Hindawi Publishing Corporation
Abstract and Applied Analysis
Volume 2011, Article ID 589862, 20 pages
doi:10.1155/2011/589862

Research Article

Project Scheduling Heuristics-Based Standard PSO for Task-Resource Assignment in Heterogeneous Grid

Ruey-Maw Chen and Chuin-Mu Wang

Department of Computer Science and Information Engineering, National Chin-Yi University of Technology, No. 35, Lane 215, Section 1, Chung-Shan Road, Taiping, Taichung 411, Taiwan

Correspondence should be addressed to Ruey-Maw Chen, raymond@mail.ncut.edu.tw

Received 13 October 2010; Revised 31 December 2010; Accepted 2 January 2011

Academic Editor: Nobuyuki Kenmochi

The task scheduling problem has been widely studied for assigning resources to tasks in heterogeneous grid environment. Effective task scheduling is an important issue for the performance of grid computing. Meanwhile, the task scheduling problem is an NP-complete problem. Hence, this investigation introduces a named "standard" particle swarm optimization (PSO) metaheuristic approach to efficiently solve the task scheduling problems in grid. Meanwhile, two promising heuristics based on multimode project scheduling are proposed to help in solving interesting scheduling problems. They are the best performance resource heuristic and the latest finish time heuristic. These two heuristics applied to the PSO scheme are for speeding up the search of the particle and improving the capability of finding a sound schedule. Moreover, both global communication topology and local ring communication topology are also investigated for efficient study of proposed scheme. Simulation results demonstrate that the proposed approach in this investigation can successfully solve the task-resource assignment problems in grid computing and similar scheduling problems.

1. Introduction

Grid computing has been widely applied many applications, such as drug discovery, economic forecasting, seismic analysis, and back-office data processing in support of e-commerce and web services. A grid is a collaborative environment in which one or more tasks can be submitted without knowing where the resources are or even who owns the resources [1]. Meanwhile, a grid environment also provides a computing service infrastructure in the cloud. There are often plenty of tasks needing to be dealt with since an application in grid are usually decomposed into many tasks; it is necessary to efficiently assign and allocate

adequate resources for tasks from anywhere. Restated, tasks assigned to inadequate resources always cause application completion time increases. Therefore, the task assignment plays a significant role in the issue of resource allocation and scheduling in grid. In [2], Chen et al. have solved the task scheduling problem in a distributed computing system as in [3], which is without the consideration of resource heterogeneity and the dependence between tasks. Then Chen et al. proposed a new task scheduling model to reflect the actual grid environment with more details, and solved it based on particle swarm optimization (PSO).

Many different task scheduling problems such as assignment, job-shop, flow-shop, vehicle routing, and other scheduling problems have been studied intensively. The studied grid task scheduling problem in this work comes from the task-resource assignment problem [2] which is much more complicated than the above-stated classic task scheduling problems. Restated, a grid application is a task scheduling problem involving partially ordered tasks and distributed heterogeneous resources, and can be represented by a directed acyclic graph (DAG) [4–6]. The scheduling target is to find optimal task-resource assignment and hence minimize application completion time. Most scheduling problems are confirmed to be NP-complete. Thus, many researchers have devoted their efforts to solving task scheduling problems. The exact algorithm such as branch-and-bound method [7] is able to find the optima of the scheduling problem. However, the execution time required is impractical as the number of tasks and resources increases. Hence, many different schemes have been presented for solving scheduling problems. Chen et al. [8] combined a competitive scheme with slack neurons into Hopfield neural networks to solve multiprocessor real-time job scheduling problems. Sandnes [9] presented a stochastic approach of employing randomization in the scheduling of tasks in multiobjective scheduling problems. An artificial immune-system based scheme was proposed to solve the dynamic economic dispatch problem of generating units [10]. Comparatively, several metaheuristics such as genetic algorithm (GA) [11], simulated annealing algorithm (SA) [12], tabu search (TS) [13, 14], ant colony optimization (ACO) [15], and the particle swarm optimization [16] have been effectively proposed for solving these difficult problems. Oh and Wu [17] presented a multiobjective genetic algorithm, which aims to minimize the number of processors required and the total tardiness of tasks. Liu and Wang [18] solved the resource-constrained project scheduling problem of minimizing activities' cost based on GA. And a thermal generating unit's commitment scheduling problem was studied by a modified GA [19]. A task scheduling problem with the involvement of the processor in communication was investigated and a GA-based scheduling heuristic was proposed to solve the problem by Sinnen et al. [20]. In [21], a derivative of simulated annealing algorithm with the consideration of the specificity of the solution space was proposed to solve the resource-constrained scheduling problem. Meanwhile, simulated annealing was applied to the berth-scheduling problem to find near-optimal solutions [22]. Tabu search is an approach to prevent the search from trapping into the local minimum, and it has been applied to solving a single machine scheduling problem with distinct due windows to minimize total weighted earliness and tardiness [23] as well as job-shop scheduling [24]. Using ACO to solve multiprocessor system scheduling with precedence and resources constraints was proposed in [25]. In [26], an ACO algorithm was used for solving a dynamic regional nurse-scheduling problem. Moreover, many PSO-based schemes were proposed for solving a variety of scheduling-related problems including process planning, production scheduling [27], job-shop scheduling [28, 29], project scheduling [30], call center scheduling [31], flow-shop scheduling [32], and others [33, 34].

In light of different scheme development, PSO is a promising metaheuristic approach for solving diverse task scheduling problems as well as other application problems. The

particle swarm optimization (PSO) was first proposed by Kennedy and Eberhart [16]. In PSO, a swarm of particles spread in the search space and the position of a particle presents a solution. Each particle would move to a new position decided with the individual experience and the global experience heading toward the global optimum. However, many variations of PSO have been studied, and one of them was named "standard" PSO, proposed by Clerc and Kennedy [35] indicating how PSO can be significantly improved. Hence, this investigation aims at enhancing the "standard" PSO for solving the interesting task scheduling in grid. Each particle represents a possible task schedule corresponding to a task-resource assignment graph (T-RAG) and regards the longest path of the task-resource assignment graph as fitness value. Hence, this investigated task scheduling problem became an optimal task-resource assignment graph selection problem.

To increase the efficiency of "standard" PSO, heuristics can be used as an aid for problem solving. Thus, this study enhances "standard" PSO by introducing additional heuristics to solve the task scheduling problem for minimizing task completion time. Nevertheless, a heuristic is usually problem specific, hence different heuristics were surveyed and evaluated. Least total resource usage (LTRU) and shortest feasible mode (SFM) heuristics are frequently applied for determining operating mode for multimode project scheduling problems. Greatest rank positional weight (GRPW), latest finish time (LFT), latest start time (LST), minimum slack (MSLK), and most total successors (MTS) heuristics are commonly used for deciding tasks' priority in project scheduling problems. Two intuitional heuristics were then proposed for speeding up the PSO's search when solving investigated task scheduling problems. They are the best performance resource (BPR) heuristic and the latest finish time (LFT) heuristic based on multimode project scheduling problems; they are herein called project scheduling heuristics.

