Expert Systems with Applications 37 (2010) 1899–1910

Contents lists available at ScienceDirect

Expert Systems with Applications

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

# Using novel particle swarm optimization scheme to solve resource-constrained scheduling problem in PSPLIB

Ruey-Maw Chen [a,*], Chung-Lun Wu [b], Chuin-Mu Wang [a], Shih-Tang Lo [c]

[a] *Department of Computer Science and Information Engineering, National Chin-yi University of Technology, Taichung 411, Taiwan, ROC*
[b] *Department of Electrics Engineering, National Chin-yi University of Technology, Taichung 411, Taiwan, ROC*
[c] *Department of Information Management, Kun Shan University, Tainan 710, Taiwan, ROC*

ARTICLE INFO

*Keywords:*
Particle swarm optimization
Scheduling
Delay local search
Bidirectional scheduling
Critical path method

ABSTRACT

This investigation proposes an improved particle swam optimization (PSO) approach to solve the resource-constrained scheduling problem. Two proposed rules named delay local search rule and bidirectional scheduling rule for PSO to solve scheduling problem are proposed and evaluated. These two suggested rules applied in proposed PSO facilitate finding global minimum (minimum *makespan*). The delay local search enables some activities delayed and altering the decided start processing time, and being capable of escaping from local minimum. The bidirectional scheduling rule which combines forward and backward scheduling to expand the searching area in the solution space for obtaining potential optimal solution. Moreover, to speed up the production of feasible solution, a critical path is adopted in this study. The critical path method is used to generate heuristic value in scheduling process. The simulation results reveal that the proposed approach in this investigation is novel and efficient for resource-constrained class scheduling problem.

## 1. Introduction

Scheduling notion has been widely applied in many fields, such as production planning (Zhai, Tiong, Bjornsson, & Chua, 2006), process scheduling in operating systems (Shaw et al., 1999), classroom arrangement (Mathaisel & Comm, 1991), aircrew-scheduling (Chang, 2002), nurse scheduling (Ohki, Morimoto, & Miyake, 2006), food industrial (Simeonov & Simeonovova, 2002), control system (Fleming & Fonseca, 1993) and grid computing (Buyya, Abramson, & Giddy, 2000). Generally, these problems commonly accompany the cost considerations related to certain constraints; therefore, scheduling scheme plays an important role in obtaining solutions for constraints satisfaction and cost minimization.

The resource-constrained scheduling problem (Merkle, Middendorf, & Schmeck, 2002) is an optimization problem considering how to schedule the activities of a multiprocessor system such that the *makespan* of the schedule is minimized while satisfying given precedence constraints between the activities and resource requirements of the scheduled activities per time unit do not exceed the given different types of resources capacity limit. However, the minimum *makespan* is hard to be obtained since the inestimable situation of constraints. Most scheduling problems are confirmed to be NP-complete combinatorial problems, especially for large scale scheduling problems. There are some algorithms were studied to solve scheduling problems and obtain optimal solution. Branch-and-bound method (Jalilvand, Khanmohammadi, & Shabaninia, 2005) is able to find optimal solutions of scheduling problem. However, the execution time required is impractical when the number of activities increases. Comparatively, several greedy-based algorithms such as shortest job first (SJF) (Lupetti & Zagorodnov, 2006) and priority scheduling (Li, Bettati, & Zhao, 1997) are able to solve the resource-constrained scheduling problem in feasible time, but it is hard to accommodate the algorithm to the changed problem's constraints situation. Hence, the optimal solution is seldom obtained by greedy-based algorithms.

Hopfield and Tank (1985) first proposed a type of artificial neural network; named Hopfield neural network (HNN) for solving optimization problems. The HNN mimics the cooperation of all neurons in the brain (cerebrum) and learns to solve problems. We have applied HNN to scheduling multiprocessor job with resource and timing constraints (Huang & Chen, 1999). Furthermore, Chen, Lo, and Huang (2007) combined competitive scheme with slack neurons to solve real-time job scheduling problems. In HNN, the neurons' output is decided based on the input information; synaptic weights and bias inputs from outside of the neuron. An energy (cost or fitness) function is adopted to confine the problems' constraints and targets. The solution of a scheduling problem is yielded as the energy function converges. However, the energy function is not easy to be designed once the constraints of the scheduling problems become complicated.

* Corresponding author. Fax: +886 4 23917426.
*E-mail addresses:* raymond@mail.ncut.edu.tw (R.-M. Chen), arashilen@yahoo.com.tw (C.-L. Wu), cmwang@mail.ncut.edu.tw (C.-M. Wang), edwardlo@mail.ksu.edu.tw (S.-T. Lo).

0957-4174/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2009.07.024

Many studies solve the resource-constrained scheduling problem using the meta-heuristics based methods, such as genetic algorithm (GA) (Liu & Wang, 2005), simulated annealing algorithm (SA) (Bouleimen & Lecocq, 2003), tabu search (TS) (Thomas & Salhi, 1998), ant colony optimization (ACO) (Merkle et al., 2002), particle swarm optimization (Luo, Wang, Tang, & Tu, 2006), and others. The GA mimics the mechanism of natural selection as global evolution (Holland, 1987); then a more superior solution part will be inherited via crossover operation, and increasing the diversity of solution via mutation process. Both crossover and mutation steps are used to expect obtaining a better solution of the studied problem.

Originally, simulated annealing was investigated by Kirkpatrick et al. as a stochastic method for combinatorial optimization problem (Kirkpatrick, Gelatt, & Vecchi, 1983). The optimal solution is a stable state when the thermal energy of the system minimized. Therefore, the required procedure is to decrease the thermal energy of the system by cooling down temperature parameter. The temperature lowered as the iteration goes. Restated, cooling the system to lower energy state. Moreover, the SA applies a probability to decide whether allows the system toward the higher energy state to avoid trapped on the local optimum. Tabu search is an approach to prevent the search from sinking into the local minimum. The key skill used in tabu search is to record the solutions which have been obtained. Therefore, the following search can avoid searching the same obtained solutions (Glover Fred., 1989).

The ACO emulates the foraging behavior of ants (Dorigo & Gambardella, 1997). The ant left pheromone on the trail of the searched path from nest to the destination. The pheromone deposited on the way is for other ants to identify and communicate with each other. Additionally, the amount of pheromone is inverse proportional to the length of path; a large amount of pheromone is accumulated at the shorter path. Accordingly, ants are toward the path with plenty of pheromone. The maximum amount of pheromone on the path can be regarded as an ant notification signal indicating where the shorter path is located at. Hence, the amount of pheromone increases rightly as more ants pass through the shortest path. It is a self-reinforce situation of searching the shortest path, this situation will result fast convergence and trap in local optimal solution. Therefore, an evaporation mechanism is applied in ACO to reducing the pheromone to obtain global optimal solution. Using ACO to solve multiprocessor system scheduling with precedence and resources constraints was proposed (Chen, Lo, Wang, & Wu, 2006; Chen, Zhang, Hao, & Dai, 2006).

The particles swarm optimization (PSO) is first proposed by Kennedy and Eberhart (1995). In PSO, a swarm of particles spreads in the space and the position of a particle represents a solution of a dedicated problem. Each particle would move to a new position based on the global experience of the swarm and the individual experience of the particle for the global optimum. The PSO has been applied to solve the scheduling problems. Luo et al. (2006) used PSO to solve resource-constrained project scheduling problem (PCPSP); they showed that the PSO is applicable to various combinatorial problems and scheduling problems. Zhang, Sun, Zhu, and Yang (2008) solved flowshop scheduling problem (FSP) in terms of the PSO, and Chen, Lo et al. (2006) and Chen, Zhang et al. (2006) solved task scheduling in grid based on PSO. Hence, this investigation further modifies the PSO by involving extra mechanisms to enhance the efficiency in solving the scheduling problem.

The characteristics of above stated algorithms are concluded as follows.

- Most of them require a fitness function to evaluate the quality and feasibility of the obtained solution.
- The features or characteristics of a good solution are referred to when constructing a new solution.
- In GA, a fragment of the good solution is combined with another fragment of another feasible solution to build a new solution.
- ACO and PSO algorithms infer new solution from the helpful experience, and are possible to produce premature convergence. Accordingly, these two algorithms will converge to the local optimal solution. Therefore, most suggested algorithms usually include additional techniques to escape from the premature convergence. For example, randomly disturbing the system or depressing the self-reinforce is applied (Selman, Kautz, & Cohen, 1994). Eventually, there are two aspects to improve the algorithm for scheduling global optimal solution; one is to enhance the heuristic ability (exploitation ability), the other is to strengthen the ability of escaping local optimum or expanding the searching area in the solution space (exploration ability).

