Chapter 4

Stochastic Greedy-Based Particle Swarm Optimization for Workflow Application in Grid

Ruey-Maw Chen and Yin-Mou Shen

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.73587

Abstract

The workflow application is a common grid application. The objective of a workflow application is to complete all the tasks within the shortest time, i.e., minimal makespan. A job scheduler with a high-efficient scheduling algorithm is required to solve workflow scheduling based on grid information. Scheduling problems are NP-complete problems, which have been well solved by metaheuristic algorithms. To attain effective solutions to workflow application, an algorithm named the stochastic greedy PSO (SGPSO) is proposed to solve workflow scheduling; a new velocity update rule based on stochastic greedy is suggested. Restated, a stochastic greedy-driven search guidance is provided to particles. Meanwhile, a stochastic greedy probability (SGP) parameter is designed to help control whether the search behavior of particles is exploitation or exploration to improve search efficiency. The advantages of the proposed scheme are retaining exploration capability during a search, reducing complexity and computation time, and easy to implement. Retaining exploration capability during a search prevents particles from getting trapped on local optimums. Additionally, the diversity of the proposed SGPSO is verified and analyzed. The experimental results demonstrate that the SGPSO proposed can effectively solve workflow class problems encountered in the grid environment.

Keywords: scheduling, optimization, stochastic greedy, workflow, particle swarm optimization

1. Introduction

Grid computing is applied mainly to utilize the heterogeneous computational resources to execute various applications. The computing ability of the grid is comparable to that of a super computer, and the grid computing environment consists of heterogeneous computing devices throughout the world and connected by low-latency and high-bandwidth networks [1]. The

main application of the grids is the sharing of computing resources. Scholars have already used the grid in real cases when requiring vast computation and immense storage space [1–3]. The basic scenario of grid computing application is shown in **Figure 1**. The job scheduler uses grid information supplied by information server which collects grid information from grid sensors. When a workflow application comprising numerous tasks with partially ordered constraint is uploaded to the grid, the job scheduler of the grid platform allocates the tasks to be processed to the computing resources on the grid. If the tasks and the resources are well scheduled, the time needed to complete all the tasks of the workflow application can then be reduced. Otherwise, the time will be extended. Restated, the makespan of workflow application on the grid is highly impacted by the quality of task-resource arrangement. Many workflow application scheduling algorithms have been presented to boost efficiency and make the resource manager more efficient when matching tasks and resources so that grid performance can be upgraded effectively.

Many studies have developed scheduling optimization methods intended to reduce the makespan of jobs (all tasks) on the grid [4, 5–9]. When restrictions regarding partially ordered tasks exist between tasks (i.e., dependent tasks), the algorithm applied must meet the needs of such sequential relationships when scheduling optimization is conducted. Besides solving the task-resource matching problem, the sequence of execution of independent tasks allocated to the same resource also has a rather significant effect on the reduction of the makespans of jobs in workflow scheduling. In other words, workflow scheduling has to simultaneously solve two subproblems in task-resource matching and those unrelated to tasks' priorities. Most scheduling problems are NP-complete, and many heuristic and metaheuristic algorithms have been proposed to solve NP problems, such as the ant colony optimization (ACO) [3], genetic algorithm (GA) [6], simulated annealing (SA) [5], and particle swarm optimization (PSO) [10]. Among them, PSO carries the advantages of easy implementation, requiring fewer parameters and having a faster convergence speed; therefore, PSO is often used to solve scheduling problems in fields aside from a grid or cloud computing, such as course timetabling problems [11], flowshop problems [12], and vehicle routing problems (VRP) [13]. Also, PSO was applied

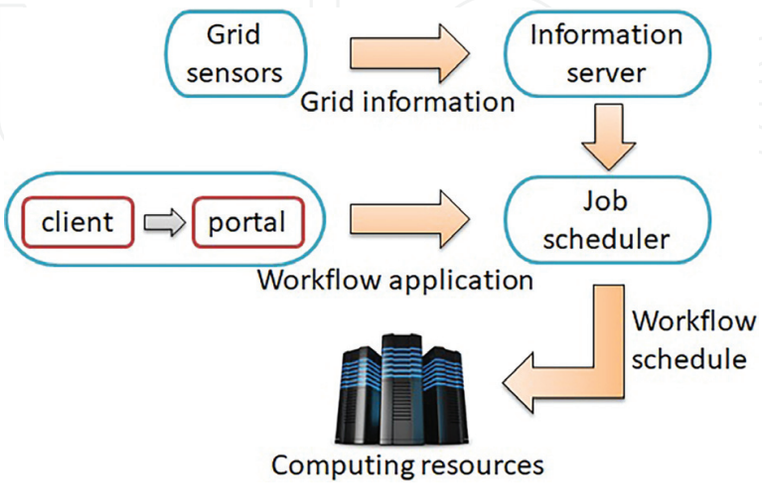

Figure 1. Grid application scenario.

to solve grid scheduling problems; Tao et al. [14] and Chen and Wang [15] have all adopted the PSO to solve grid scheduling problems effectively.

Two subproblems have to be dealt with in workflow scheduling: the task-resource mapping and the execution priorities of tasks without precedence constraint in the same computing resource. To these two subproblems, two PSO algorithms were designed to find the corresponding solutions: a discrete-type PSO algorithm to solve the task-resource mapping subproblem and a constriction-type PSO algorithm to solve the task priority subproblem. In PSO, the location of each particle represents a solution to the problem to be solved, and each particle moves with reference to global experience and individual experience, resulting in a new solution. In this study, a new velocity update rule developed from state transformation rules used in ant colony optimization (ACO) [16] was proposed rather than the velocity update rule in the contraction-type PSO, where movement is based on both experiences. This new PSO is named the stochastic greedy PSO (SGPSO) herein. In ACO, the ant moves by referencing the highest pheromone. Besides movement guided by the highest pheromone trail, the ant also references the other trail, determined using the roulette wheel rule to move. In this study, the global experience of the particle swarm is regarded as the path of ants with the highest pheromone. Thus, a new velocity update rule was introduced to allow the particles with the probability to explore on their own in the solution search process and prevent them from getting trapped on local optimums. Restated, the particle moves in accordance with either the global experience or individual experience in this work. Moreover, different problem cases with small-scale and large-scale problems are designed and tested to verify the performance of the proposed SGPSO. In the end, the workflow scheduling problems in [15] were tested, and comparative analysis was conducted. Furthermore, the diversity of the proposed SGPSO is also defined and verified. The remaining parts of this paper are arranged as follows: Section 2 presents the workflow scheduling problems, Section 3 describes the new velocity update rule applied in the particle swarm optimization algorithm and the concept behind its adoption from ACO, in Section 4 the experimental results and comparative analysis are provided, and the conclusions are presented in Section 5.