Moreover, the performance of the proposed PSO scheme with different swarm communication topology is also evaluated. Restated, global communication and local communication topologies for obtaining the global experience are analyzed and compared. Finally, the experiment results indicate that the scheme proposed in this work is effective for solving similar class task scheduling problems.

This article is organized as follows. Section 2 introduces the task scheduling problem. The traditional PSO is described in Section 3. Section 4 illustrates application of the PSO to the task scheduling problem, and introduces the additional heuristics proposed in this study. The simulations are presented in Section 5. Finally, Section 6 presents the conclusions.

2. The Task Scheduling Problem

Most grid applications usually involve partially ordered tasks and heterogeneous resources distributed in grid. Hence, the studied new task scheduling problem addresses precedence considerations and resource heterogeneity in grid environment. A simple example is given to illustrate the complexity and difficulty of the investigated scheduling problem; suppose a grid application is decomposed into 5 partially ordered tasks (with different workloads) as shown in Figure 1—two heterogeneous resources (with different abilities) in the grid environment. Therefore, tasks assigned to different resources would require different processing times to run. Additionally, involvement of communication costs (such as data transfer) for partially ordered tasks is considered. Meanwhile, assume that resource M2 has better performance than resource M1, but resource M1 has higher bandwidth than resource M2. Thus, communication time depends on the communication cost and the minimum

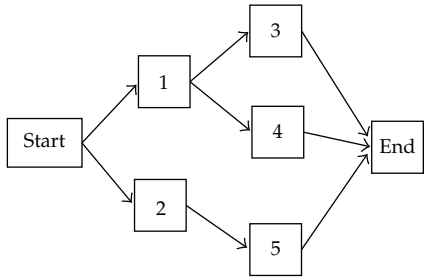


Figure 1: An example of a studied task scheduling problem.

Table 1

Task no.	Resource	Execution time
Task 1	M1	22.22
	M2	10
Task 2	M1	50
	M2	22.5
Task 3	M1	40
	M2	18
Task 4	M1	88.88
	M2	40
Task 5	M1	11.11
	M2	5

Table 2

Task no.	Successors	Communication time
Task 1	Task 3	2
	Task 4	5.5
Task 2	Task 5	4

bandwidth between resources is computed if partially ordered tasks are not assigned to the same resources for processing. Execution time and communication time required for this example are assumed as shown in Tables 1 and 2.

Many task scheduling result cases exist for this simple problem. For instance, consider the following cases.

Case 2.1. Tasks 1 and 3 are assigned to resource M1, and tasks 2, 4, and 5 are distributed to M2. Meanwhile, the processing order of tasks for resource M1 is $1 \rightarrow 3$; the processing order of tasks for resource M2 is $2 \rightarrow 5 \rightarrow 4$. There are three process paths to complete application, and therefore three corresponding time costs (see Table 3).

It requires 62.22 (22.22 + 40) to process tasks on path 1 since task 1 and task 3 are on the same resource without data transfer. Nevertheless, task 4 on path 2 has to wait until both path 3 finished (22.5 + 5) and task 1 finished including data transfer (22.22 + 5.5). Hence, the completion time of path 2 is 67.72 (27.72 + 40). Accordingly, the completion time of application is the maximum time cost (67.72) corresponding to the largest cost path. Additional scheduling result cases are also given as follows.

Table 3

Paths	Time cost
(1): start → Task1/M1 → Task3/M1 → end	22.22 + 40 = 62.22
(2): start → Task1/M1 → Task4/M2 → end	max(22.5 + 5, 22.22 + 5.5) + 40 = 67.72
(3): start → Task2/M2 → Task5/M2 → end	22.5 + 5 = 27.5

Table 4

Paths	Time cost
(1): start → Task1/M1 → Task3/M1 → end	22.22 + 40 = 62.22
(2): start → Task1/M1 → Task4/M2 → end	max(22.22 + 5.5, 10) + 40 = 67.72
(3): start → Task2/M2 → Task5/M2 → end	max(22.5, 22.22 + 5.5 + 40) + 5 = 72.72

Table 5

Paths	Time cost
(1): start → Task1/M2 → Task3/M1 → end	max(10 + 2, 50) + 40 = 90
(2): start → Task1/M2 → Task4/M2 → end	10 + 40 = 50
(3): start → Task2/M1 → Task5/M2 → end	max(50 + 4, 10 + 40) + 5 = 59

Table 6

Paths	Time cost
(1): start → Task1/M2 → Task3/M1 → end	max(10 + 2, 50) + 40 = 90
(2): start → Task1/M2 → Task4/M2 → end	max(10, 50 + 4 + 5) + 40 = 99
(3): start → Task2/M1 → Task5/M2 → end	max(50 + 4, 10) + 5 = 59

Case 2.2. With the same task-resource assignment and processing order of tasks for resource M1 as Case 2.1, but the processing order of tasks for resource M2 is 2 → 4 → 5. The completion time of application is 72.72 (see Table 4).

Task 4 on path 2 can only be executed when both task 1 is finished plus data transferred (22.22 + 5.5 = 27.72) and task 2 on M2 is done (10). Hence, task 4 starts to execute at 27.72 and completes at 67.72 (27.72 + 40). In path 3, task 2 has to wait until task 2 finishes (22.5) and task 4 is completed (67.72). Therefore, time cost for path 3 is 72.72 (67.72 + 5).

Case 2.3. With different task-resource assignment and processing order of tasks for resources as Case 2.1. Assume that tasks 2 and 3 are assigned to resource M1, and tasks 1, 4 and, 5 are allocated to resource M2. Meanwhile, the processing order of tasks for resource M1 is 2 → 3; the processing order of tasks for resource M2 is 1 → 4 → 5. Then, the completion time of application is 90 which can be determined (see Table 5).

Case 2.4. With the same task-resource assignment and processing order of tasks for resource M1 as Case 2.4. And, the processing order of tasks for resource M2 is 1 → 5 → 4. Then, the completion time of application is 99 which can be determined (see Table 6).

However, there are still many other scheduling result cases of this simple example. Hence, the studied task scheduling problem in grid is indeed complicated. To obtain the best schedule with minimal completion time of application, the task-resource assignment and the processing order (priority) determination of tasks have to be decided accurately.

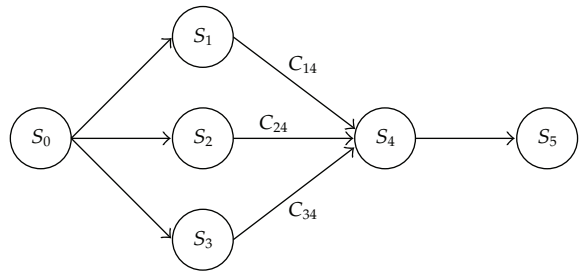


Figure 2: Directed acyclic graph (DAG).

2.1. The Representation of the Task Scheduling Problem

In this interesting scheduling problem [2], there are $K + 2$ tasks $S = \{S_0, S_1, \ldots, S_K, S_{K+1}\}$, and N heterogeneous resources $M = \{M_1, M_2, \ldots, M_{N-1}, M_N\}$, where S_0 and S_{K+1} are pseudotasks for scheduling indicating the start and finish states. The workload of task S_u is $w(S_u)$, and the computing ability of resource M_i is $a(M_i)$, once the S_u is assigned to M_i, then the S_u's execution time $E_u = w(S_u)/a(M_i)$ can then be obtained.