This study improved PSO to solve scheduling problem for optima or near-optima schedule by minimizing *makespan*. This improved PSO uses the critical path method to enhance the heuristic ability (exploitation ability). The critical path method considers only logical dependencies between activities but resources availability. However, the critical path method still helps speeding up PSO to gain the feasible solution when the conflict of resources is not acute. Meanwhile, a delay local search to lead the solution is applied in this improved PSO to escape from the local optima (exploration ability). With delay local search, the scheduling system leaves some activities not to be activated occasionally and hence remains some unused resources available for other activities. Moreover, bidirectional scheduling combines forward and backward scheduling to expand the searching area in the solution space is integrated into this improved PSO (Li & Willis, 1992). The resources allocation for forward and backward scheduling is different, and hence two different schedules may be obtained, namely, schedules become diversified. The experimental simulations demonstrate that the approach of this study for the cases of RCPSP is able to obtain more optima or near-optima solutions.

This article is organized as follows. Section 2 introduces the RCPSP and PSO, and describes how to use PSO to solve RCPSP. Section 3 presents that the proposed novel PSO uses the critical path method to enhance the heuristic ability (exploitation), escapes from the local optima using delay local search (exploration) and expands the searching area in the solution space in terms of integrating the bidirectional scheduling. The simulated cases and results of experiments are displayed in Section 4. Finally, Section 5 presents the conclusions and discussions.

## 2. The resource-constrained scheduling problem and particle swam optimization (PSO)

### 2.1. Resource-constrained scheduling problem

The scheduling application has been applied in various fields. Among them, the resource-constrained scheduling application is a general scheduling problem involving activities to be scheduled to some identical processors with precedence and resource constraints satisfied. Activities are confined to the various constraints and achieve a certain object. The studied scheduling problem with resource constraint in this investigation is defined as follows:

- The object of the scheduling is to find the minimal *makespan* schedule.
- There're $N$ activities in the multiprocessor system, Every activity requires a processing duration $d_j$ ($j = 1,...,N$). Meanwhile, activities are non-preemptive in the schedule.

- Activities have precedence constraints; a feasible schedule is an assignment of activities to processors such that an activity comes in the schedule when all its predecessors have finished. Let *Pj* (*Sj*) be the set of direct predecessors (relative to the direct successors) of activity *j*. Activity 0 is the start activity that has no predecessor while activity $(N+1)$ is the finishing activity that has no successor. The start activity and the finishing activity have no resource requirements for any type of resources and have no processing duration.
- There are different types of resources available in the system. Each type of resource has a limited quantity. Let $Q$ be a set of $q$ resource types and $R_k$ $(k = 1, \ldots, q)$ is the resource capacity for resource type $k$. Every activity $j$ requires various resources $r_{j,1}, r_{j,2}, \ldots, r_{j,q}$, where $r_{j,k}$ denotes the quantity of resource type $k$ demanded by activity $j$ when activity $j$ is scheduled. Moreover, resource can be reallocated to other activities when that resource is released from finished activities.
- Moreover, the resource constraints also assumed that the total amount of a resource type $k$ required by scheduled activities can not exceed $R_k$ at any time, such that $\sum_{j \in S_i(t)} r_{j,k} \leq R_k$, where the $S_i(t)$ is the set of activities scheduled by particle $i$ at time $t$. Therefore, the solution $S_i$ contains different activities scheduled at different time.

### 2.2. The particle swarm optimization (PSO)

The particle swarm optimization (PSO) is a multi-agent general meta-heuristic method, and can be applied extensively in solving many tough problems. The PSO consists of a swarm of particles in the space; the position of a particle is indicated by a vector which presents a solution. PSO is initialized with a population of $M$ particles randomly spreads in the space and then starts to search for the best position (namely, schedule herein). At each step or iteration, the local best of each particle and global best of the swarm are determined through evaluating the performances, i.e., the fitness values or *makespan*, of current distribution of particles. A particle moves to a new position as a new solution which is guided by the velocity (a vector). Therefore, the velocity plays an important role in creating a new solution.

There are two experience positions are used in the PSO. One is the global experience position of all particles of the swarm, which memorizes the global best solution obtained from all positions (solutions) of all particles. The other is the individual experience position of each particle, which memorizes the local best solution acquired from the positions (solutions) which have been passing through by the corresponding particle. These two experience positions combining the weighted previous velocities are used to determine the impact or influence on the current velocity. Restated, the current velocity retains part of previous velocity (called the inertia) and driving particle toward the direction based on the global experience position and the individual experience position. Accordingly, the particles can reach new positions (solutions) based on their own inertia and experiences.

Let an $N$ dimension space (the number of dimension is corresponding to the components of solution) has $M$ particles. For the $i$th particle $(i = 1, \ldots, M)$, the position (solution) consists of $N$ components, i.e., $X_i = \{X_{i1}, \ldots, X_{iN}\}$, $X_{ij}$ is the $j$th component of the position. The velocity of particle $i$ is $V_i = \{V_{i1}, \ldots, V_{iN}\}$, where $V_{ij}$ is the $j$th component of the velocity. The individual experience is a position, $L_i = \{L_{i1}, \ldots, L_{iN}\}$ denotes the local best solution for the $i$th particle. Moreover, $G = \{G_1, \ldots, G_N\}$ represents the global best experience shared among all the population of particles achieved so far. The mentioned parameters above are used to update the $j$th component of the position and the $j$th component of velocity for the $i$th particle, as shown in Eq. (1).

$$\begin{cases} V_{ij}^{new} = w \times V_{ij} + c_1 \times r_1 \times (L_{ij} - X_{ij}) + c_2 \times r_2 \times (G_j - X_{ij}) \\ X_{ij}^{new} = X_{ij} + V_{ij}^{new} \end{cases}, \quad j = 1, \ldots, N \tag{1}$$

where $w$ is an inertia weight used to determine the influence of the previous velocity on the new velocity. The $c_1$ and $c_2$ are learning factors. They are used to drive the $i$th particle how to approach the new position either for close to the individual experience position or expecting close to global experience position. Furthermore, the $r_1$ and $r_2$ are the random numbers uniformly distributed in [0, 1], controlling the tradeoff between the global and local exploration abilities during search. The values of $w$, $c_1$ and $c_2$ are assumed to be positive herein.

## 3. Improved particle swarm optimization

The procedure of the proposed improved particle swam optimization is shown as in Table 1. The delay local search rule and bidirectional scheduling rule will be discussed in Section 4.

For particle $i$, each activity $j$ has a defined priority $pr_{ij} > 0$ $(j = 1, \ldots, N)$ for deciding which activity will be selected. Generally, the activity with the higher priority will be selected first, and assign to one processor. In this proposed PSO, the activity's priority $pr_{ij}$ determination is based on the $X_{ij}^{new}$.

### 3.1. Heuristic function design with critical path consideration

To speed up convergence of the algorithm for problem, the critical path is involved in this study. The critical path is computed without the consideration of resources constraints. One activity has a higher heuristic value, while the time is close to or over the activity's latest starting time obtained from critical path. Restated, the critical path is applied to determine the heuristic value which hence affects the probability based on greedy property for selecting activity $j$ of particle $i$ at time $t$, and is used in the activity selection rule. The heuristic value definition is given in Eq. (2).

The critical path method in this study is used to obtain the activity's earliest starting time $Estst_j$, latest starting time $Lstst_j$ and the difference $D_j$ between $Lstst_j$ and $t$ $(D_j = |Lstst_j - t|)$. The $g_{ijt}$ is defined as the heuristic value for scheduling activity $j$ of $i$th particle at time $t$. Intrinsically, the $g_{ijt}$ is defined for the timing constraints for activity $j$, to meet the timing requirement. The activity $j$ is unable to be selected at time $t$ when the time $t$ is earlier than activity's earliest starting time, then the $g_{ijt}$ is set to zero.

**Table 1**
Pseudo code of proposed novel PSO (NPSO).

```
Initialize
Loop
i = 0
The bidirectional scheduling rule decides using forward scheduling
  or backward scheduling solution
  Loop
    i = i + 1
    Initialize t = 0, and J_i(t = 0)
    Loop
      Loop
        Select one activity $j' \in J_i(t)$ with activity selection rule
        If delay local search is activated then either continue or
  delay activities
        else schedule activity $j'$ start at time $t$
        $J_i(t) = J_i(t) - \{j'\}$
      Until $j' = \varnothing$
      $t = t + 1$
    Until $J_i(t) = \varnothing$
    Update the local or global optimal solution if necessary
  Until all particles have built a complete solution
  Updating position and velocity of every particle by Eq. (1)
Until End condition is reached
```

Furthermore, the activity *j*'s earliest starting time is ahead of current time *t* but the latest starting time is behind the time *t*, the $g_{ijt}$ is inverse proportional to the $D_j$. The $g_{ijt}$ increases as the $D_j$ increases when the time *t* is already later than the activity *j*'s latest starting time. Restated, the activity *j* is associated with higher $g_{ijt}$ if the time *t* approaches the latest starting time of activity *j*.