2. Description of workflow scheduling problems

In the grid environment, distributed computing is conducted with the resources that are scattered among different places around the world and connected together through networks. The composition of the grid environment is shown in **Figure 2(a)**. The scattered computational resources are linked through the Internet. Each resource has its own computing ability and external bandwidth represented by AB_u and BW_u, respectively. Moreover, the resources in the grid environment are heterogeneous resources, meaning that the computing ability and external bandwidth of each resource are dissimilar. Generally, the grid environment can be represented with a schematic $G(R, C)$ composed of nodes and edges. Each node stands for a resource ($\boldsymbol{R} = \{R_i\}$). $i = 1 \sim N$ represents the set of all resources. N is the total amount of resources in the grid environment. The connections between resources are represented by an edge. $\boldsymbol{C} = \{C_{uv}\}$ stands for the set of resource-resource connections. C_{uv} is the connection between

resource u and resource v. The grid computing environmental schematic is shown in **Figure 2 (b)**. The workflow application can be represented with a directed acyclic graph (DAG) $G(\boldsymbol{V}, \boldsymbol{T})$, as shown in **Figure 3**, in which $\boldsymbol{V} = \{V_i\}$, where $i = 1 \sim M$ is the set of all tasks and M is the total number of tasks. Precedence exists between certain tasks, and each task has a workload; w_i represents the workload of task i. **Figure 3** also indicates the precedence relationship between tasks such as the following: task 1 is the predecessor of tasks 3 and 4, and tasks 3 and 4 are the successors of task 1. Meanwhile, task 5 cannot be executed until task 2 is done, and task 6 has to wait till tasks 3, 4, and 5 are accomplished. Certain required data for execution on successors have to be transmitted to the successors when the predecessor is completed, i.e., transmission costs exist. TC_{ij} represents the amount of data transmitted between tasks i and j. If the predecessor and the successors are arranged to be executed with different resources, a transmission cost exists. On the contrary, if the predecessor and the successors are arranged to be executed with the same resource, there will be no transmission cost. Meanwhile, task 0 and task 8 in the figure are virtual tasks representing the start and the end, respectively. They have no workload and involve no data transmission costs.

The goals of workflow scheduling optimization are to appropriately match tasks to resources and to suitably assign execution priorities to tasks without precedence restriction in the same computing resource to reduce the makespan of the application execution. The cost includes the resource processing time of the path and the data transmission time. In **Figure 3**, for example,

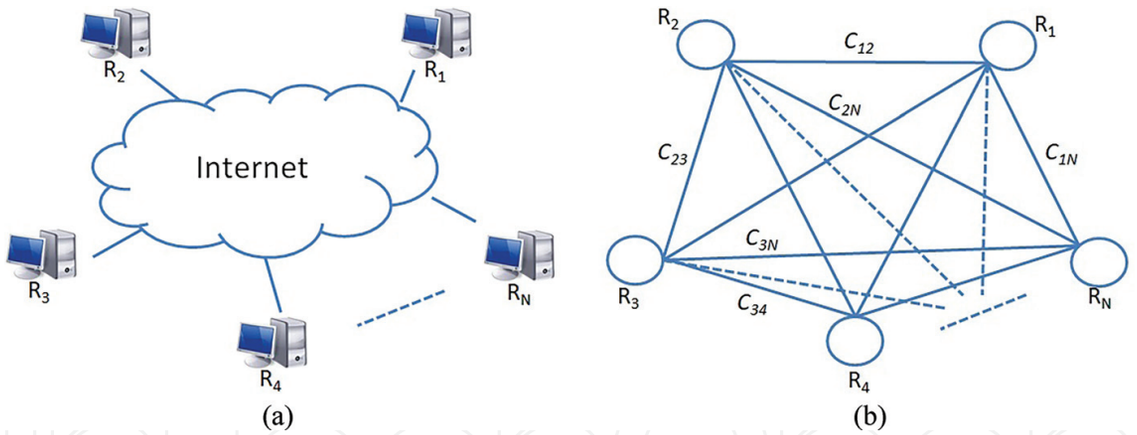

Figure 2. Grid computing environment. (a) The composition of the grid environment. (b) The grid computing environmental schematic.

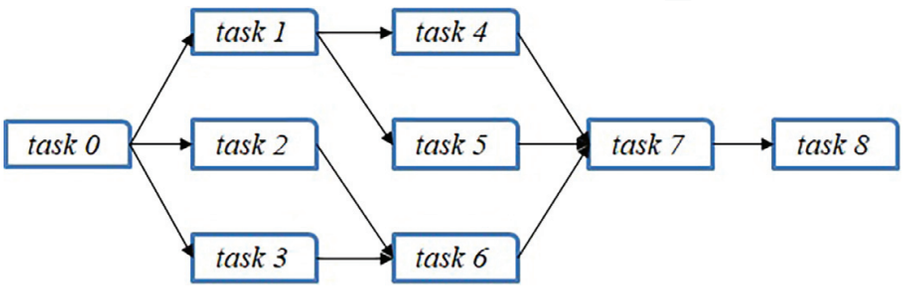

Figure 3. Directed acyclic graph (DAG) of a workflow application on the grid.

there are four execution sequence paths: (p1) task 0→task 1→task 4→task 7→task 8, (p2) task 0→task 1→task 5→task 7→task 8, (p3) task 0→task 2→task 6→task 7→task 8, and (p4) task 0→task 3→task 6→task 7→task 8. All the tasks in the four execution sequences must be executed to complete the job on the grid. The makespan is subject to the time of the longest execution sequence path. That is, the max(*cost*(p_i)). *cost*(p_i) is the cost of an execution sequence path (p_i) in DAG in the grid environment.

