Conference on Global Economics*, pp. 2937-2941 (2003).

35. Hedayatzadeh, R., Akhavan Salmasi, F., Keshtgari, M., Akbari, R. and Ziarati, K. "Termite colony optimization: a novel approach for optimizing continous problems", *Proceeding of 18th Iraninan International Conference on Electerical Engineeding*, pp. 1-6 (2010).

36. Yinand, P.Y. and Wang, J.Y. "A particle swarm optimization approach to the nonlinearresource allocation problem", *Applied Mathematics and Computation*, **183**, pp. 232-242 (2006).

37. Agarwal, A., Colak, S. and Erenguc, S. "A Neurogenetic approach for the resource-constrained project scheduling problem", *Computers and Operations Research*, **38(1)**, pp. 44-50 (2011).

38. Project Scheduling Problem Library - PSPLIB: <http://www.129.187.106.231/psplib/ >.

39. Mendes, J.J., Gonc-alves, J.F. and Resende, M.G.C. "A random key based genetic algorithm for the resource constrained project scheduling problem", *Journal of Computers and Operations Research*, **36**, pp. 92-109 (2009).

40. Chen, W., Shi, Y.J., Teng, H.F., Lan, X.P. and Hu, L.C. "An efficient hybrid algorithm for resource-constrained project scheduling", *Information Sciences*, **180**, pp. 1031-1039 (2010).

41. Tseng, L.Y. and Chen, S.C. "A hybrid metaheuristic for the resource-constrained project scheduling problem", *European Journal of Operational Research*, **175**, pp. 707-721 (2006).

42. Schirmer, A. "Case-based reasoning and improved adaptive search for project scheduling", *Naval Research Logistics*, **47**, pp. 201-222 (2000).

43. Nonobe, K. and Ibaraki, T. "Formulation and tabu search algorithm for the resource constrained project scheduling problem", Ribeiro, C.C., and Hansen P., Editors, *Essays and Surveys in Metaheuristics*, Dordrecht, Kluwer Academic Publishers, pp. 557-88 (2002).

44. Kolisch, R. "Serial and parallel resource-constrained project scheduling methods revisite: Theory and computation", *European Journal of Operational Research*, **90**, pp. 320-33 (1996).

45. Baar, T., Brucker, P. and Knust, S. "Tabu-search algorithms and lower bounds for the resource-constrained project scheduling problem", Voss, S., Martello, S., Osman, I., Roucairol, C., Editors, *Meta-Heurisitics: Advances and Trends in Local Search Paradigms for Optimization*. Dordrecht, Kluwer, pp. 1-8 (1998).

46. Bouleimen, K. and Lecocq, H. "A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version", *European Journal of Operational Research*, **149**, pp. 268-81 (2003).

47. Hartmann, S. "A competitive genetic algorithm for resource-constrained project scheduling", *Naval Research Logistics*, **45**, pp. 279-302 (1998).

48. Coelho, J. and Tavares, L. "Comparative analysis of meta-heuricstics for the resource constrained project scheduling problem", *Technical Report*, Department of Civil Engineering, Instituto Superior Tecnico, Portugal (2003).

49. Kolisch, R. and Drexl, A. "Adaptative search for solving hard project scheduling problems", *Naval Research Logistics*, **43**, pp. 23-43 (1996).

50. Kolisch, R. "Project scheduling under resource constraints: Efficient heuristics for several problem classes", *Wurzburg: Physica-Verlag* (1995).

51. Kolisch, R. "Efficient priority rules for the resource-constrained project scheduling problem", *Journal of Operations Management*, **14**, pp. 179-92 (1996).

52. Leon, V.J. and Ramamoorthy, B. "Strength and adaptability of problem-space based neighborhoods for resource constrained scheduling", *Operations Research Spektrum*, **17**, pp. 173-182 (1995).

Biographies

Vahid Zeighami received his BS degree in Applied Mathematics and his MSc degree in Operations Research from Shiraz University in 2008 and 2011, repectively. Currently, he is a PhD candidate at Ecole Polytechnique de Montreal, Canada. His areas of research include meta-heuristic algorithms, optimization problems, operation research, integer programming, linear and non-linear optimization.

Reza Akbari received his MSc in artificial intelligence from Isfahan University of Technology (2006). He received his Ph.D. from Shiraz University (2011). He is an Assistant Professor at the Department of Computer Engineering and Information Technology in Shiraz University of Technology. His areas of research include

evolutionary computation, engineering optimization, and search based software engineering.

Koorush Ziarati graduated in Electronics Engineering (control), in University of Science & Technologies Houari Boumedienne in Algeria (1990). He obtained his MSc degree in Control-Robotics and his PhD in Operations Research from Ecole Polytechnique de Montreal, Canada, in 1993 and 1997, respectively. He has been a faculty member of Department of Computer Science & Engineering and Information Technology in Shiraz University since 1999. Currently, his research interests concern transportation, assignment and scheduling problem, heuristic methods for global optimization problem, integer and convex programming.