Moreover, the notion of state transition rule of ACO is utilized to generate the new solution in this study. The state transition rule used in ACO is computed by the pheromone ad heuristic value. Accordingly, the heuristic value applied in this proposed novel PSO would affect the probability of selecting activity *j* of particle *i* at time *t.*

$$G_{ijt} = \begin{cases} 0, & \text{if } Estst_j > t \\ \frac{1}{(D_j+1)}, & \text{if } Estst_j \leqslant t < Lstst_j\,, \quad j \in J_i(t) \\ \frac{1}{(2-D_j/c_)}, & \text{otherwise} \end{cases} \tag{2}$$

where $D_j = |Lstst_j - t|$and *c* is a large enough constant

### 3.2. Priority decision of activity

In this proposed PSO, the new solution generation is based on activity priority. The priority $pr_{ij}$ is calculated based on as the inverse of the solution $X_{ij}^{new}$, where the $X_{ij}^{new}$ is the start processing times of activity *j* for new solution of particle *i.* The new solution (position) generated by particle is not used directly for schedule, but transformed to a priority. Therefore, a smaller $X_{ij}^{new}$ has a higher priority $pr_{ij}$ indicating the new solution suggests the activity *j* with high priority needs to be started earlier. Restated, a larger $pr_{ij}$ indicates that the activity *j* is possible selected earlier. The priority of activity is defined as listed in Eq. (3).

$$Pr_{ij}^{new} = \left(X_{ij}^{new}\right)^{-1} \tag{3}$$

### 3.3. Activity selection rule and resource consideration

The priority $pr_{ij}$ is extended to define $prcg_{ijt}$ under resource constraints and other consideration. For the resource constraints, each activity *j* is associated with a constraint state value $c_{ijt}$ (*j = 1,*…*,N*) for particle *i* at time *t*. The value of $c_{ijt}$ is decided to indicate the resources availability of remaining resources at time *t*, since some resources are already allocated to scheduled activities. Restated, the $c_{ijt}$ is a constraint state value based on the resource constraints at time *t*. This constraint state value is set to zero ($c_{ijt} = 0$), once activity *j* is prohibited to be selected at time *t* due to that the required resources are more than remaining resources. Otherwise, the constraint state value is set to one, $c_{ijt} = 1$. Restated, activity *j* at time *t* is a candidate activity when $c_{ijt}$ =1. Meanwhile, $J_i$ (*t*) is a set consisting of all candidate activities which can be assigned to run at time *t*. An activity is included in $J_i$ (*t*) based on the precedence among activities. An activity is then excluded from $J_i$ (*t*), if the activity is selected for execution.

The probability, $Pb_{ijt}$, of activity $j \in Ji(t)$ used for activity selection at time *t* is determined by $prcg_{ijt}$ (the product of $pr_{ij}$, $c_{ijt}$, and $g_{ijt}$). The $pb_{ijt}$ is a normalized $prcg_{ijt}$ as shown in Eq. (4). Moreover, the activity selection rule at time *t* is designed as shown in Eq. (5).

$$Prcg_{ijt} = pr_{ij} \times c_{ijt} \times g_{ijt}$$

$$Pb_{ijt} = \begin{cases} \frac{prcg_{jit}}{\sum_{k \in J_i(t) prcg_{ikt}}}, & \text{if } j \in J_i(t) \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

$$j' = \begin{cases} \underset{j \in J_i(t)}{\arg\max}\{prcg_{ijt}\}, & \text{if } q < q' \& \sum\limits_{j \in J_i(t)} c_{ijt} \neq 0 \\ S, & \text{if } q \geqslant q' \& \sum\limits_{j \in J_i(t)} c_{ijt} \neq 0 \end{cases} \tag{5}$$

The activity *j*' is selected from $J_i$ (*t*) according to the product of $pr_{ij}$ and $g_{ijt}$ values, for those activity *j* with $c_{ijt}$ = 1, then the scheduling process is processed until $J_i$ (*t*) is empty. The activity selection is decided by maximum $prcg_{ijt}$ when $q < q'$. Otherwise activity *j*′ is chosen according to *S* value. *S* is a random variable selected according to the probability distribution, $Pb_{ijt}$, as given in Eq. (4). The *q* is a random number uniformly distributed in [0, 1], and the *q*′ is a predefined parameter in [0, 1]. The $g_{ijt}$ value will be higher, if the time *t* is closing to the latest starting time of activity *j*, or the time *t* is over the latest starting time based on Eq. (2), then a higher $prcg_{ijt}$ will be induced. Accordingly, the activity *j* will have the higher probability ($Pb_{ijt}$) to be selected.

### 3.4. Solution representation and schedule generation

The position vector of each particle presents a solution; all the components of the position vector are the start times of all activities. If there are *N* activities in the scheduling system, the dimension of the space in which the particles move is *N*. Thus, the *N* components of the position vector present *N* start times of *N* activities. The schedule generation procedure for particle *i* of this investigation is summarized as Fig. 1.

In the new solution generation rule of this proposed PSO, the start time of activities ($X_{ij}$) and priority ($pr_{ij}$) of activities are updated according to Eqs. (1) and (3) respectively. Afterward, the particles construct new schedule using activity selection rule Eq. (5). Furthermore, the individual (or global) experience is replaced by the new solution when the new solution is better than existing individual (or global) experience.

## 4. Solution searching mechanisms

In this work, two solution space searching mechanisms are applied. One is called delay local search rule which is used to alter some activities in schedule while generating new solution. The other is the bidirectional scheduling rule which employs alternate forward with backward scheduling to expand the searching area in the solution space.

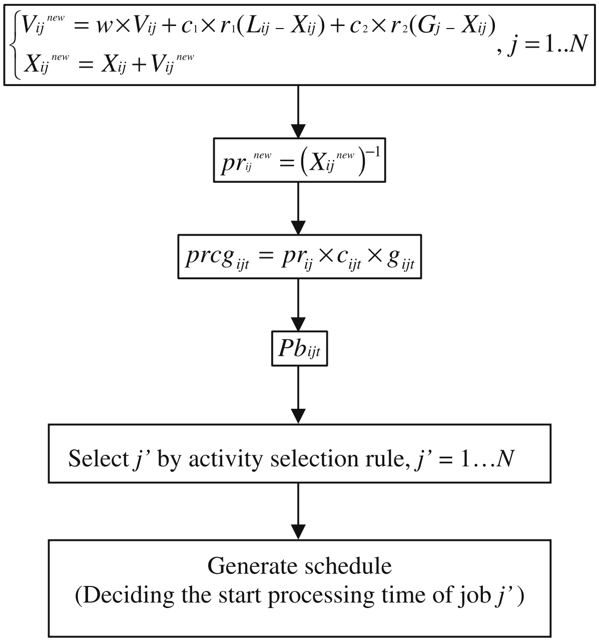

$\begin{cases} V_{ij}^{new} = w \times V_{ij} + c_1 \times r_1(L_{ij} - X_{ij}) + c_2 \times r_2(G_j - X_{ij}) \\ X_{ij}^{new} = X_{ij} + V_{ij}^{new} \end{cases}, j = 1..N$
$pr_{ij}^{new} = \left(X_{ij}^{new}\right)^{-1}$
$prcg_{ijt} = pr_{ij} \times c_{ijt} \times g_{ijt}$
$Pb_{ijt}$
Select *j'* by activity selection rule, *j'* = 1…*N*
Generate schedule
(Deciding the start processing time of job *j'* )

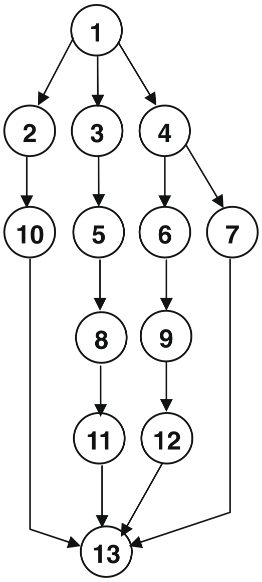
**Fig. 1.** The schedule generation procedure of the proposed PSO.

### 4.1. Delay local search rule

Generally, the allocation of resources in the scheduling process is to have as maximum activities processed as possible at any time, and the resulting schedule is sound. However, this way may cause some resources occupied by activities for a span of time, and lead some crucial activities to delay start at an inappropriate time for optimal scheduling (minimum *makespan*). Restated, this delay is mainly caused by the short of certain resources occupied by previous scheduled activities. For instance, if one activity requiring many resources at time $t$ is selected for execution, then some other activities requiring same resources will be prohibited from being selected to process for period of time. Accordingly, these delayed activities result in a larger *makespan* than that of the optimal solution. Hence, this study involves a mechanism of delay local search to delay some activities being selected from $J_i(t)$. This mechanism would leave remaining resources available and these available resources can be allocated to other activities properly. Restated, one activity can be processed later to let the other activities be processed ahead to yield global optimal solution under the resource constraints. This delay local search is indicated in the step 11 of proposed novel PSO in Table 1.