Calculation of *cost*(p_i) is shown in Eq. (1). *cost*(p_i) is the aggregate of the total resource processing time on execution sequence path p_i and the total data transmission time on that path. In Eq. (1), $u(w_t)$ represents the workload of the tasks allocated to resource u, and *cost*(*tf*) is the data transmission cost or time on that execution sequence path, as shown in Eq. (2):

$$cost(p_i) = \frac{\sum_{u \in p_i} u(w_t)}{AB_u} + \sum_{tf \in \mathbf{T}} cost(tf) \tag{1}$$

$$cost(tf) = \frac{TC_{ij}}{\min\{BW_u, BW_v\}} \tag{2}$$

If task i and task j are, respectively, allocated to be executed with the resources u and v, between the two tasks exists the amount of data transmission (*TCij*). Since resource u and resource v have different bandwidths, the data transmission time is subject to the smaller bandwidth (BW_{uv} = min{BW_u, BW_v}). Hence, the data transmission time or cost is the amount of data transmitted divided by the smaller resource bandwidth.

Therefore, the makespan is defined as the fitness function to denote the quality of workflow scheduling. The definition of fitness function (*FIT*) is shown in Eq. (3). The objective of workflow scheduling is then to find the shortest makespan (min(*FIT*)):

$$FIT = \max\{\cos t(p_i) | p_i \in DAG\}, \tag{3}$$

3. The proposed method

Many nature-inspired optimization algorithms have been proposed to find optimal solutions to workflow scheduling problems and metaheuristic algorithms that imitate the behaviors of biological creatures. Some that are extensively applied include ACO, GA, bee colony optimization (BCO), and the PSO adopted in this study. Among them, PSO requires fewer parameters and is easier to implement. Therefore, it has been well applied to solve diverse *NP*-complete problems, and the results have been rather remarkable. Meanwhile, PSO has also been employed to solve workflow scheduling problems with effectiveness.

As shown in **Figure 3**, if task 2 and task 6 are allocated to be computed by different resources, there will be a transmission time, and the makespan will be extended. On the contrary, if they are arranged to be executed by the same resource, there will be no transmission time, and the makespan will be shortened. Furthermore, if task 1 and task 2 are arranged to be executed by the same resource, executing task 1 before task 2 or vice versa will have an effect on the

makespan of the workflow application on the grid. Hence, the study of minimization of the makespan in workflow scheduling involves two issues: task-resource matching and task execution priority determination. **Figure 4** illustrates an example with two heterogeneous resources (R_1 and R_2) with different abilities in the grid environment; these two subproblems need to be solved for the studied workflow application scheduling problem. **Figure 4(a)** displays the task-resource matching subproblem, and **Figure 4(b)** indicates some possible execution priorities of the tasks (tasks 1, 3, 4, and 5) assigned to resource 1 (R_1).

To deal with these issues, the constriction PSO is used to solve the task execution priority issue and the discrete PSO to cope with the task-resource matching issue.

3.1. Used PSOs

PSO was first introduced by Kennedy and Ebrhart in 1995 [17]. After observing birds flying, fish seeking food, and other social behaviors of animals, they discovered that each particle would move to their next position according to information (experience) shared among the members of the swarm. Restated, when particles move, they refer to individual experience (*pbest*) and global experience (*gbest*). At each moving step, a particle will refer to both kinds of experience to make the next move. The particle represents the solution of the problem to be solved. Hence, the movement of particles is regarded as a solution search in a solution space. The position update equation of PSO is as shown in Eq. (4):

$$\begin{cases} V_{ij}^{new} = \omega V_{ij} + c_1 r_1 \left(pbest_{ij} - X_{ij} \right) + c_2 r_2 \left(gbest_j - X_{ij} \right) \\ X_{ij}^{new} = X_{ij} + V_{ij}^{new} \end{cases} \tag{4}$$

Each particle of a swarm has its own velocity V. The V_{ij} represents the jth velocity component of the particle i. X is the position of the particle and X_{ij} the jth position component of particle i. $pbest_{ij}$ is the jth component of the best individual experience of the particle i. The $gbest_j$ is the jth component of the best global experience. w is the inertia weight adopted mainly for controlling the level of influence of the previous velocity on the velocity of the current iteration. c_1 and c_2 are the learning factors for controlling the influence of individual best experience and global

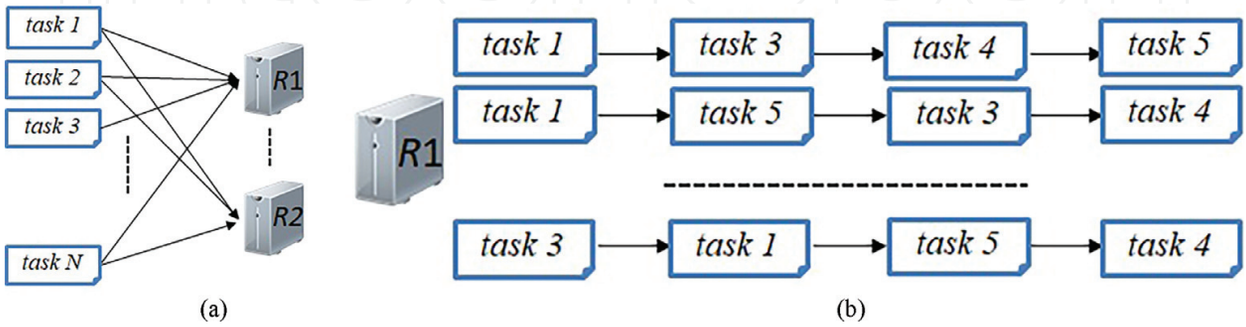

Figure 4. Two subproblems of a workflow application in grid. (a) The task-resource matching subproblem and (b) the task execution priority subproblem.

best experience on the velocity of iteration. r_1 and r_2 are the random numbers between 0 and 1, adopted to increase the diversity of particle movement and prevent particles from moving only toward the individual best experience or the global best experience.