Hence, this work focuses on how to assign these K tasks (ignoring the pseudotasks) to N resources for gaining the shortest completion time of scheduling (i.e., S_K's finish time).

Herein, with the precedence between tasks addressed, the task scheduling problem is expressed as a directed acyclic graph (DAG) as in Figure 2. An arrow from S_u to S_v indicates that S_u is the immediate predecessor of S_v, and the connection weight is the communication cost c_{uv} for S_u transferring data to S_v, where pseudotasks are without data transfer. If the S_u, and S_v are assigned to different resources M_i and M_j, respectively, the bandwidth between M_i and M_j is $bw(M_i, M_j)$, $M_i, M_j \in M$, then the communication time $T_{uv} = c_{uv} / bw(M_i, M_j)$ can be determined. Restated, the S_v can't start until the S_u has finished with execution time E_u and communication time T_{uv}.

In [19], the $bw(M_i, M_j)$ is derived from a predefined table as the $N \times N$ matrix **BW** as follows:

$$\mathbf{BW} = \begin{bmatrix} \infty & bw(M_1, M_2) & \cdots & bw(M_1, M_N) \\ bw(M_2, M_1) & \infty & \cdots & bw(M_2, M_N) \\ \vdots & \vdots & \ddots & \vdots \\ bw(M_N, M_1) & bw(M_N, M_2) & \cdots & \infty \end{bmatrix}, \tag{2.1}$$

where $bw(M_i, M_j) = \infty$ when $i = j$, that is, the communication time $T_{uv} = 0$ once the S_u, S_v are assigned to the same resource.

In a real situation, the bandwidth between two resources is limited by the resource which has lower bandwidth. Therefore, the approach for deriving the $bw(M_i, M_j)$ is modified instead of using the matrix **BW** , in this study. Restated, resource $M_i$has its own bandwidth $bwl(M_i)$, then the bandwidth between two resources $M_i$and M_j can be treated as $bw(M_i, M_j) = \min(bwl(M_i), bwl(M_j))$. For example, $bwl(M_1) = 1.544$ Mbps, $bwl(M_2) = 2.048$ Mbps, $bw(M_1, M_2) = \min(bwl(M_1),$ and $bwl(M_2)) = \min(1.544$ Mbps, 2.048 Mbps$) = 1.544$ Mbps.

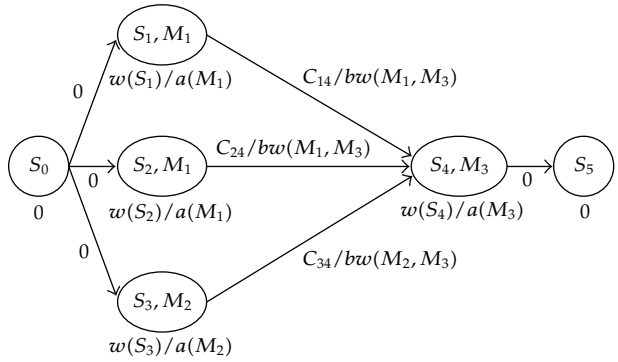


Figure 3: Task-resource assignment graph (T-RAG).

The objective of the task-resource assignment in heterogeneous grid is to minimize tasks' completion time of application. The task-resource assignment in heterogeneous grid can be expressed by the task-resource assignment graph (T-RAG) as displayed in Figure 3. Restated, a task-resource assignment graph corresponds to a solution of the studied scheduling problem. Figure 3 is an example of a task scheduling problem including 6 tasks and 3 heterogeneous resources. After assigning the resources to tasks, the tasks' execution and communication times are settled. Hence, the scheduling completion time of tasks can be calculated after the T-RAG is determined.

In the T-RAG, G, a set of nodes and edges, consists of paths from $S_0$to S_{K+1}. For example, there are 3 paths in Figure 2, where $S_0$and S_5 are pseudotasks for scheduling indicating the start and finish states in the graph.

For path p, $V(p)$ and $E(p)$ are the sets of nodes and edges on path p. Once the node $u \in V(p)$ is correlated to S_u, the node cost of node u is $\text{cost}(u) = E_u$. And when the edge $e \in E(p)$, which is correlated to S_u and S_v, the edge cost is $\text{cost}(e) = T_{uv}$.

Hence, the path cost of path p can be calculated by

$$\text{cost}(p) = \sum_{u \in V(p)} \text{cost}(u) + \sum_{e \in E(p)} \text{cost}(e). \tag{2.2}$$

The path with the largest cost in the T-RAG G corresponds to a feasible scheduling completion time, that is, a solution of the task scheduling problem is associated with a fitness f as

$$f = \max\left(\text{cost}(p) \mid p \in G\right). \tag{2.3}$$

However, the fitness f is set to a very big value if the schedule is an infeasible solution (such as in a deadlock). The primary objective of this study is to minimize the fitness f, $\min(f)$.

Table 7

Task	S_0	S_1	S_2	S_3	S_4	S_5
Depth	0	1	1	1	2	3

Therefore, the objective of studied scheduling problem is summarized as indicated in

$$\begin{aligned}
&\min(f) = \min\{\max(\text{cost}(p) \mid p \in G)\},\\
&\text{where,}\\
&\text{cost}(p) = \sum_{u \in V(p)} \text{cost}(u) + \sum_{e \in E(p)} \text{cost}(e),\\
&\text{cost}(u) = E(u) = \frac{w(S_u)}{a(M_i)},\\
&\text{cost}(e) = T_{uv} = \frac{c_{uv}}{bw(M_i, M_j)}, \quad bw(M_i, M_j) = \min\{bwl(M_i), bwl(M_j)\}.
\end{aligned} \tag{2.4}$$

2.2. Task Queue

The feasible task-resource assignment in heterogeneous grid is subject to precedence constraint. In the example in Figure 3, the S_1 and S_2 are assigned to use resource M_1 at the same time, and hence the sequence for the S_1 and S_2 must be indicated. Therefore, the task queue $Q_i = \{p_1, p_2, \ldots, p_{n(i)}\}$ for M_i (resource i) is designated, where the p_1 is the first executed task on M_i, and the number of tasks to be executed on M_i is $n(i)$. For example, if $Q_1 = \{S_2, S_1\}$, then S_2 must be executed prior to S_1. However, a new precedence for the resource can be constructed as shown by the dashed arrow in Figure 4 and the calculation of cost in (2.2) should include the new precedence (edge). Meanwhile, fitness calculation is based on the new T-RAG constructed according to the new precedence.

After constructing the new precedence, if the corresponding T-RAG does not satisfy the ordering relationship among tasks, it is called illegal T-RAG, that is, an infeasible solution. For example, S_1, S_2, and S_4 tasks are assigned to resource M_2, and $Q_2 = \{S_2, S_4, S_1\}$. The corresponding T-RAG as displayed in Figure 5 is an illegal T-RAG; hence an infeasible solution is produced since S_4 is executed prior to S_1, violating the ordering relationship between S_1 and S_4 tasks indicated in Figure 2. Restated, the illegal T-RAG may occur for an improper task queue. However, this situation could be detected via depth of tasks in DAG, that is, the task with less depth has higher priority than the task with more depth in DAG. In Figure 5, the depth of each task is shown (see Table 7). Hence S_1 should be executed on M_3 prior to S_4.