The delay local search rule as illustrated in Eq. (6). Using this delay local search, some resources requested by selected activity will be reserved for other activities. A activity is delayed if $q \leqq q_0$, where $q$ is the random number uniformly distributed in [0, 1], $q_0$ is a predefined small value ($0 < q_0 < 1$). The $q_0$ is defined as a *delay rate* to indicate the delay probability. Restated, a activity $j'$ selected from $J_i(t)$ will not be assigned to processor at time $t$ once the activity being selected and $q \leqq q_0$. This mechanism is similar to the mutation operation of genetic algorithm (GA).

$$\text{delay}(j', t) = \begin{cases} \text{true}, & \text{if } q \le q_0 \\ \text{false}, & \text{otherwise} \end{cases} \tag{6}$$



| Activity# | duration | Required Resources | | | |
|---|---|---|---|---|---|
| | | R 1 | R 2 | R 3 | R 4 |
| 1 | 0 | 0 | 0 | 0 | 0 |
| 2 | 3 | 0 | 0 | 9 | 0 |
| 3 | 9 | 0 | 7 | 0 | 0 |
| 4 | 4 | 9 | 0 | 0 | 2 |
| 5 | 10 | 10 | 3 | 3 | 10 |
| 6 | 9 | 0 | 0 | 8 | 4 |
| 7 | 2 | 2 | 6 | 1 | 0 |
| 8 | 4 | 8 | 0 | 8 | 4 |
| 9 | 4 | 0 | 0 | 4 | 0 |
| 10 | 10 | 0 | 0 | 8 | 0 |
| 11 | 7 | 10 | 0 | 0 | 0 |
| 12 | 7 | 0 | 2 | 5 | 6 |
| 13 | 0 | 0 | 0 | 0 | 0 |
| Available Resources | | 15 | 14 | 12 | 12 |

**Fig. 2.** Simulation case for 13 activities with precedence and resource constraints.

Fig. 2 depicts an example case for 13 activities with precedence and resource constraints. Figs. 3–6 all use the forward scheduling mechanism. Figs. 3 and 4 are the schedule results with and without delay local search applied, the resources allocation is also shown in Fig. 5. Fig. 6 lists the remaining available resources for each type of resources for each time slot with and without using the delay local search.

Suppose $S_i(t)$ is the set of activity scheduled by particle $i$ at time $t$. In the example of Fig. 2 and the resulting schedule in Fig. 3, $S_i$ (1) = {2, 3, 4} at time 1, the activity 2 will be finished at time 3. Activity 10 would be assigned for processing at time 4 according to the precedence, i.e., $S_i$ (4) = {3, 4, 10}. The candidate activities at time 5 are represented by the set $J_i$ (5) = {3, 6, 7, 10}. Nevertheless, the set of activity scheduled by particle $i$ at time 5 is $S_i$ (5) = {3, 7, 10} but $S_i$ (5) = {3, 6, 7, 10} since the total amount of R3 resources available is 12, which is insufficient for processing activities 6 and 10 concurrently. And, activity 6 and 10 must be mutual excluded at the same time. Hence, the activity 6 has to be delayed to start until activity 10 finish at time 13.

However, if the delay local search is applied appropriately at time 4, the activity 10 is excluded from being assigned for running. Accordingly, the set of the selected activities at time 5, $S_i$ (5) = {3, 6, 7} is possible, and the activity 6 can start earlier. Restated, activities 6 and 10 are not allowed to use resource R3 simultaneously, based on the resource requirement. Moreover, activities 5 and 6 need more resource R4 than system provided, hence, activities 5

| Time | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| P1 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 9 | 9 | 9 | 9 | 12 | 12 | 12 | 12 | 12 | 12 | 12 |
| P2 | 4 | 4 | 4 | 4 | 7 | 7 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 8 | 8 | 8 | 8 | 11 | 11 | 11 | 11 | 11 | 11 | 11 |
| P3 | 2 | 2 | 2 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |

**Fig. 3.** Scheduling result without using delay local search, with forward scheduling (*Makespan* = 39, 100 iterations, 10 particles, 3 processors: P1, P2, P3 are used).

| Time | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| P1 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | - | - | - | - | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 9 | 9 | 9 | 9 | 12 | 12 | 12 | 12 | 12 | 12 | 12 |
| P2 | 4 | 4 | 4 | 4 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 8 | 8 | 8 | 8 | 11 | 11 | 11 | 11 | 11 | 11 | 11 |
| P3 | 2 | 2 | 2 | - | 7 | 7 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |

**Fig. 4.** Scheduling result by using delay local search method with forward scheduling (*Makespan* = 34, 100 iterations, 10 particles, 3 processors: P1, P2, P3 are used).

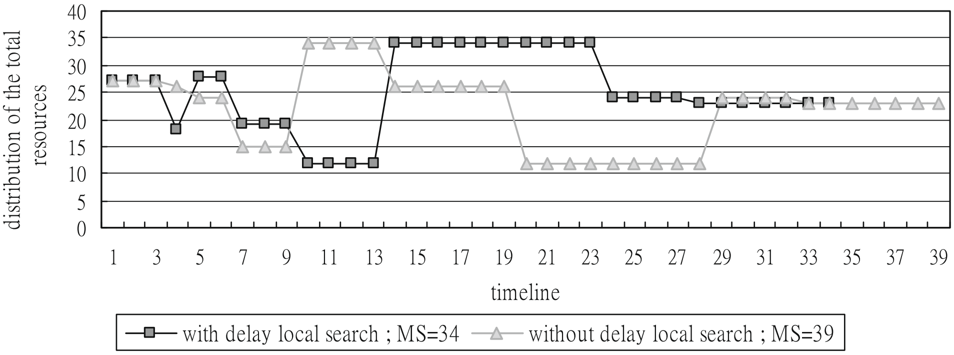


**Fig. 5.** The total distribution of the resources used with/without delay local search applied.

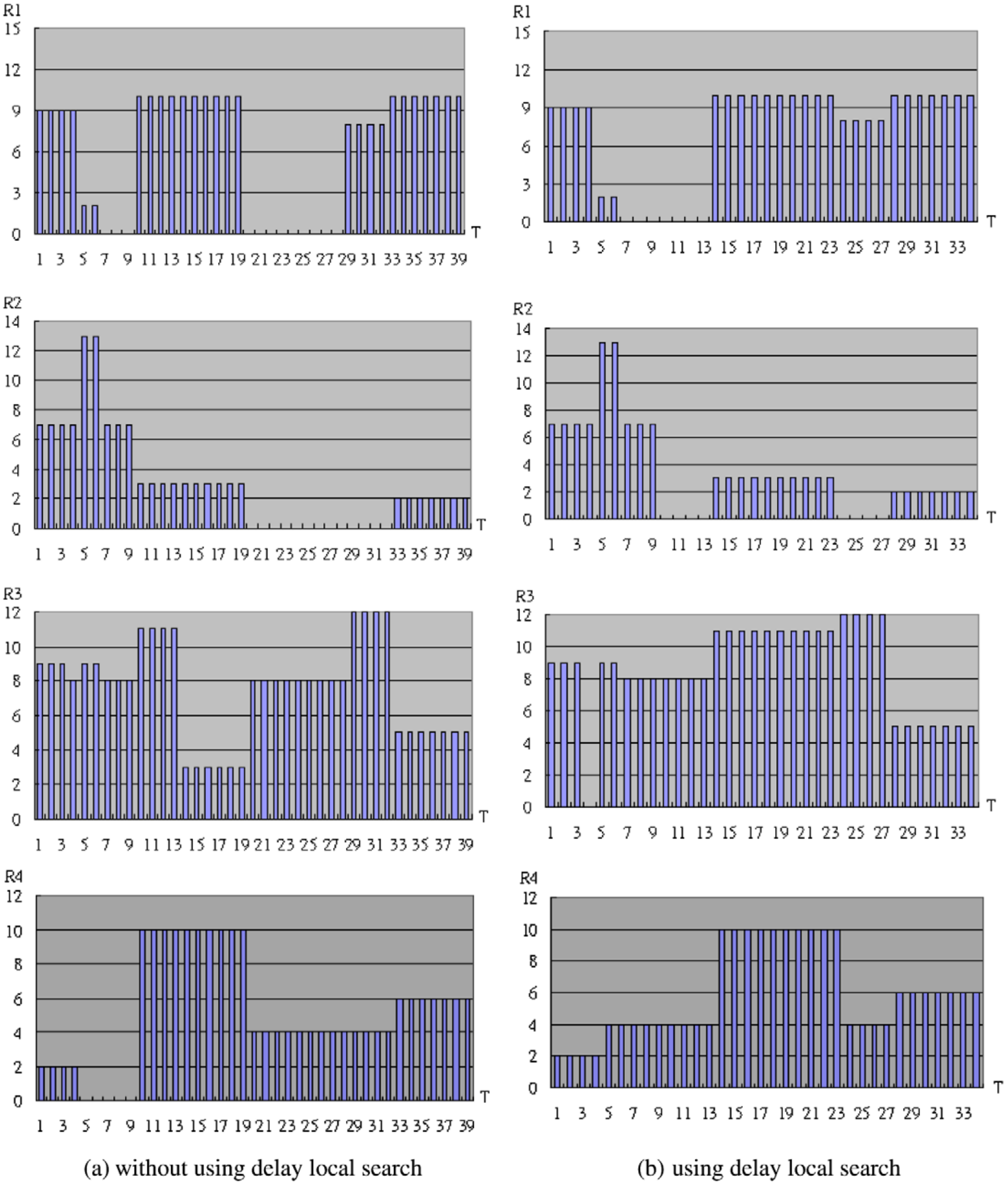


**Fig. 6.** The resources allocation for each type of resources.

and 6 are prohibited to execute at same time. Consequently, the application of delay local search increases the resource utilization and hence is able to yield shorter *makespan* solution. If the delay local search is activated at time 4 and put away the selection of activity 10, the *makespan* is 34 as shown in Fig. 4, otherwise, the *makespan* is larger than 34 as displayed in Fig. 3.

| Time | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| P1 | 4 | 4 | 4 | 4 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 8 | 8 | 8 | 8 | 11 | 11 | 11 | 11 | 11 | 11 | 11 |
| P2 | - | 2 | 2 | 2 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 9 | 9 | 9 | 9 | 12 | 12 | 12 | 12 | 12 | 12 | 12 |
| P3 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 7 | 7 |

**Fig. 7.** The Gantt chart of resulting schedule only by using backward scheduling (*Makespan* = 34, 100 iterations, 10 particles, 3 processors: P1, P2, P3 are used).