In this work, a constriction PSO is applied to solve the task execution priority problem. Similar to the inertia weight, a constriction factor χ was introduced into PSO to balance global and local searches, i.e., the constriction factor used to limit the movement size, and is named constriction PSO [18]. The velocity update rule used in constriction PSO is indicated in Eq. (5):

$$\begin{cases} V_{ij}^{new} = \chi\left[V_{ij} + c_1 r_1\left(gbest - X_{ij}\right) + c_2 r_2\left(pbest_{ij} - X_{ij}\right)\right] \\ X_{ij}^{new} = X_{ij} + V_{ij}^{new} \end{cases} \tag{5}$$

The discrete PSO (DPSO) was proposed by Kennedy and Ebrhart in 1997 [19]. Unlike in conventional PSO, the particle position components of the discrete PSO are binary. In other words, $X_{ij} \in (0,1)$. The velocity update is similar to Eq. (5), but the constriction factor is set to 1. The position vector X_{ij} is calculated with the sigmoid function $S(V_{ij})$ in conjunction with velocity vector V_{ij}, and the real valued velocity is converted into a probability. This probability is then compared with a random number to update the position component value either 0 or 1. The position update rule for discrete PSO is displayed in Eq. (6):

$$X_{ij}^{new} = \begin{cases} 0, & S\left(V_{ij}^{new}\right) < \text{rand} \\ 1, & S\left(V_{ij}^{new}\right) \geq \text{rand} \end{cases} \tag{6}$$

The sigmoid function is applied to convert the velocity value into a probability between 0 and 1. However, to prevent the value of $S(V_{ij})$ from becoming too close to 0 or 1, the value of V_{ij} is normally limited to between [$-V_{max}$ and V_{max}]. In this study, DPSO is applied to solve the task-resource matching problem. The particle positions are designed as a two-dimensional matrix $M \times N$. M represents the number of tasks, and N is the number of binary bits, which is the floor function of $\log_2$(resource quantity) plus 1. If the number of resources is R, $N = \lfloor \log_2(R) \rfloor + 1$. In other words, $X_i = [X_{ipq}]$, $p = 1 \sim M$, and $q = 1 \sim N$. The combination of number (N) of binary numbers (after the binary values are converted to metric values) of the p row is the resource allocated for the task p. Suppose that the number of resources is 3 ($R = 3$), then two bits ($N = \lfloor \log_2(3) \rfloor + 1 = 2$) are required.

3.2. Stochastic greedy in ant colony optimization (ACO)

Stochastic greedy is greedy by chance; it has been well applied in constructing the path of ants in ant colony optimization. Ant colony optimization, which was initially introduced by Dorigo et al. in 1996 [20], imitates the foraging behavior of ants. The ACO was first applied to solve the traveling salesman problem [16, 21]. In ACO, ants lay down pheromones on the foraging paths. The deposited pheromone is the stigmergy used to communicate with ants. The amount of pheromone deposited on a particular foraging path increases with the number of ants traveling along the path. An ant foraging path corresponds to a feasible solution to the studied

scheduling problem; an ant establishes a path by using a transition rule to choose nodes to visit, i.e., each movement is determined by stochastic greedy rule as displayed in Eq. (7):

$$j = \begin{cases} \arg\max_{l \in J_k(i)} \left\{ [\tau(i,l)]^{\alpha} [\eta(i,l)]^{\beta} \right\}, & q \le q_0 \\ J, & q > q_0 \end{cases} \tag{7}$$

where $\tau(i, l)$ denotes the pheromone left on edge (i, l) and $\eta(i, l)$ is the heuristic value. α and β are used to determine the relative importance between the pheromone and the heuristic value. $J_k(i)$ represents the set of neighborhood nodes, which can be visited at node V_i by ant k. And, q_0 is a predefined probability usually set to a higher value, and q is a random number between 0 and 1. The next node j to be visited is chosen from $J_k(i)$. When $q \le q_0$, the node with the highest pheromone times heuristic value is selected as the next node V_j (exploitation). If $q > q_0$, V_j is usually determined from $J_k(i)$ by the roulette wheel selection rule. Restated, an ant has a q_0 probability to visit the node with the highest pheromone times the heuristic value and has a $(1 - q_0)$ probability to visit a node other than the node with the highest pheromone times the heuristic value (exploration).

3.3. Stochastic greedy PSO (SGPSO)

This section will introduce the proposed PSO using a new velocity update rule in this study and the design philosophy behind it. In PSO, the particles always move and search for optimums in the solution space in accordance with the best individual experience and the global best experience. In this work, the global best experience in PSO is similar to the path with the highest pheromone level in ACO. Restated, $gbest_j$ in PSO is regarded as the $max\{[\tau(i,j)]^{\alpha} [\eta(i,j)]^{\beta}\}$ in ACO. Without other guidance, PSO can lead other particles to move toward the current global best experience. As a result, the particles can achieve local optimums at an early iteration and become trapped there (local optimum). This indicates that PSO may converge quickly (premature convergence), but trapping on local optimums is likely to happen. The same situation occurs with ACO, if the ants always choose the path with the highest pheromone times the heuristic value (exploitation), it can also lead to the path of local optimum. Hence, ACO retains a certain probability and uses the roulette wheel selection rule (Eq. (7)) to allow ants to explore paths other than that with the highest pheromone so as to prevent from trapping on local optimum. In ACO, a default parameter *SGP* is adopted to determine the relative importance of exploitation and exploration. When the *SGP* value is large, exploitation of the global best path tends to occur, whereas small *SGP* values are more likely to result in individual exploration. To be with the adequate diversity during search, it is important for PSO to find the optimal solution. Hence, to strengthen the exploration capacity of PSO in the solution search process to keep diversity and avoid getting trapped on local optimums, the search mechanism of ACO based on the stochastic greedy rule was implemented in PSO, and hence a new velocity update rule was designed to replace the velocity update rule of referring to both experiences in Eq. (5). Thus, as a stochastic greedy probability (SGP) for global search, *SGP* is designed. Meanwhile, the roulette wheel selection rule used in ACO is modified, i.e., a particle references its own experience rather than using roulette wheel selection rule to reference other particle's experiences. The design of the proposed velocity update rule is as shown in Eq. (8):

$$
\begin{aligned}
V_{ij}^{new} &= \chi \times \left[V_{ij} + c_1 \times r_1 \times \left(Guidance_{ij} - X_{ij}\right)\right] \\
X_{ij}^{new} &= X_{ij} + V_{ij}^{new} \\
Guidance_{ij} &= \begin{cases} gbest_j, & q \le SGP \ (\text{exploitation}) \\ pbest_{ij}, & q > SGP \ (\text{exploration}) \end{cases}
\end{aligned} \tag{8}
$$

where q is a random number between 0 and 1. When q is larger than SGP, the particle velocity is updated in accordance with the individual best experience ($pbest_{ij}$); when it is smaller than SGP, the particle velocity is updated in accordance with the global best experience ($gbest_j$). Restated, a particle has an SGP probability to search following the global experience (exploitation search) and a (1-SGP) probability to search according to individual experience distributed in solution space (exploration search). Restated, the search behavior of particles is driven by the stochastic greedy rule. The PSO using this new velocity update rule is named stochastic greedy PSO (SGPSO) herein. This SGPSO is applied only in the constriction PSO (Eq. (8)) to solve the task execution priority problem, not in the DPSO that deals with the task-resource