3. Particle Swarm Optimization

The particle swarm optimization (PSO) was first proposed by Kennedy and Eberhart (1995) [16]. It is a multiagent general metaheuristic, and can be applied extensively to solve many complex problems. The PSO consists of a swarm of particles in the search space; the position

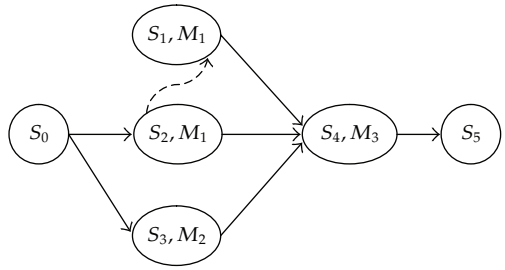


Figure 4: Constructing new precedence.

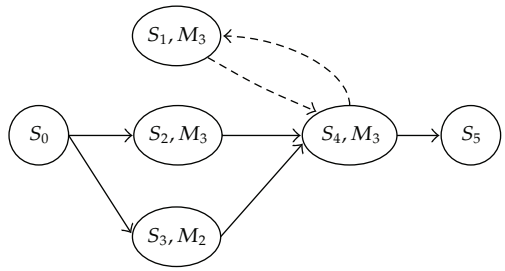


Figure 5: Illegal T-RAG.

of a particle is indicated by a vector which represents a solution. PSO is initialized with a population of random positioned particles and searches for the best position with best fitness.

In each generation or iteration, every particle moves to a new position and the new position is guided by the velocity (which is vector), then the corresponding fitness of the particles would be calculated. Therefore, the velocity plays an important role in searching for a solution with better fitness. There are two experience positions used in the PSO for updating the velocity; one is the global experience position of all particles, which remembers the global best solution obtained by all particles; the other is each particle's individual experience, which recalls the best position that particle has been in. These two experience positions are used to determine the velocity.

Let an N dimensional space (the number of dimensions is typically concerned with the definition of the problem) have M particles. For the ith particle ($i = 1, \ldots, M$), its position consists of N components $X_i = \{X_{i1}, \ldots, X_{iN}\}$, where X_{ij} is the jth component of the position, the velocity of particle i is $V_i = \{V_{i1}, \ldots, V_{iN}\}$, and the particle's individual best experience is $L_i = \{L_{i1}, \ldots, L_{iN}\}$. Additionally, $G = \{G_1, \ldots, G_N\}$ represents the global best experience shared among all the particles. The particle's individual best experience (L_i) and the global best experience of the swarm (G) are recorded and are updated in every step. When updating, the jth component of the position and velocity of the ith particle are according to (2.4) shown in

$$V_{ij}^{\text{new}} = w \times V_{ij} + c_1 \times r_1 \times \left(L_{ij} - X_{ij}\right) + c_2 \times r_2 \times \left(G_j - X_{ij}\right),$$
$$X_{ij}^{\text{new}} = X_{ij} + V_{ij}^{\text{new}}, \tag{3.1}$$

```
for each time do
   for each particle i in the swarm do
      update position X_i^new using (2.4)
      calculate particle's fitness f(X_i^new)
      update L_i&G
   end for
end for
```

Algorithm 1: The pseudocode of the PSO update process.

where w is an inertial weight used to determine the influence of the previous velocity to the new velocity. The c_1 and c_2 are learning factors used to derive how the ith particle approaching the position closes in on the individual experience position or global experience position, respectively. Furthermore, r_1 and r_2 are the random numbers uniformly distributed in [0, 1], influencing the tradeoff between the global and local exploration abilities during search. In every step, a fitness (fitness(i)) of new particle i position (X_i^{new}) obtained via (3.1) is computed and compared with the fitness (fitness(L_i)) of the recorded individual best experience (L_i). The recorded individual best experience (L_i) is replaced by X_i^{new} if fitness(i) $\leqq$ fitness(L_i). Otherwise, L_i is maintained. Meanwhile, is compared with the fitness (fitness(G)) of the recorded global best experience (G). The recorded global best experience (G) is replaced by X_i^{new} if fitness(i) $\leqq$ fitness(G). Otherwise, G is maintained. The procedure of the particle swarm optimization is as shown in Algorithm 1.

4. Enhanced Particle Swarm Optimization Scheme

4.1. Encoding Task-Resource Assignment and Task Priority

In PSO, the position of a particle is indicated by a vector which represents the solution. How to map the position vector to the solution is significant for the PSO process. Since the task scheduling problem can be solved by task-resource assignment, a scheme for encoding task-resource assignment into the position vector is necessary.

In this study, an easy discrete transfer scheme is applied for transferring the position vector to the task-resource assignment. For the position vector X^A with K components (K tasks), the values of X^A are bounded in (0, 1], and the ith component of X^A, which can be transferred to the resource m assigned to the S_i by the discrete transfer scheme as listed in(3.1)

$$m = \left\lceil X_i^A \times N \right\rceil, \quad X_i^A \in (0,1],\ i = 1,2,\ldots,K, \tag{4.1}$$

where N is the number of resources, if $N = 2$ and $X_4^A = 0.7$, the resource assigned to the S_4 would be resource M_2. Suppose that the generated task-resource assignment vector, A, based on (3.1) is as follows (Table 8).

Moreover, when the same resource is assigned to more than one task, the task priority needs to be decided for the task queue. Hence, the task priority vector $\Pr = \{p_1, p_2, \ldots, p_K\}$ is used, where p_1 is the task with highest priority. For example, if $\Pr = \{S_5, S_1, S_3, S_2, S_4\}$,

Table 8

Dimension	1	2	3	4	5
X^A	0.4	0.2	0.3	0.7	0.8
A	1	1	1	2	2

Table 9

Key	1	2	3	4	5
X^R	1.5	3.6	0.15	8.9	0.2

Table 10

Key	3	5	1	2	4
Sorted X^R	0.15	0.2	1.5	3.6	8.9

where S_1, S_2, S_3 are to use the resource M_1, and S_4, S_5 use resource M_2, then the task queue of M_1 and M_2 would be determined to be $Q_1 = \{S_1,\ S_3,\ S_2\}$ and $Q_2 = \{S_5,\ S_4\}$ based on Pr.

Since the task priority vector Pr is required, the position vector X^R with K components is used for mapping. It's important to note that Pr is a permutation without repeating values; the discrete transfer scheme is not applicable. Instead, the random key scheme is suitable for permutation type solutions. Assume there are 5 keys corresponding to 5 task numbers, and the position vector X^R with 5 components is given as follows (Table 9).

After sorting X^R by increasing order, the keys also rearranged as follows (Table 10).

Then the order of keys (3, 5, 1, 2, 4) can be treated as the task priority vector Pr, that is, $\text{Pr} = \{S_3,\ S_5,\ S_1,\ S_2,\ S_4\}$.

Hence, two position vectors X^A and X^R for PSO are required to represent the task-resource assignment and task priority, and they can be combined to one position vector $X = X^A \cup X^R$; for convenience, $X = \{X_1^A, \ldots, X_K^A, X_1^R, \ldots, X_K^R\}$ has $2K$ components.