### 4.2. Bidirectional scheduling rule

The general scheduling scheme allots activities to processors to yield solution while satisfying all the required constraints such as resources, precedence and other constraints. Generally, the system has different resources allocation would cause diverse solutions. Hence, Li and Willis (1992) suggested applying bidirectional scheduling to discover other solutions, since the situations of resources requested by activities in forward and backward scheduling are different, the resulting schedules are different. The forward scheduling and backward scheduling would search for solutions in different searching area of the solution space.

Some scheduling cases are appropriate by forward scheduling, and some scheduling cases are suitable using backward scheduling. Hence, this study alternates forward scheduling with backward scheduling during iteration to increase the scheduling efficiency. In this study, two particle swarms are employed; one consists of forward scheduling particles and the other comprises of backward scheduling particles. Then, the forward scheduling particle works on the ordinary problem instance, and the backward one works on the reversed problem instance. Restated, the directed paths of the precedence graph are reversed. Forward and backward particles work separately on their own position and velocity matrices. Although the solutions generated by backward scheduling are not always better than the one by forward scheduling. However, the backward scheduling provides an opportunity to diversify the solutions and increase the opportunity to obtain the optimal solution.

The simulation results of the example case (illustrated in Fig. 2) scheduled by bidirectional scheduling without delay local search are displayed in Figs. 7–9. The Gantt charts of resulting schedule by the *forward/backward scheduling* without using delay local search are show in Figs. 3 and 7 respectively. And the allocation of the total resources consumed by the *forward/backward scheduling* without using delay local search is listed in Fig. 8. The usage of the individual resource allocated to both the forward scheduling and backward scheduling on the simulation case is displayed in Fig. 9. The *makespan* of the resulting schedule using backward scheduling is 34, and the *makespan* is larger than 34 for the forward scheduling as shown in Fig. 3. Restated, the optimal schedule can be obtained by backward scheduling.

## 5. Experimental examples and results

This study simulates the Single Mode Data Sets cases in PSPLIB [29] library, these cases consist of 30–120 activities. The simulation parameters used in this investigation are set as listed in Table 2. The feasible solutions (the position of particles) are maintained by every activity's start processing time during iteration. The objective of the studied algorithm is to find the schedule with minimal *makespan*. The simulation program is coded by C language and running at the PC with Pentinum4 3.4 GHz CPU.

The following simulations were applied to verify how many cases of PSPLIB library can be solved to yield optimal solution or lower bound solution by the proposed novel PSO. In PSPLIB library, the benchmark optimal or lower bound solution for each instance is also included. There is no processor constraint is given in PSPLIB. Hence, in the simulations, the number of processor is assumed sufficient for all instance cases.

There are 5 approaches are compared in this study and as listed in Table 3, they are proposed novel particle swarm optimization for scheduling (denoted by PSO+), PSO with bidirectional scheduling (indicated by PSO Bidirectional), PSO with delay local search (PSO Delay), traditional PSO (PSO), and ant colony optimization for scheduling as in (Chen et al., 2006; Chen et al., 2006) is denoted by ACO.

Figs. 10–13 demonstrate the simulation results of all 480 instances for 30, 60, 90 activities cases and 600 instances for 120 activities case. To evaluate the studied algorithm, each instance was run for 10, 100, and 1000 iterations. These figures show that the proposed novel PSO (denoted by PSO+, combining the critical path method, delay local search and bidirectional scheduling method) is able to find more optimal schedules than the other classes of PSO and ACO algorithms.

Table 4 illustrates the 30 activities example instance j301_6 in PSPLIB. Fig. 14 display the case j301_6 simulation results and total distributions of the resources used by the PSO and PSO+ methods.

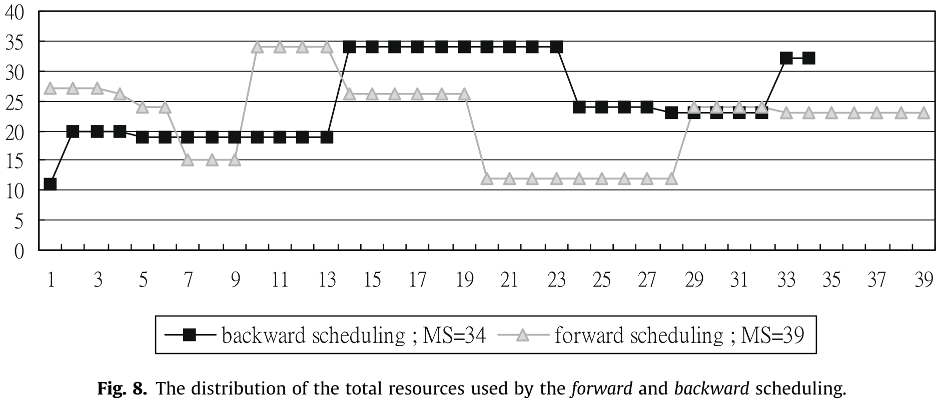
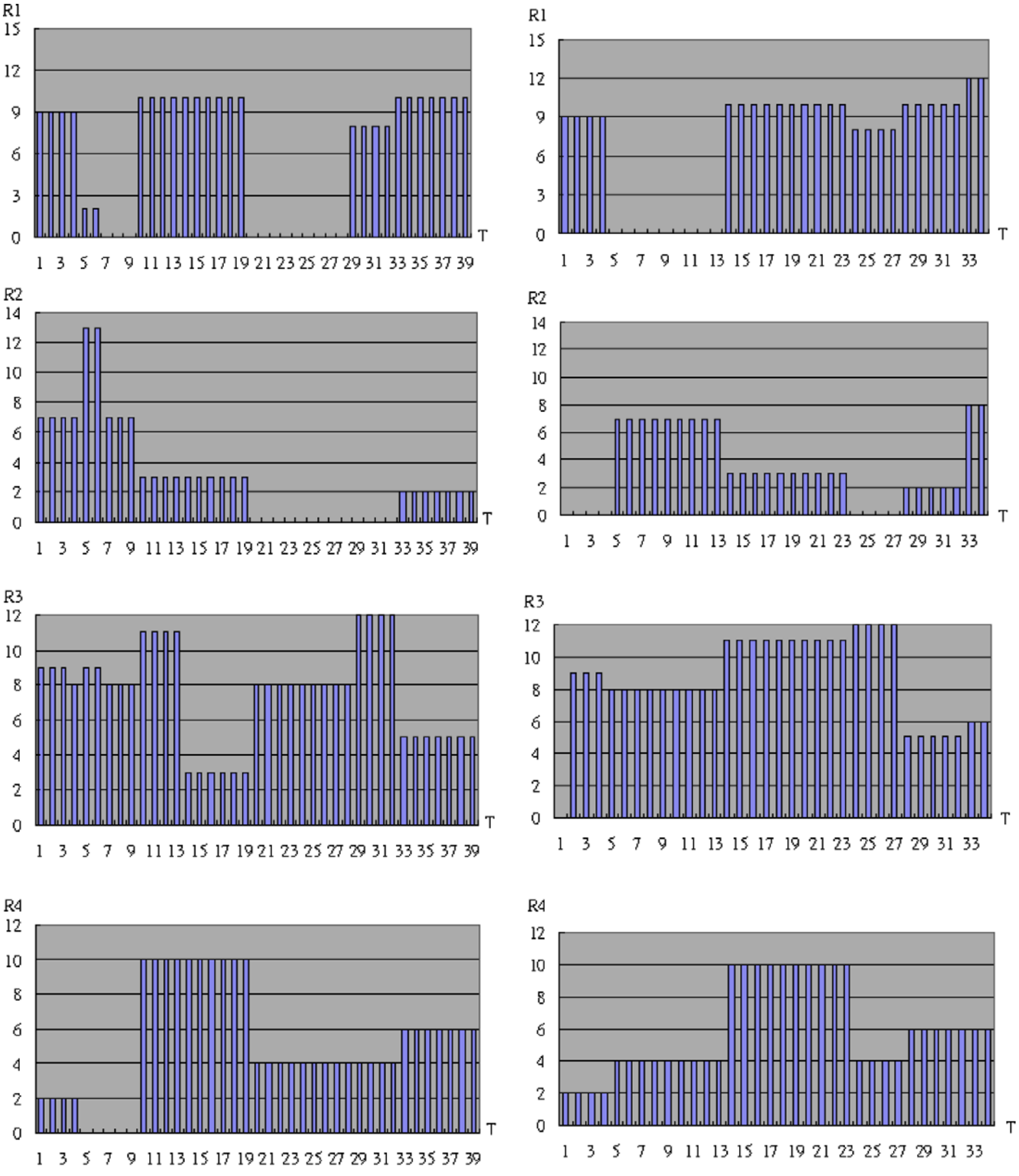