Initialization phase:
- Initializes SGPSO: XSGPSO and VSGPSO and DPSO: XDPSO and VDPSO
- Calculates fitness(X), X=XP-SGPSO+XP-DPSO
- Determines individual experience XP=XP-SGPSO+XP-DPSO
- Determines global experience G=GSGPSO+XDPSO

PSO update phase: including priority determination and resource matching

SGPSO:
- Updates VSGPSO and XSGPSO (Eq. (8))
- Determines tasks priorities Pr
- Repairs Pr to Pr' if violating precedence constraint

DPSO:
- Updates VDPSO (Eq. (4)) and XDPSO (Eq. (6))
- Determines task-resource matching Tr

Schedule generation and fitness calculation phase:
- Generates schedule S based on Pr/Pr' and Tr
- Calculates fitness(Xnew), Xnew=XSGPSO+XDPSO

Experiences update phase:

SGPSO:
- Update individual experience if fitness(Xnew)<fitness(XP)
  XP-SGPSO=XSGPSO
- Update global experience if fitness(G)<fitness(XP)
  GSGPSO=XSGPSO

DPSO:
- Update individual experience if fitness(Xnew)<fitness(XP)
  XP-DPSO=XDPSO
- Update global experience if fitness(G)<fitness(XP)
  GDPSO=XDPSO

Back to PSO update phase when the termination condition not met

Figure 5. Operation procedures of the proposed scheme.

matching problem. The operation procedures of the proposed scheme of applying SGPSO and DPSO to solve the interested workflow application scheduling problem in the grid are listed in **Figure 5**.

3.4. The diversity of SGPSO

The diversity of the swarm during moving in the solution space impacts the solution quality. High diversity of the particles in the initial stage is desired for possible most solution space to find a good seed of search. Conversely, in the latter stage, the particles ought to proceed fine search for the better solution, i.e., low diversity of the population should be provided. To analyze the diversity of the SGPSO, the diversity (*DIV*) of a particle swarm is defined by the average absolute distance of whole particles as given in Eq. (9) [22]:

$$DIV = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \frac{Dis(X_i, X_j)}{C_2^N}$$
$$Dis(X_i, X_j) = \sum_{k=1}^{D} |X_{ik}, X_{jk}| \tag{9}$$

where $Dis(X_i, X_j)$ is the absolute distance between particles X_i and X_j. D is the dimension of the particle, and N is the number of particles.

4. Experimental results and discussion

Since there is not any specific library providing grid task scheduling problems for workflow applications, workflow scheduling problems involving larger numbers of tasks were also designed for this study to test whether the proposed method can also perform well on large-scale problems. Intrinsically, the workflow scheduling problem can be regarded as a derivative of the multimode resource-constrained project scheduling problem (MRCPSP). Therefore, the task precedence constraints of the designed workflow scheduling problems in this work are generated based on the problem cases (J10, J12, J14, J16, J18, J20, and J30) of the MRCPSP in the PSPLIB library; each problem case has 50 different instances generated. The workload of an activity is a randomly produced number with 1000 as the base unit. The data transmission between predecessors and successors is created by using random numbers with 10 as the base unit. The processing ability and the external bandwidth of computing resources in the grid were generated as those in the problem designed by [15] as listed in **Table 1**. **Table 2** illustrates the workflow application example of a J14 instance including workload, number of successors, precedence (successor), and communication cost. In **Table 2**, the tasks 0 and 15 are pseudo tasks representing the start and end. The corresponding DAG of the workflow application instance is displayed in **Figure 6**. The settings of the parameters in constriction and discrete PSOs are $\chi = 0.72984$, $\chi = 1$, and $c_1 = c_2 = 2$. The values of different SGP parameters SGP = {0, 0.1, 0.2, …, 0.9, 1} were tested on the all designed problems (J10, J12, J14, J16, J18, J20, and J30) to understand their influence on the SGPSO performance. To evaluate the performance of the workflow application scheduling on the grid, Avg.Dev(%) is used and defined as in Eq. (10):

$$Dev(\%) = \sum_{i \in \text{instances}} \left(\frac{FIT_i - best_i}{best_i} \times 100\% \right) / |\text{instances}|$$
$$Avg.Dev(\%) = \frac{\sum_{t=1 \sim T} Dev(\%)}{T} \tag{10}$$

Resources	MIPS	Bandwidth
R_1	450	8
R_2	1000	2
R_3	650	10
R_4	1500	8
R_5	800	10
R_6	4000	2
R_7	2000	15
R_8	1250	6
R_9	250	20
R_{10}	750	5

Table 1. Grid environment example.

Task No.	Workload	No. of successors	Successors	Communication cost
0	0	3	1 2 3	0 0 0
1	35,000	2	4 5	14 12
2	7000	2	5 9	5 3
3	5000	2	9 13	7 2
4	28,000	3	6 7 11	11 17 6
5	20,000	3	8 11 14	13 11 13
6	18,000	3	9 10 14	11 16 13
7	4000	2	8 14	7 13
8	27,000	2	10 12	3 3
9	4000	1	12	17
10	19,000	1	13	3
11	32,000	2	12 13	4 4
12	31,000	1	15	13
13	16,000	1	15	11
14	22,000	1	15	3
15	0	0		

Table 2. A workflow application example of J14.