4.2. Heuristics for Task Scheduling Problem

A heuristic is an experience-based strategy or technique that can be used as an aid in problem solving. Generally, heuristics use accessible information to control problem-solving processes. For PSO, a randomly assigned initial particle position may lead the algorithm to be inefficient. Therefore, for solving the problem efficiently, the corresponding heuristics are often studied and integrated into metaheuristics in most research. In this study, there are two intuitional heuristics proposed for speeding up the PSO's search when solving a task scheduling problem. They are the best performance resource (BPR) heuristic and the latest finish time (LFT) heuristic, which are used for PSO to initialize the position vectors X^A and X^R, respectively.

These two heuristics are based on a multimode project scheduling problem, with the multimode project scheduling problem is similar to the task-resource assignment problem in this study. In [36], the definitions of a multimode project scheduling problem include the precedence between tasks (in which, the task is called activity), and each task needs to be assigned a "mode" for determining the task's resource type and requirements, hence the task's duration would be a variable depending on the resource type. Restated, a multimode project scheduling problem is regarded as a "task-mode" assignment problem rather than a "task-resource" assignment problem. Without the communication cost consideration, the

studied task scheduling problem and multimode project scheduling problem are similarly defined. Both are constrained by precedence, and execution time of task or activity (based on the assigned resource or mode).

Some heuristics are proposed for multimode project scheduling problems. They are usually classified into two types of rules: the mode priority rule for choosing mode and activity priority rule for deciding priority among activities. Two heuristics were chosen after various heuristics had been reviewed.

The shortest feasible mode (SFM) heuristic [37] in a multimode project scheduling problem is the basis of the proposed best performance resource (BPR) heuristic for task-resource assignment in this study. The rationale behind the SFM heuristic is clear; the shortest feasible mode is selected to minimize the project duration. Restated, the SFM guides tasks to choose the mode with shortest activity duration. Similarly, the logic of the BPR heuristic is apparent with the best performance resource assigned to shorten completion time. Therefore, the BPR heuristic is proposed to assign the best performance resource to all of the tasks during assignment, and then shorten the execution time of tasks reasonably, that is, assigning the resource with maximum ability among resources to $S_u \in S$ and then decide the $\min(E_u) = w(S_u)/\max(a(M_i \mid M_i \in M))$.

In MRCPSP, the latest finish time (LFT) heuristic [38] is often used for deciding the priority of tasks. Meanwhile, the LFT heuristic is applied in this work to give higher priority to the tasks with smaller latest finish times LFTs, where the LFT is similar to that in the program evaluation and review technique (PERT). Moreover, the LFT is calculated by the upper bound of scheduling completion time:

$$\overline{T} = \sum_{i=1}^{K} \max(T_i). \tag{4.2}$$

In (4.2), T_i is determined by the critical path. Then perform a traditional backward recursion using shortest execution time $\min(E_u)$ for all tasks.

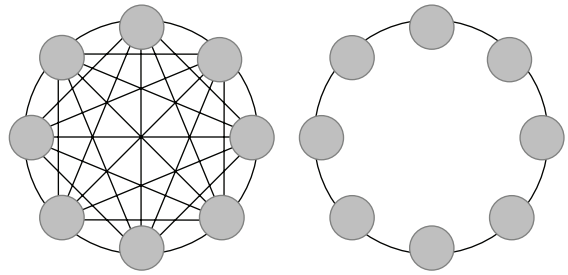
4.3. The Version of Used Particle Swarm Optimization

Many variations of PSO have been proposed, and one of them is the "standard" PSO proposed by Bratton and Kennedy [39] implying that PSO can be significantly improved. In the standard PSO, a different version of velocity update rule is suggested as follows:

$$\begin{aligned} V_{ij}^{\text{new}} &= \chi \times \left(V_{ij} + c_1 \times r_1 \times \left(L_{ij} - X_{ij}\right) + c_2 \times r_2 \times \left(G_j - X_{ij}\right)\right), \\ X_{ij}^{\text{new}} &= X_{ij} + V_{ij}^{\text{new}}, \end{aligned} \tag{4.3}$$

where the χ is constriction factor for adjusting the velocity, and this velocity update rule is suggested for its stability [35]. Hence, this velocity update rule is applied in this study.

In [39], the standard PSO presented two swarm communication topologies for PSO; the "gbest" topology in Figure 6(a) has been studied in most research. The gbest is the global best model where every particle is able to share information with each other quickly, and it is outstanding because of its global communication ability. However, the gbest's global communication ability usually leads to premature convergence. Hence, the "lbest" topology,



(a) The gbest topology (b) The lbest topology

Figure 6: Two topologies of PSO.

like in Figure 6(b), has greatly attracted researchers' attention recently. The feature of lbest is its limited communication; every particle can just communicate with a portion of the swarm. Meanwhile, the lbest topology can be varied such as a ring formation, Von Neumann neighborhood, and so forth. The "lbest" model used in [35] is the simplest form of a local topology, known as the aforementioned *ring* model. The lbest ring model connects each particle to only two other particles in the swarm. Obviously, lbest has a slower convergence rate relative to the gbest.

Bratton et al. note that the gbest usually results in better performance on simple unimodal problems than using lbest, since the situation about falling into local optima does not happen frequently under a unimodal condition. However, the lbest surpasses the gbest in a number of function evaluations, especially in multimodal problems.

For clearly understanding the performance of both topologies for our task scheduling focus, both topologies were tested in Section 5. The global best experience (G) in (4.2) is obtained on the basis of either global communication topology or local communication topology. In this paper, the lbest topology is based on the ring topology, it is a "small-world network" [40] as indicated in Figure 6(b). In other words, the global experience (G) based on local communication topology (*lbest*) is determined by X_{i+1}, X_i, and X_{i-1} particle positions. On the other hand, the global experience (G) is determined by all particles when applying global communication topology (*gbest*).

Accordingly, the procedure of our proposed enhanced particle swarm optimization is summarized as shown in Algorithm 2.

5. Experimental Results and Comparisons

For verifying the suggested scheme in this investigation, simulated cases based on [2, 41] are generated. The simulation case has 15 tasks and 2 pseudotasks as described in Table 11 . In Table 11, the workload, successors, and the communication costs are depicted. For example, task 1 has a workload $w(S_1) = 10000$, its successors are S_2, S_3, S_4, and the corresponding communication costs are $c_{12} = 5$, $c_{13} = 4$, and $c_{14} = 10$.

Three instances of resources A, B, and C with 2, 5, and 10 resources, respectively, are displayed in Table 12. They are used for testing 3 differing resource situations.