**Fig. 8.** The distribution of the total resources used by the *forward* and *backward* scheduling.



(a) using forward scheduling (b) using backward scheduling

**Fig. 9.** The resource allocation of each type of resources.

**Table 2**
Parameters setting.

| Parameters | Value |
|---|---|
| Number of particles: *M* | 10 |
| Initial positions of particles: *Xi* | Random value |
| Initial priority: $pr_{ij}$ | Random value |
| Initial velocity of particles: *Vi* | 0 |
| Inertia weight: *w* | 0.7 |
| Learning factors: $c_1$ and $c_2$ | 0.7 |
| Predefined parameter for delay rate: $q_0$ | 0.05 |
| Parameter in heuristic value determination: *c* | 50 |
| Parameter in activity selection: $q'$ | 0.95 |

**Table 3**
Five approaches setting.

| | Critical path method | Delay local search | Bidirectional scheduling |
|---|---|---|---|
| PSO+ | ○ | ○ | ○ |
| PSO bidirectional | ○ | × | ○ |
| PSO delay | ○ | ○ | × |
| PSO | × | × | × |
| ACO | ○ | × | × |

In this instance case, there are 4 types of resource, 30 activities and sufficient processors available are assumed. The *makespan* of scheduling result using PSO is 49 and the *makespan* of scheduling result by PSO+ is 48(optimal). Fig. 14a shows the total resource usage comparison by using PSO and PSO+ methods. Fig. 14b and c display the resulting schedules using traditional PSO and PSO+ respectively. Moreover, the simulation results of the 120 activities instance (j12022_4 in PSPLIB) using the PSO and PSO+ methods are demonstrated in Fig. 15. The simulation results of case j12022_4

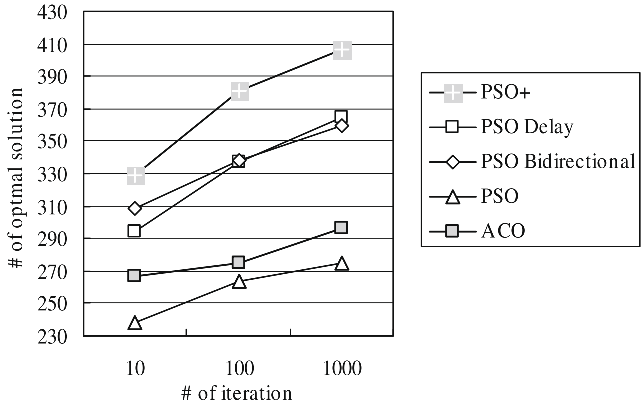


**Fig. 10.** The simulation results of using PSO+, PSO bidirectional, PSO delay, PSO, and ACO for 30 activities.

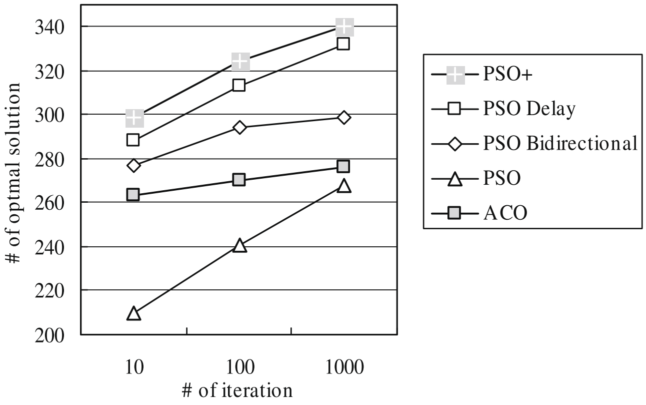


**Fig. 11.** The simulation results of using PSO+, PSO bidirectional, PSO delay, PSO, and ACO for 60 activities.

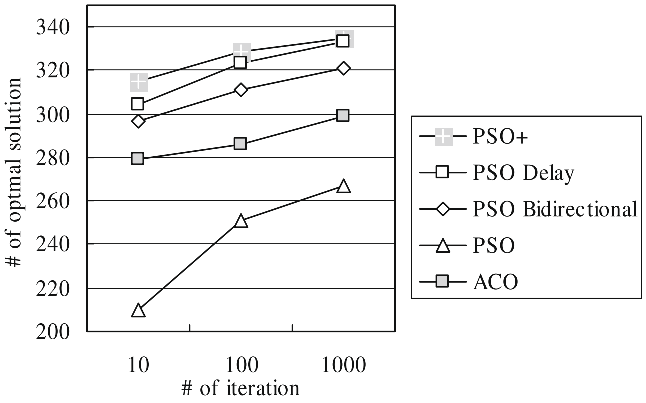


**Fig. 12.** The simulation results of using PSO+, PSO bidirectional, PSO delay, PSO, and ACO for 90 activities.

are similar to that of case j301_6. Restated, the PSO+ method can yield optimal *makespan* of the solution schedule.

The scheduling problems in PSPLIB have 30, 60, 90, and 120 activities cases. The 30, 60 and 90 activities problem cases each has 480 instances, 120 activities problem case has 600 instances. To further evaluate the studied algorithm, each instance was run for 1000 iterations; and each problem size case was tested 10 trials.

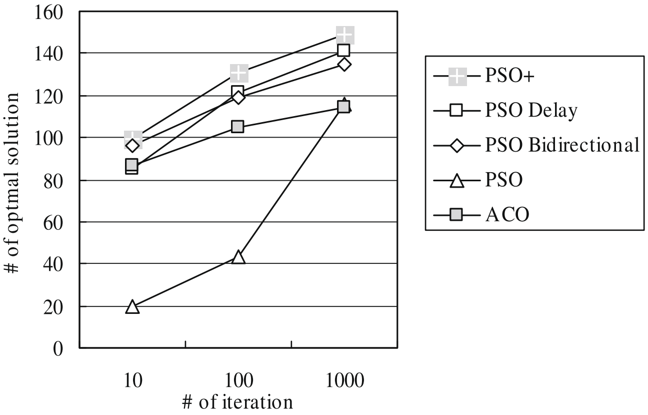


**Fig. 13.** The simulation results of using PSO+, PSO bidirectional, PSO delay, PSO, and ACO for 120 activities.

**Table 4**
30 activities case (j301_6) with precedence and resource requirement constraints.

| Activity# | Successors | | | Activity# | Duration | Required resources | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | R1 | R2 | R3 | R4 |
| 1 | 2 | 3 | 4 | 1 | 0 | 0 | 0 | 0 | 0 |
| 2 | 5 | 7 | 8 | 2 | 10 | 0 | 0 | 0 | 4 |
| 3 | 11 | | | 3 | 1 | 0 | 0 | 0 | 10 |
| 4 | 6 | 16 | | 4 | 9 | 4 | 0 | 0 | 0 |
| 5 | 15 | 23 | | 5 | 3 | 6 | 0 | 0 | 0 |
| 6 | 10 | 12 | | 6 | 1 | 3 | 0 | 0 | 0 |
| 7 | 9 | 14 | 25 | 7 | 7 | 0 | 4 | 0 | 0 |
| 8 | 13 | | | 8 | 1 | 0 | 0 | 0 | 2 |
| 9 | 24 | | | 9 | 4 | 10 | 0 | 0 | 0 |
| 10 | 22 | | | 10 | 10 | 0 | 0 | 0 | 2 |
| 11 | 14 | 16 | 24 | 11 | 6 | 0 | 0 | 10 | 0 |
| 12 | 13 | 21 | | 12 | 2 | 0 | 0 | 0 | 6 |
| 13 | 17 | 24 | 30 | 13 | 3 | 0 | 7 | 0 | 0 |
| 14 | 18 | | | 14 | 1 | 0 | 0 | 3 | 0 |
| 15 | 16 | 29 | | 15 | 3 | 0 | 0 | 0 | 6 |
| 16 | 19 | | | 16 | 1 | 0 | 0 | 10 | 0 |
| 17 | 18 | | | 17 | 3 | 0 | 0 | 0 | 7 |
| 18 | 20 | 31 | | 18 | 10 | 0 | 0 | 0 | 9 |
| 19 | 28 | | | 19 | 1 | 0 | 6 | 0 | 0 |
| 20 | 26 | | | 20 | 3 | 5 | 0 | 0 | 0 |
| 21 | 28 | | | 21 | 4 | 0 | 3 | 0 | 0 |
| 22 | 28 | | | 22 | 2 | 8 | 0 | 0 | 0 |
| 23 | 27 | | | 23 | 4 | 1 | 0 | 0 | 0 |
| 24 | 26 | 31 | | 24 | 2 | 3 | 0 | 0 | 0 |
| 25 | 30 | | | 25 | 4 | 0 | 9 | 0 | 0 |
| 26 | 29 | | | 26 | 6 | 0 | 0 | 0 | 7 |
| 27 | 30 | | | 27 | 9 | 0 | 0 | 0 | 7 |
| 28 | 31 | | | 28 | 2 | 0 | 0 | 0 | 5 |
| 29 | 32 | | | 29 | 1 | 0 | 0 | 9 | 0 |
| 30 | 32 | | | 30 | 1 | 0 | 0 | 9 | 0 |
| 31 | 32 | | | 31 | 9 | 0 | 0 | 4 | 0 |
| 32 | | | | 32 | 0 | 0 | 0 | 0 | 0 |
| | | | | Available resources | | 12 | 10 | 10 | 12 |