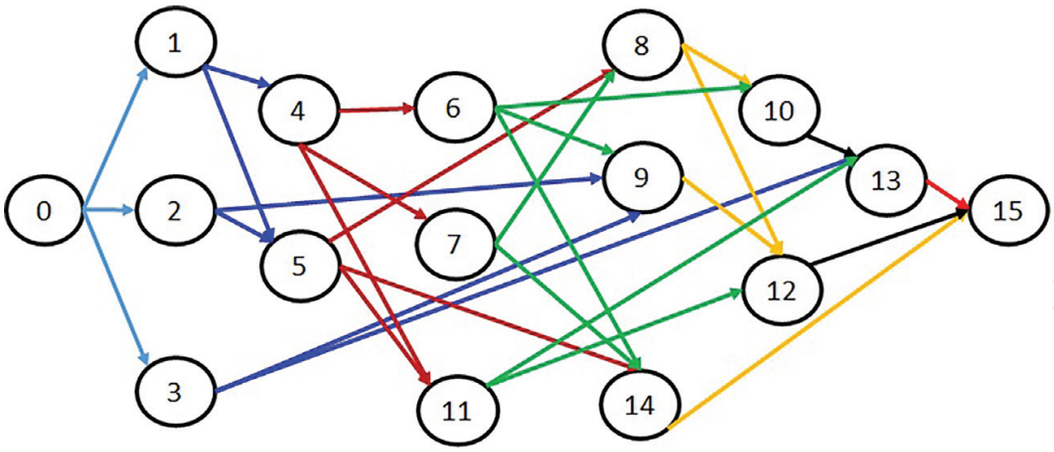

Figure 6. DAG of J14 example in **Table 2**.

where FIT_i indicates the fitness of instance i and $best_i$ is the best known solution of instance i. The |instances| denotes the number of instances of a problem case; it is 50 in this study. Hence, the *Dev*(%) represents the average deviation from the best known solutions which is the best solution found so far. The Avg.Dev(%) is the average of T trials; in this work, 10 trials ($T = 10$) were conducted.

The performance evaluation on J10 to J30 associates with different SGP values is displayed in **Table 3**. When the SGPs are 0 and 1, the averages of all problem cases' Avg.Dev are 12.43 and 12.05%; they are higher than other Avg.Dev results via other SGPs. This is because only global experience is referred to when SGP = 1 (exploitation only), and convergence of the algorithm in the process of searching in the vast solution space is premature; it is easy to get trapped on local optimums and impossible to obtain the global optimum. When SGP = 0 (exploration

SGP	J10	J12	J14	J16	J18	J20	J30	Avg.
0	3.86	6.61	7.87	15.55	13.11	16.28	23.70	12.43
0.1	3.88	6.58	7.32	15.43	12.14	14.31	17.27	10.99
0.2	4.44	6.87	7.67	15.95	12.39	14.03	15.93	11.04
0.3	4.18	6.74	8.11	15.93	12.46	14.62	15.13	11.02
0.4	4.27	7.27	8.63	16.49	12.54	15.25	14.92	11.34
0.5	4.55	7.26	8.36	16.52	12.49	14.73	15.22	11.30
0.6	4.69	7.82	8.68	16.74	12.72	14.88	15.62	11.59
0.7	4.60	7.39	8.55	17.35	13.21	14.92	15.41	11.63
0.8	4.93	7.50	8.68	17.40	13.13	15.40	14.61	11.66
0.9	4.97	7.77	8.94	16.78	13.38	15.63	15.53	11.86
1	4.79	7.66	9.06	17.36	13.63	16.13	15.72	12.05

Table 3. Performance evaluation on J10 to J30.

only), only local experience is referred to, and slow convergence exhibits while searching in a vast solution space, i.e., obtained optimal solution demands much more iterations.

Meanwhile, the obtained minimum Avg.Dev corresponding to the SGP for each problem case is shown in **Table 4**. The SGP of minimum Avg.Dev for J10 is zero, which is because the best solution in small solution space can be found by exploration search only. The SGPs of minimum Avg.Dev for J12 to J18 and J20 are 0.1 and 0.2, respectively. That is, more exploration search is adequate for small- and medium-scale problems. Since J30 is much more complex than J20, the solution space is vast. Thus, the SGP of the minimum *Avg.Dev* for J30 is 0.8 indicating that more exploitation search for large-scale problems is desired. Restated, the small-scale problem requires smaller SGP, and the large-scale problem needs larger SGP to obtain optimal solution.

Moreover, the comparisons between conventional PSO (ω = 0.8) and the proposed SGPSO (χ = 0.72984, SGP = 0.1) are provided in **Table 5**. **Table 5** indicates that SGPSO outperforms conventional PSO; SGPSO obtains the minimum Avg.Dev for most problem cases except the largest-scale problem (J30). The particle of conventional PSO swarm refers both global and individual experiences to move, i.e., involving both exploitation and exploration during solution search. However, the particle of the SGPSO swarm with SGP = 0.1 mostly refers to individual experience to be the moving guidance, i.e., exploration is carried out during solution search. As concluded above, larger SGP is required to obtain the optimal solution for large-scale problems; hence, SGP = 0.1 conducting more exploration would cause slow convergence. Therefore, when SGP = 0.8 is applied for solving J30, the resulting Avg.Dev (14.61%) is lower than that (15.92%) by using the conventional PSO as shown in **Table 5**.

A resulting schedule of the corresponding DAG of the workflow application scheduling instance with 14 tasks (**Figure 6**) is displayed in **Figure 7**. The fitness evolution of the J14 instance with different SGPs is displayed in **Figure 8**.

Additionally, to further realize the search behavior of the proposed scheme, the diversity evolution of the proposed SGPSO is checked. **Figure 9** displays the diversity evolution of a J14 instance with SGP = 0, SGP = 0.1, SGP = 0.8, and SGP = 1. The diversity to be checked in this study is defined as in Eq. (9).

	J10	J12	J14	J16	J18	J20	J30
SGP	0.0	0.1	0.1	0.1	0.1	0.2	0.8
Avg.Dev	3.86	6.58	7.32	15.43	12.14	14.03	14.61

Table 4. Minimum Avg.Dev corresponds to the *SGP*.

	J10	J12	J14	J16	J18	J20	J30	Avg.
SGPSO	3.88	6.58	7.32	15.43	12.14	14.31	17.27	10.99
PSO	4.52	7.42	8.72	20.55	13.12	15.12	15.92	12.20

Table 5. Comparisons between conventional PSO and SGPSO (*SGP* = 0.1).