The simulation parameters are set as follows: the number of particles is 20 and the initial positions and velocities of particles are randomly assigned except for the heuristics used. Meanwhile, the velocity update rule with constriction factor is used, the constriction

(1) Initialize particles' positions ($X = X^A \bigcup X^R$) by applying heuristics: LFT (for X^R) and BRP (for X^A).
(2) Iteration Loop
  (2.1) For each particle i in the swarm do:
  (2.2) Update velocity vector ($V_i{}^{A^{\text{new}}}$,$V_i{}^{R^{\text{new}}}$) and position vector ($X_i{}^{A^{\text{new}}}$,$X_i{}^{R^{\text{new}}}$) according to (4.2) ("standard" PSO).
  (2.3) Calculate task-resource assignment vector A based on $X_i{}^{A^{\text{new}}}$ (3.1).
  (2.4) Calculate task priority vector Pr by applying the random key scheme to $X_i{}^{R^{\text{new}}}$ and constructing a new precedence for the task queue.
  (2.5) Calculate particle's fitness (based on vector A and vector Pr) via scheduling completion time calculation (2.3).
  (2.6) Update L_i.
  (2.7) Update G (*gbest* or *lbest*).
(3) Until End condition is reached, return solution.

Algorithm 2: The pseudocode of the proposed enhanced PSO.

Table 11: A case consisting of 15 tasks.

Task no.	Workload	Successors no.			Comm. cost		
0	0	1			0		
1	10000	2	3	4	5	4	10
2	20000	5			8		
3	5000	5	8		1	13	
4	40000	6	9		4	15	
5	15000	7			10		
6	1000	10			5		
7	12000	9	10		10	20	
8	8000	10	12		7	15	
9	30000	12			2		
10	11000	11	13		2	20	
11	1000	12			5		
12	1000	14	15		15	11	
13	20000	15			8		
14	6000	15			2		
15	13000	16			0		
16	0						

factor χ is a variable based on the circumstances, and different constriction factor values were tested. The learning factors c_1 and c_2 are set to 2 as suggested in [16]. Meanwhile, r_1 and r_2 are the random numbers uniformly distributed in [0, 1].

There are a total of 24 sets of testing cases simulated as shown in Table 13. The experimental results are also displayed in Table 13. Each set of a testing case is tested for 30 trials to obtain the minimum and average scheduling completion times. The simulation

Table 12: Three instances of resources.

(a) Instance A

Resource no.	Ability	Bandwidth
0	450	8
1	1000	2

(b) Instance B

Resource no.	Ability	Bandwidth
0	450	8
1	1000	2
2	650	10
3	1500	8
4	800	10

(c) Instance C

Resource no.	Ability	Bandwidth
0	450	8
1	1000	2
2	650	10
3	1500	8
4	800	10
5	4000	2
6	2000	15
7	1250	6
8	250	20
9	750	5

result is expressed as the format: *minimum completion time/average completion time*. And each trial was iterated 100 and 300 times. Moreover, the two heuristics and two topologies of PSO were also tested. To verify the influences of both LFT and BPR heuristics, four different situations are tested: without both LFT and BPR; with LFT and without BPR; without LFT and with BPR; with both LFT and BPR. Different global best experiences based on global and local communication topologies impact to the performance are also tested. The global experience (G) for local communication topology (*lbest*) is determined by X_{i+1}, X_i, and X_{i-1} particle positions. Additionally, the best constriction factor for balancing the global and local searches to enhance the performance is surveyed.

The simulation results indicate that the *lbest* topology is apparently better than the *gbest* on average. The LFT heuristic is efficient in the test especially for instance C. However, if the BPR is polarized, it leads to worse scheduling for testing instance A, but outstanding scheduling for instances B and C. The situation indicates that applying the BPR to the task scheduling problem with fewer resources is improper. Moreover, the BPR is able to perform better than LFT for instances B and C. Hence, these two heuristics are worth applying to improve the efficiency of solving task scheduling problems in grid, but the BPR is not suitable for problems with poor resources.

To verify the effectiveness and efficiency of the proposed scheme, some similar scheduling problems were tested since there is no benchmark for task-resource assignment

Table 13: Comparisons of experimental results.

Instance	Iter.	LFT	BPR	χ	Gbest results	χ	Lbest results
A	100			0.9	155.94/165.96	0.7	155.94/159.9
	300				155.94/164.45		155.94/159.18
	100	v			155.94/163.94		155.94/159.34
	300	v			**155.94/163.17**		**155.94/159.04**
	100		v		155.94/182.18		155.94/164.39
	300		v		155.94/181.43		155.94/163.16
	100	v	v		155.94/186.01		155.94/161.33
	300	v	v		155.94/180.87		155.94/161.55
B	100			0.75	93.33/103.61	0.55	93.33/98.77
	300				93.33/102.54		93.33/96.56
	100	v			93.33/100.04		93.33/98.87
	300	v			93.33/99.72		93.33/96.29
	100		v		93.33/96.6		93.33/95.47
	300		v		**93.33/94.93**		93.33/94.57
	100	v	v		93.33/95.6		93.33/96.4
	300	v	v		93.33/95.5		**93.33/94.09**
C	100			0.75	45.5/60.15	0.55	44.5/54.26
	300				44.5/55.45		44.5/50.36
	100	v			44.5/57.51		44.5/54.09
	300	v			44.5/52.59		44.5/48.32
	100		v		44.5/49.52		44.5/48.39
	300		v		**44.5/47.19**		44.5/46.48
	100	v	v		44.5/48.54		44.5/48.07
	300	v	v		44.5/47.23		**44.5/46.02**

Table 14: Complexity of the scheduling problem.

	Solution space	Example (solutions)
TRA-G (n tasks, m processors)	$n! \times m^n$	$16! \times 10^{16} \fallingdotseq 2.1 \times 10^{29}$ ($n = 16$, $m = 10$)
MRCPSP (n tasks, m modes)	$n! \times m^n$	$30! \times 3^{30} \fallingdotseq 5.5 \times 10^{46}$($n = 30$, $m = 3$)

of heterogeneous grid. As stated in Section 4.2, the multimode project scheduling problem is similar to the studied task-resource assignment problem. There are some multimode resource constrained project scheduling problem (MRCPSP) instances in the project scheduling problem library (PSPLIB) [42], including scheduling problems with 10, 12, 14, 16, 18, 20, and 30 jobs (they are denoted by J10, J12, J14, J16, J18, J20, and J30, resp.). Every job case has different instances (e.g., 536 instances for J10 case and 552 instances for J30 case) with different available modes. Meanwhile, MRCPSP has been confirmed to be an NP-hard optimization problem [43]. Table 14 shows the difficulty of solving the studied problem. Meanwhile, the "Example" column represents the possible solutions of the studied problems. The time required by exhaustive search for simulated TRA-G problem would be $2.1 \times 10^{29} \times 10^{-8}$ seconds $\fallingdotseq 2.4 \times 10^{16}$days (a solution that can be found in 0.01 μsec (10^{-8} sec) is assumed). The time required by exhaustive search for simulated MRCPSP-J30 problem would be $5.5 \times 10^{46} \times 10^{-8}$ seconds $\fallingdotseq 6.3 \times 10^{33}$ days.

Table 15: Comparison of algorithms with respect to schedules—5000 on J30.

Algorithm	Lower bound solutions found (%)	CPU-time (sec.)
PSO (this work)	52.46/55.02/57.97*	0.769
CPSO [44]	57.41	N/A
GA [45]	42.93	0.34
SA [46]	25.60	41.8

*minimum/average/maximum percentages.

Table 16: Comparison of algorithms with respect to computation time—1 sec. on J30.