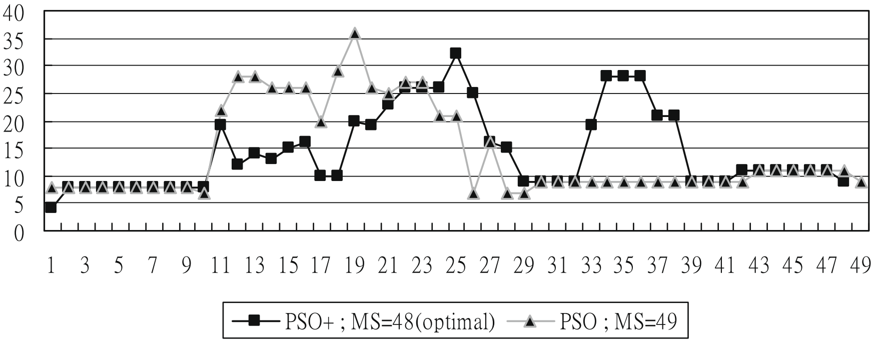
The assessment of different approaches is represented by the solution quality. The solution quality is commonly measured by the relative percentage deviation (*RPD*):

$$RPD = \frac{\sum^{\text{instances}}(100\% \times \frac{Obtained-Best}{Best})}{\text{instances}} \tag{7}$$

$$ARPD = \frac{\sum^{\text{trials}} RPD}{\text{trials}} \tag{8}$$

$$BRPD = \min\{RPD\} \tag{9}$$

In Eq. (7), the *Obtained* is the *makespan* of solution obtained by certain approach for certain instance, *best* is the optimum solution or the lower bound of this instance in the library. Therefore, *RPD* of



(a)

| Time | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| P1 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 3 | 11 | 11 | 11 | 11 | 11 | 11 | 13 | 13 | 13 | 19 | 22 | 22 | 28 | 28 | - | 17 | 17 | 17 | 18 | 18 | 18 | 18 | 18 | 18 | 18 | 18 | 18 | 18 | 20 | 20 | 20 | 26 | 26 | 26 | 26 | 26 | 26 | 29 |
| P2 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 6 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 14 | 16 | - | - | 24 | 24 | - | - | - | 30 | - | - | - | - | - | - | - | - | - | - | - | - | 31 | 31 | 31 | 31 | 31 | 31 | 31 | 31 | 31 | - |
| P3 | - | - | - | - | - | - | - | - | - | - | 8 | 12 | 12 | 15 | 15 | 15 | - | 9 | 9 | 9 | 9 | 25 | 25 | 25 | 25 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| P4 | - | - | - | - | - | - | - | - | - | - | 5 | 5 | 5 | 23 | 23 | 23 | 23 | 27 | 27 | 27 | 27 | 27 | 27 | 27 | 27 | 27 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| P5 | - | - | - | - | - | - | - | - | - | - | - | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| P6 | - | - | - | - | - | - | - | - | - | - | - | - | - | 21 | 21 | 21 | 21 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |

(b)

| Time | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| P1 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 3 | - | 8 | - | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 16 | 19 | 28 | 28 | 18 | 18 | 18 | 18 | 18 | 18 | 18 | 18 | 18 | 18 | 31 | 31 | 31 | 31 | 31 | 31 | 31 | 31 | 31 | 29 |
| P2 | - | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 5 | 5 | 5 | 13 | 13 | 13 | 17 | 17 | 17 | 27 | 27 | 27 | 27 | 27 | 27 | 27 | 27 | 27 | - | - | - | - | - | 25 | 25 | 25 | 25 | 30 | 20 | 20 | 20 | 26 | 26 | 26 | 26 | 26 | 26 | - |
| P3 | - | - | - | - | - | - | - | - | - | - | 6 | 12 | 12 | - | - | 23 | 23 | 23 | 23 | - | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 14 | - | - | - | - | 9 | 9 | 9 | 9 | 24 | 24 | - | - | - | - | - | - | - | - | - | - |
| Pr4 | - | - | - | - | - | - | - | - | - | - | - | - | - | 15 | 15 | 15 | - | - | 11 | 11 | 11 | 11 | 11 | 11 | 22 | 22 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| P5 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 21 | 21 | 21 | 21 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| P6 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |

(c)

**Fig. 14.** (a) The distribution of the total resources used by PSO and PSO+, (b) PSO scheduling result and (c) PSO+ scheduling result of case j301_6.

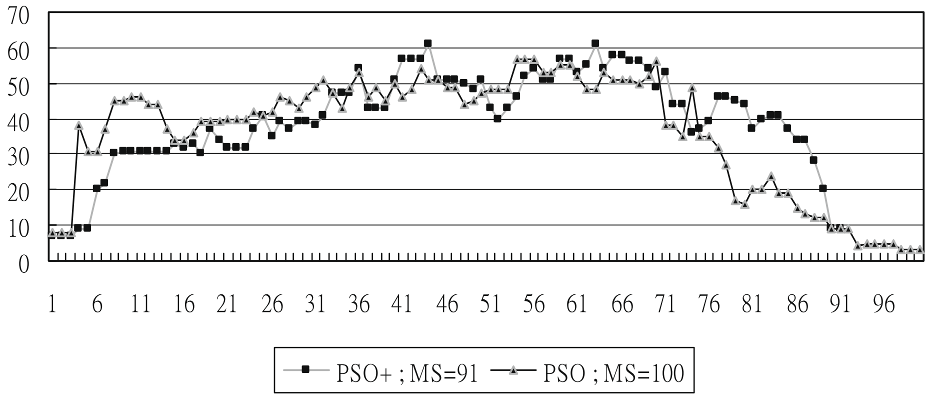


**Fig. 15.** The distribution of the total resources used by the PSO and PSO+ scheduling of case j12022_4.

**Table 5**
The comparison of *ARPD* among different approaches.

| Problem size/approach | PSO+ (%) | PSO bidirectional (%) | PSO delay (%) | PSO (%) | ACO (%) |
|---|---|---|---|---|---|
| 30 | **0.54** | 0.84 | 1.03 | 2.07 | 1.57 |
| 60 | **3.71** | 3.80 | 4.04 | 5.26 | 4.34 |
| 90 | **3.79** | 3.89 | 4.42 | 5.61 | 4.32 |
| 120 | **9.38** | **9.38** | 11.39 | 14.00 | 10.21 |
| Average | **4.36** | 4.48 | 5.22 | 6.73 | 5.11 |

**Table 6**
The copmparison of *BRPD* among different approaches.

| Problem size/approach | PSO+ (%) | PSO bidirectional (%) | PSO delay (%) | PSO (%) | ACO (%) |
|---|---|---|---|---|---|
| 30 | **0.31** | 0.58 | 0.62 | 1.33 | 1.29 |
| 60 | **3.20** | 3.40 | 3.51 | 4.23 | 3.85 |
| 90 | **3.45** | 3.52 | 3.98 | 4.71 | 3.88 |
| 120 | **8.55** | 8.57 | 10.35 | 12.17 | 9.25 |
| Average | **3.88** | 4.01 | 4.62 | 5.61 | 4.57 |

**Table 7**
The comparison of *ARPD* among different delay rates.

| Problem size/delay rate | 0.05 (%) | 0.01 (%) | 0.005 (%) | 0.001 (%) |
|---|---|---|---|---|
| 30 | **0.301** | 0.347 | 0.368 | 0.433 |
| 60 | 3.290 | **3.157** | 3.186 | 3.237 |
| 90 | 3.711 | 3.467 | **3.443** | 3.455 |
| 120 | 9.633 | 8.740 | 8.663 | **8.630** |
| Average | 4.234 | 3.928 | **3.915** | 3.939 |

**Table 8**
The comparison of *BRPD* among different delay rates.

| Problem size/delay rate | 0.05 (%) | 0.01 (%) | 0.005 (%) | 0.001 (%) |
|---|---|---|---|---|
| 30 | **0.115** | 0.161 | 0.184 | 0.215 |
| 60 | 2.950 | **2.845** | 2.854 | 2.895 |
| 90 | 3.408 | 3.182 | 3.179 | **3.167** |
| 120 | 8.907 | 8.101 | 8.009 | **7.965** |
| Average | 3.845 | 3.572 | **3.556** | 3.560 |

certain problem size is calculated by averaging the difference between *Obtained* and *best* of all instances. Since each problem size case was simulated 10 trials, an average relative percentage deviation (*ARPD*) is defined as shown in Eq. (8) and a best relative percentage deviation (*BRPD*) is defined as listed in Eq. (9).