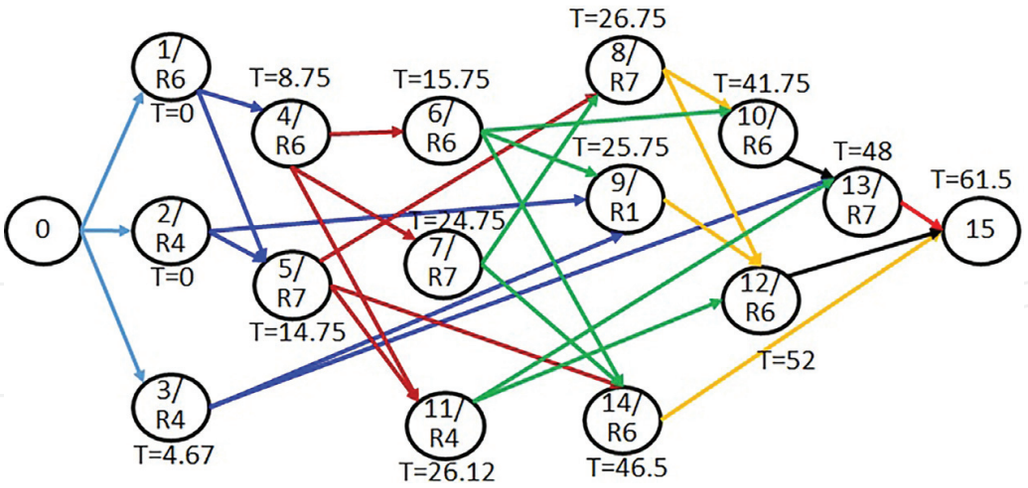

Figure 7. Workflow schedule of the J14 instance.

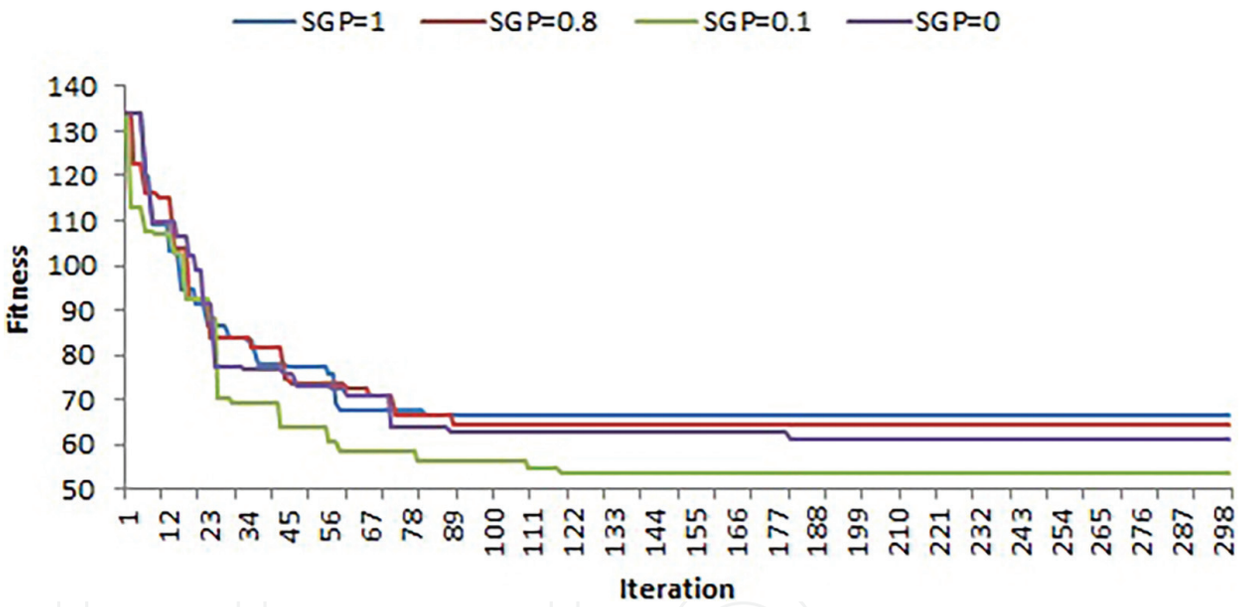

Figure 8. Fitness evolutions on J14 with different SGPs.

In **Figure 9**, the diversity remains high until the end of the operation when SGP = 0 without referencing global experience. Restated, particle search behavior keeps exploration until the end and hence suffers slow convergence. Conversely, the diversity quickly drops to none for SGP = 1, i.e., the particles go to exploitation search and therefore lead to premature convergence and trap on local optima. When SGP = 0.8, the behavior is similar to that of SGP = 1 but provides some exploration ability. Hence, SGP = 0.8 has the higher diversity than that of SGP = 1. With the setting of SGP = 0.1, high diversity in the early stage and diversity gradually lowered after the middle stage to the end are provided. Therefore, giving global search in the early stage gradually shrinks the search area in the later stage, hence providing an ideal search process from exploration to exploitation for finding the optimal schedule.

Finally, this work tested the workflow application problems examined in [15] to verify the proposed method. The comparison is made mainly with the largest-scale case in [15] which involves 15 tasks (represented here as J15).

The settings of the parameters in constriction and discrete PSOs are $\chi = 0.72984$, $\omega = 1$, and $c1 = c2 = 2$. The values of different SGP parameters SGP = {0, 0.1, 0.2, …, 0.9, 1} were tested. The minimum average fitness of the simulation results of iteration = 300 is shown in **Figure 10**.