Algorithm	Min. Dev. (%)	Avg. Dev. (%)	Max. Dev. (%)	Feasible solutions found (%)
PSO (this work)	0%	4.48	78.95	92.39
GA [36]	N/A	16.93	151.9	86.3
Truncated B&B [48]	N/A	57.22	244.0	55.8

In this study, the suggested algorithm was applied to solve the "largest" scale instance, the J30 case. In J30 case, there are 3 modes available for each activity. Additionally, most of the research performance comparison among different methods is conducted by evaluating the same number of schedules for each case, for example, 1000 schedules, 5000 schedules, or 50000 schedules. Meanwhile, MRCPSP is a task-mode assignment problem; hence, the SFM heuristic is tested instead of the BPR heuristic. Table 15 shows the simulation results of all 552 instances of the J30 case. Each instance simulation was stopped whenever a total number of 5000 schedules were evaluated. The comparison is on the basis of lower bound solutions since no optimal solution is known, and for many instances of that set, a feasible solution does not exist. Hence, the ratio of lower bound solutions was measured, (lower bound solutions found/552)$*$100%, in the test; test results are demonstrated in Table 15. Moreover, CPU times employed by algorithms are also displayed.

Table 15 illustrates that the proposed scheme yields 57.97% lower bound solutions on J30 case. Currently, no study's results exceed 60% optimal solutions of all 552 instances. Nevertheless, some studies present their scheme performance with over 60% optimal solutions found, since they did not include all 552 instances (such as [47]) for their tests. Meanwhile, the CPU time for finding 5000 schedules by proposed PSO is about 0.769 second. However, the performance comparison based on computation time was also conducted by Hartman [36]. Hence, the comparisons of simulation results from different algorithms for a time limit of one second are displayed in Table 16. The simulation results for the set with J30 reflect deviations from a lower bound makespan. The average deviation, maximum deviation, and found feasible solutions ratio comparisons are given in Table 16. In this work, more than 6000 schedules (for one instance of J30 case) can be obtained from our algorithm in one second. Restated, the consumed CPU time to find a feasible solution for every instance for J30 would be less than 0.167 ms. The average and maximal deviations are less than 5% and 80%, respectively; more than 90% feasible solution can be obtained in one second. Hence, the proposed scheme is effective for solving similar problems in task-resource assignment.

6. Conclusions and Discussion

The task scheduling of resource allocation is a critical issue in grid. Most task scheduling problems are complicated and NP-complete. The studied task scheduling problem is

regarded as a task-resource assignment graph optimization problem. This study proposes an enhanced scheme based on the "standard" PSO to solve the task scheduling problem in grid. Meanwhile, there are two extra heuristics proposed to enhance the efficiency of problem solving: the LFT heuristic and BPR heuristic are applied, where the LFT heuristic helps PSO to decide the priority of tasks in the task queue, and performing well for each test in this study. The BPR heuristic helps PSO to determine task-resource assignment, and performs well for problems with more available resources. However, applying the proposed BPR for task-resource assignment would be worse with a problem with poor resources, since fewer resources can be chosen for obtaining the best performance resource. Moreover, two PSO communication topologies in obtaining global best experience are evaluated for solving task scheduling in this study; they are global communication topology (gbest) and local communication topology (lbest). The "lbest" performs better than the "gbest" in most tests of this problem class. According to simulation results as displayed in Table 13, this proposed scheme, involving two heuristics and "*lbest*" communication topology on top of the "standard" PSO, is effective and efficient for solving task-resource assignment problems in grid. Meanwhile, the proposed scheme can obtain 57.97% lower bound solutions of the largest instances of PSPLIB [42] in 0.769 second, as displayed in Table 15. More than 90% feasible solutions can be yielded in one second; resulting minimal and maximal deviations are less than 5% and 80%. Restated, the simulation results verify that the proposed scheme is adequate and efficient for solving this class of scheduling problems.

Although the scheme proposed in the study is able to solve the task scheduling problem efficiently, further improvement is possible if the BPR heuristic can be improved and made to perform well for any problem. Hence, future work should investigate more suitable heuristics. Meanwhile, this study does not consider the contention for communication resources. To be more close to the real situation, the contention consideration as in [20] should be included in future work. Moreover, more complex task scheduling problems such as processor utilization consideration are planned to be further investigated.

Acknowledgment

This work was partly supported by the National Science Council, Taiwan, under contract NSC 99-2221-E-167-007.

References

[1] I. Foster, C. Kesselman, and S. Tuecke, "The anatomy of the grid: enabling scalable virtual organizations," *International Journal of High Performance Computing Applications*, vol. 15, no. 3, pp. 200–222, 2001.

[2] T. Chen, B. Zhang, X. Hao, and Y. Dai, "Task scheduling in grid based on particle swarm optimization," in *Proceedings of the 5th International Symposium on Parallel and Distributed Computing (ISPDC '06)*, pp. 238–245, July 2006.

[3] A. Salman, I. Ahmad, and S. Al-Madani, "Particle swarm optimization for task assignment problem," *Microprocessors and Microsystems*, vol. 26, no. 8, pp. 363–371, 2002.

[4] M. Aggarwal, R. D. Kent, and A. Ngom, "Genetic algorithm based scheduler for computational grids," in *Proceedings of the 19th International Symposium on High Performance Computing Systems and Applications (HPCS '05)*, pp. 209–215, May 2005.

[5] G. Malewicz, A. L. Rosenberg, and M. Yurkewych, "On scheduling complex dags for internet-based computing," in *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS '05)*, April 2005.