Table 5 lists the comparison of *ARPD* among different approaches for different scale problems. For example, the *ARPD*, an average of 4.36% and a maximum of 9.38% were achieved using proposed PSO+. Meanwhile, the comparison of *BRPD* among different approaches is given in Table 6.

According to Tables 5 and 6, the experiment results on different problem sizes by proposed PSO+ are better than that by other approaches.

Moreover, to verify the effect on the solution of using the proposed delay local search, different delay rates were investigated in this work. The delay rate is similar to the mutation rate in GA method. The *ARPD* and *BRPD* of the simulation results using various delay rates are shown in Tables 7 and 8.

## 6. Conclusions and discussion

This study proposes a novel PSO scheme to solve precedence and resource-constrained scheduling problems. A critical path method is applied to facilitate the new activity selection in Eqs. (4) and (5). Moreover, a delay local search mechanism (Eq. (6)) is utilized to escape from local optimum solution. Furthermore, this investigation applied bidirectional scheduling mechanism of alternate forward and backward scheduling to expand the searching area in the solution space. The simulation results reveal that this improved novel PSO is effective and efficient to solve the class of PSPLIB scheduling problems. The number of optimal solutions yielded by proposed novel PSO is obvious superior to other traditional methods such as ACS and PSO as shown in Figs. 10–13. The more optimal solutions found as iteration increased. Especially, for the problem size of 30 activities instances, the proposed method is able to find 450 optimal solutions in 480 instances (93.75%) when 10,000 iterations tested. Meanwhile, the PSO+ approach derived form PSO outperforms ACO approach in PSPLIB problems. It can be seen that the PSO+ integrated with appropriate search methods in this study can effectively improve the algorithm's efficiency as listed in Tables 5 and 6.

Moreover, a small value of the delay rate is suggested to find sound solution quality. Essentially, delay local search is to stir a small disturbance in solution space. However, a small fixed delay rate used in large scale problem would cause large disturbance since large scale problem has large amount of solution population. Hence, to yield small *PRD,* the larger scale problem requires the smaller delay rate as proved in Tables 7 and 8.

Furthermore, some other important features of this study are summarized below.

1. The variation of resources usage is an important factor in the scheduling algorithms. Therefore, make efficient use of resources shorten the resulting schedule *makespan*.
2. The suggested delay local search makes some determined activities delayed, and is able to have the better resource utilization of the system. Hence, the order of processing activities is changed and therefore yields the resulting optimal solution.
3. Additionally, bidirectional scheduling alternatively applying forward and backward scheduling mechanism can further increase resource utilization efficiency while searching for the optimal solution in different area in the solution space. Therefore, optimal solution can be yielded.
4. One of the important characteristics about the scheduling algorithms is its efficiency, the computation complexity. The execution time of this improved PSO is proportional to the number of particles *M* and activities *N*. For each particle, a solution requires *N* executions of activity selection rule, and activity selection rule would calculate the probability of selection for each activity (also *N* executions). Restated, the computation complexity of $O(M \times N^2)$ for each iteration is provided.

This study mainly studies the scheduling problems with precedence and resource constraints, and the processors are assumed to be identical. More complex conditions or situations should be further considered such as the setup time between activities of a certain machine, or if there is communication cost between two activities which are processing in different processors. Moreover, in a dynamic situation, there may be some emergency activities arriving at a certain time or changing the resources available and requirement. Meanwhile, the design of heuristic value is also possible to be improved for better solution.

## References

Bouleimen, K., & Lecocq, H. (2003). A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version. *European Journal of Operational Research, 140*(2), 268–281.

Buyya, R., Abramson, D., & Giddy, J. (2000). Nimrod/G: An Architecture for a Resource Management and Scheduling System in a Global Computational Grid. *The Fourth International Conference on High-Performance Computing in the Asia-Pacific Region*, 1, 283.

Chang, S. C. (2002). A new aircrew-scheduling model for short-haul routes. *Journal of Air Transport Management, 8*(4), 249–260.

Chen, R. M., Lo, S. T., Wang, C. J., & Wu, C. L. (2006). *Multiprocessor system scheduling with precedence and resources constraints by ant colony system*. 2006 ICS Conference.

Chen, T., Zhang, B., Hao, X., & Dai, Y. (2006). Task scheduling in grid based on particle swarm optimization. *Parallel and distributed computing, 2006. ISPDC '06. The fifth international symposium on* (pp. 238–245).

Chen, R. M., Lo, S. T., & Huang, Y. M. (2007). Combining competitive scheme with slack neurons to solve real-time job scheduling problem. *Expert Systems with Applications, 33*(1), 75–85.

Dorigo, M., & Gambardella, L. M. (1997). Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation, 1*(1), 53–66.

Fleming, P. J., & Fonseca, C. M. (1993). Genetic algorithms in control systems engineering: a brief introduction. *IEE colloquium on genetic algorithms for control systems engineering* (pp. 1/1–1/5).

Glover F. (1989). Tabu Search – Part I. *ORSA JOURNAL ON COMPUTING*, 1(3), 190–206.

Holland, John H. (1987). Genetic algorithms and classifier systems: foundations and future directions. *Proceedings of the second international conference on genetic algorithms on genetic algorithms and their application*, 82–89.

Hopfield, J. J., & Tank, D. W. (1985). Neural computation of decision in optimization problems. *Biological Cybernetics, 52*, 141–152.

Huang, Y. M., & Chen, R. M. (1999). Scheduling multiprocessor job with resource and timing constraints using neural network. *IEEE Transactions on System, Man and Cybernetics. Part B, 29*(4), 490–502.

Jalilvand, A., Khanmohammadi, S., & Shabaninia, F. (2005). Scheduling of sequence-dependant jobs on parallel multiprocessor systems using a branch and bound-based Petri net. In *Emerging technologies, proceedings of the IEEE symposium* (pp. 334–339).

Kennedy, J., & Eberhart, R. C. (1995). Particle swarm optimization. In *Proceedings IEEE international conference on neural networks* (Vol. 4, pp. 1942–1948).

Kirkpatrick, S., Gelatt, C. D., Jr., & Vecchi, M. P. (1983). Optimization by Simulated Annealing. *Science, 220*(4598), 671–680.

Li, C., Bettati, R., & Zhao, W. (1997). Static priority scheduling for ATM networks. In *18th IEEE real-time systems symposium (RTSS '97)* (pp. 264–273).

Li, K. Y., & Willis, R. J. (1992). An iterative scheduling technique for resource-constrained project scheduling. *European Journal of Operational Research, 56*, 370–379.

Liu, Z., & Wang, H. (2005). GA-based resource-constrained project scheduling with the objective of minimizing activities' cost. *Lecture Notes in Computer Science, 3644*, 937–946.

Luo, X., Wang, D., Tang, J., & Tu, Y. (2006). An improved PSO algorithm for resource-constrained project scheduling problem, intelligent control and automation, 2006. In *The sixth world congress on WCICA 2006* (Vol. 1, pp. 3514–3518).

Lupetti, S., & Zagorodnov, D. (2006). Data popularity and shortest-job-first scheduling of network transfers. *International Conference on Digital Telecommunications* (ICDT'06), p. 26.

Mathaisel, D., & Comm, C. (1991). Course and classroom scheduling: An interactive computer graphics approach. *Journal of Systems and Software, 15*, 149–157.

Merkle, D., Middendorf, M., & Schmeck, H. (2002). Ant colony optimization for resource-constrained project scheduling. *IEEE Transactions on Evolutionary Computation, 6*(4), 333–346.

Ohki, M., & Morimoto, A., & Miyake, K. (2006). Nurse Scheduling by using cooperative GA with efficient mutation and mountain-climbing operators. In *Third international IEEE conference on intelligent systems* (pp. 164–169).

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

Selman, B., Kautz, H. A., & Cohen, B. (1994). Noise Strategies for Improving Local Search. *Proceedings of the twelfth national conference on artificial intelligence, 1*, 337–343.

Shaw, K. J., Nortcliffe, A. L., Thompson, M., Love, J., Fleming, P. J., & Fonseca, C. M. (1999). Assessing the performance of multiobjective genetic algorithms for optimization of a batch process scheduling problem. In *Proceedings of the 1999 congress on evolutionary computation, CEC 99* (Vol. 7, p. 45).

Simeonov, S., & Simeonovova, J. (2002). Simulation scheduling in food industry application. Mathematical and statistical methods. *Food Processing and Preservation, 20*(1), 31–37.

Thomas, P. R., & Salhi, S. (1998). A Tabu search approach for the resource-constrained project scheduling problem. *Journal of Heuristics, 4*(2), 123–139.

Zhai, X., Tiong, R. L. K., Bjornsson, H. C., Chua, D. K. H. (2006). A simulation-ga based model for production planning in precast plant. In *Proceedings of the 38th conference on winter simulation winter simulation conference* (pp. 1796–1803).

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