[6] L. He, S. A. Jarvis, D. P. Spooner, D. Bacigalupo, G. Tan, and G. R. Nudd, "Mapping DAG-based applications to multiclusters with background workload," in *Proceedings of the IEEE International Symposium on Cluster Computing and the Grid*, pp. 855–862, May 2005.
[7] G. T. Ross and R. M. Soland, "A branch and bound algorithm for the generalized assignment problem," *Mathematical Programming*, vol. 8, pp. 91–103, 1975.
[8] R. M. Chen, S. T. Lo, and Y. M. Huang, "Combining competitive scheme with slack neurons to solve real-time job scheduling problem," *Expert Systems with Applications*, vol. 33, no. 1, pp. 75–85, 2007.
[9] F. E. Sandnes, "Secure distributed configuration management with randomised scheduling of system-administration tasks," *IEICE Transactions on Information and Systems*, vol. E86-D, no. 9, pp. 1601–1610, 2003.
[10] M. Basu, "Hybridization of artificial immune systems and sequential quadratic programming for dynamic economic dispatch," *Electric Power Components and Systems*, vol. 37, no. 9, pp. 1036–1045, 2009.
[11] J. H. Holland, "Genetic algorithms and classifier systems: foundations and future directions," in *Proceedings of the 2nd International Conference on Genetic Algorithms and Their Application*, 1987.
[12] S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.
[13] F. Glover, "Tabu search—part I," *ORSA Journal on Computing*, vol. 1, no. 3, pp. 190–206, 1989.
[14] F. Glover, "Tabu search—part II," *ORSA Journal on Computing*, vol. 2, no. 1, pp. 4–32, 1990.
[15] M. Dorigo and L. M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 53–66, 1997.
[16] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the 4th IEEE International Conference on Neural Networks*, pp. 1942–1948, December 1995.
[17] J. Oh and C. Wu, "Genetic-algorithm-based real-time task scheduling with multiple goals," *Journal of Systems and Software*, vol. 71, no. 3, pp. 245–258, 2004.
[18] Z. Liu and H. Wang, "GA-based resource-constrained project scheduling with the objective of minimizing activities' cost," in *Proceedings of the International Conference on Intelligent Computing (ICIC '05)*, vol. 3644 of *Lecture Notes in Computer Science*, pp. 937–946, August 2005.
[19] N. Amjady and A. Shirzadi, "Unit commitment using a new integer coded genetic algorithm," *European Transactions on Electrical Power*, vol. 19, no. 8, pp. 1161–1176, 2009.
[20] O. Sinnen, L. A. Sousa, and F. E. Sandnes, "Toward a realistic task scheduling model," *IEEE Transactions on Parallel and Distributed Systems*, vol. 17, no. 3, pp. 263–275, 2006.
[21] K. Bouleimen and H. Lecocq, "A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version," *European Journal of Operational Research*, vol. 149, no. 2, pp. 268–281, 2003.
[22] K. H. Kim and K. C. Moon, "Berth scheduling by simulated annealing," *Transportation Research B*, vol. 37, no. 6, pp. 541–560, 2003.
[23] G. Wan and B. P.-C. Yen, "Tabu search for single machine scheduling with distinct due windows and weighted earliness/tardiness penalties," *European Journal of Operational Research*, vol. 142, no. 2, pp. 271–281, 2002.
[24] S. G. Ponnambalam, P. Aravindan, and S. V. Rajesh, "Tabu search algorithm for job shop scheduling," *International Journal of Advanced Manufacturing Technology*, vol. 16, no. 10, pp. 765–771, 2000.
[25] S. T. Lo, R. M. Chen, Y. M. Huang, and C. L. Wu, "Multiprocessor system scheduling with precedence and resource constraints using an enhanced ant colony system," *Expert Systems with Applications*, vol. 34, no. 3, pp. 2071–2081, 2008.
[26] W. J. Gutjahr and M. S. Rauner, "An ACO algorithm for a dynamic regional nurse-scheduling problem in Austria," *Computers and Operations Research*, vol. 34, no. 3, pp. 642–666, 2007.
[27] F. Zhao, Y. Hong, D. Yu, Y. Yang, and Q. Zhang, "A hybrid particle swarm optimisation algorithm and fuzzy logic for process planning and production scheduling integration in holonic manufacturing systems," *International Journal of Computer Integrated Manufacturing*, vol. 23, no. 1, pp. 20–39, 2010.
[28] T. L. Lin, S. J. Horng, T. W. Kao et al., "An efficient job-shop scheduling algorithm based on particle swarm optimization," *Expert Systems with Applications*, vol. 37, no. 3, pp. 2629–2636, 2010.
[29] H. Liu, A. Abraham, and Z. Wang, "A multi-swarm approach to multi-objective flexible job-shop scheduling problems," *Fundamenta Informaticae*, vol. 95, no. 4, pp. 465–489, 2009.
[30] R. M. Chen, C. L. Wu, C. M. Wang, and S. T. Lo, "Using novel particle swarm optimization scheme to solve resource-constrained scheduling problem in PSPLIB," *Expert Systems with Applications*, vol. 37, no. 3, pp. 1899–1910, 2010.

[31] C. Chiu, M. J. J. Wu, Y. T. Tsai, N. H. Chiu, M. S. H. Ho, and H. J. Shyu, "Constrain-based particle swarm optimization (CBPSO) for call center scheduling," *International Journal of Innovative Computing, Information and Control*, vol. 5, no. 12, pp. 4541–4549, 2009.

[32] M. F. Tasgetiren, Y. C. Liang, M. Sevkli, and G. Gencyilmaz, "A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem," *European Journal of Operational Research*, vol. 177, no. 3, pp. 1930–1947, 2007.

[33] J. Behnamian, M. Zandieh, and S. M. T. Fatemi Ghomi, "Due windows group scheduling using an effective hybrid optimization approach," *International Journal of Advanced Manufacturing Technology*, vol. 46, no. 5–8, pp. 721–735, 2010.

[34] Y. Hei, X. Li, K. Yi, and H. Yang, "Novel scheduling strategy for downlink multiuser MIMO system: particle swarm optimization," *Science in China F*, vol. 52, no. 12, pp. 2279–2289, 2009.

[35] M. Clerc and J. Kennedy, "The particle swarm-explosion, stability, and convergence in a multidimensional complex space," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002.

[36] S. Hartmann, "Project scheduling with multiple modes: a genetic algorithm," *Annals of Operations Research*, vol. 102, no. 1, pp. 111–135, 2001.

[37] F. F. Boctor, "Heuristics for scheduling projects with resource restrictions and several resource-duration modes," *International Journal of Production Research*, vol. 31, no. 11, pp. 2547–2558, 1993.

[38] P. Brucker, A. Drexl, R. Möhring, K. Neumann, and E. Pesch, "Resource-constrained project scheduling: notation, classification, models, and methods," *European Journal of Operational Research*, vol. 112, no. 1, pp. 3–41, 1999.

[39] D. Bratton and J. Kennedy, "Defining a standard for particle swarm optimization," in *Proceedings of the IEEE Swarm Intelligence Symposium (SIS '07)*, pp. 120–127, April 2007.

[40] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, no. 6684, pp. 440–442, 1998.

[41] J. N. Lin and H. Z. Wu, "Scheduling in grid computing environment based on genetic algorithm," *Journal of Computer Research and Development*, vol. 41, no. 12, pp. 2195–2199, 2004 (Chinese).

[42] Project Scheduling Problem Library, PSPLIB, http://129.187.106.231/psplib/.

[43] W. Herroelen, B. De Reyck, and E. Demeulemeester, "Resource-constrained project scheduling: a survey of recent developments," *Computers & Operations Research*, vol. 25, no. 4, pp. 279–302, 1998.

[44] B. Jarboui, N. Damak, P. Siarry, and A. Rebai, "A combinatorial particle swarm optimization for solving multi-mode resource-constrained project scheduling problems," *Applied Mathematics and Computation*, vol. 195, no. 1, pp. 299–308, 2008.

[45] J. Alcaraz, C. Maroto, and R. Ruiz, "Solving the multi-mode resource-constrained project scheduling problem with genetic algorithms," *Journal of the Operational Research Society*, vol. 54, no. 6, pp. 614–626, 2003.

[46] J. Józefowska, M. Mika, R. Różycki, G. Waligóra, and J. Węglarz, "Simulated annealing for multi-mode resource-constrained project scheduling," *Annals of Operations Research*, vol. 102, no. 1–4, pp. 137–155, 2001.

[47] C. W. Chiang, Y. Q. Huang, and W. Y. Wang, "Ant colony optimization with parameter adaptation for multi-mode resource-constrained project scheduling," *Journal of Intelligent and Fuzzy Systems*, vol. 19, no. 4-5, pp. 345–358, 2008.

[48] S. Hartmann and A. Drexl, "Project scheduling with multiple modes: a comparison of exact algorithms," *Networks*, vol. 32, no. 4, pp. 283–297, 1998.