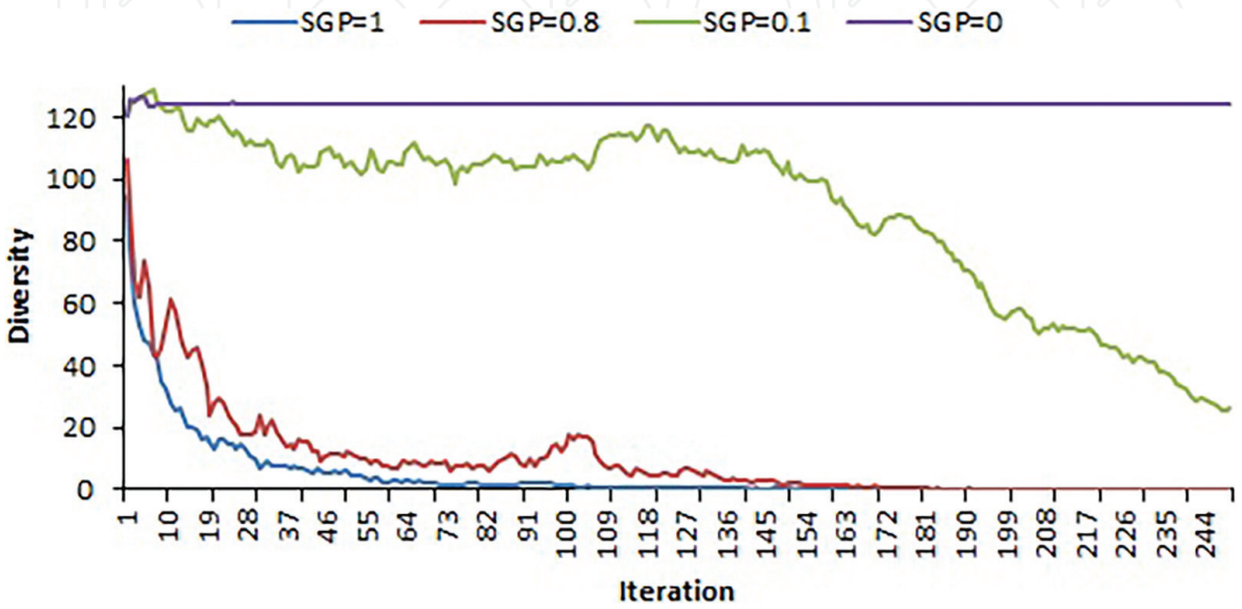

Figure 9. The diversity evolutions of the swarm of a J14 instance with different SGPs.

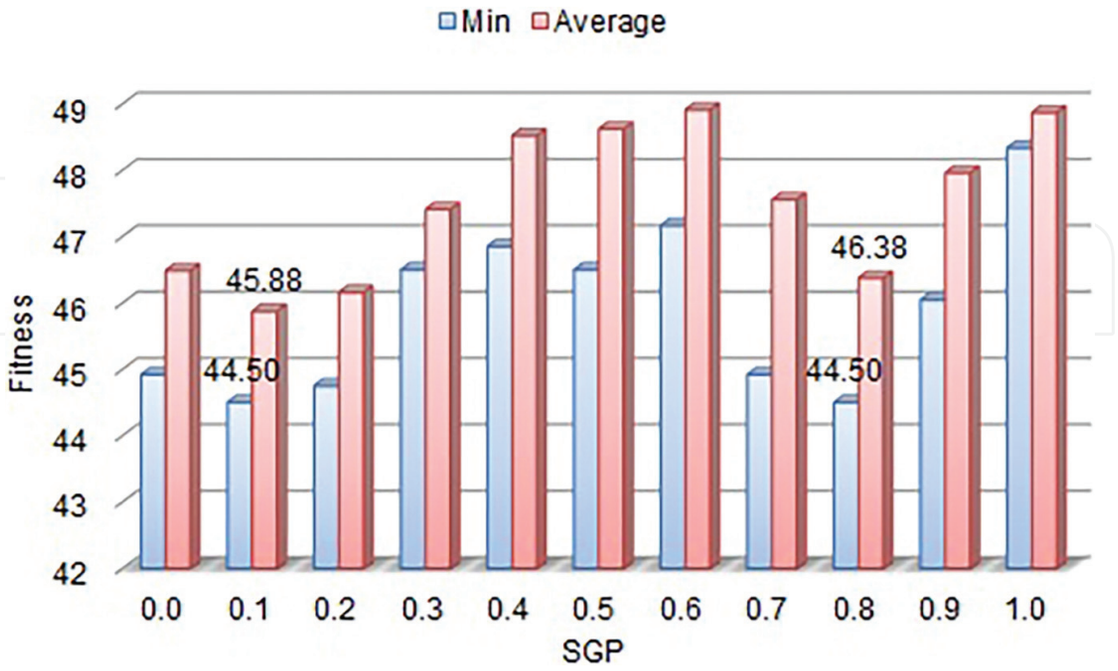

Figure 10. The minimum average fitness of J15.

Figure 10 shows that the minimal fitness (44.5) can be obtained when SGP = 0.1 and SGP = 0.8. However, the minimum average fitness (45.88) is obtained when SGP = 0.1. The results coincide with the above experiment consequences. The comparison between this work and [15] is listed in **Table 6**.

According to **Table 6**, with a small SGP value, the task scheduling outcomes of J15 will become better as the number of iterations increases (100 iterations → 300 iterations). This is because small SGP values tend to lead to the use of individual experience. In other words, the particles perform exploration search in solution space. In consequence, to obtain the optimums in the vast solution space will consume much time, and the convergence is delayed.

		Chen [15]	**Chen [15]**	**This work**
		χ = 0.75	**χ = 0.5**	**χ = 0.72984, SGP = 0.1**
100 iter.	Min.	45.50	44.50	46.75
	Avg.	60.15	54.26	47.71
300 iter.	Min.	44.50	44.50	44.50
	Avg.	55.45	50.36	45.88

Table 6. Performance comparison on J15 in [15].

5. Conclusions

Workflow application is the most common application in the grid. However, the workflow scheduling heavily affects the performance of workflow execution application. Two PSOs were used to solve task-resource matching and task execution priority subproblems of the workflow scheduling. A new and simplified velocity update rule extended from the ACO state transition rule is designed in constriction PSO for solving the task execution priority subproblem. Restated, the search control is based on a suggested SGP inspired by the ACO's transition rule. This constriction PSO-based algorithm is named stochastic greedy PSO (SGPSO), which provides both exploration and exploitation abilities during search. The main purpose is to strengthen the exploration capacity of the PSO in the solution search process while providing certain exploitation capability to avoid getting trapped in local optimums.

According to experimental results as indicated in **Table 3**, high SGP provides global experience guidance and causes premature convergence, hence easy to trap on local optimal such as only exploitation applied in SGP = 1.0 and Avg.Dev = 12.05% yielded. When SGP = 0, the algorithm would conduct self-search such that only exploration is enabled that causes slow convergence and Avg.Dev = 12.43% obtained. Better solutions can be found while providing enough exploration and certain exploitation capabilities such as SGP = 0.1; the lowest Avg. Dev = 10.99% can be obtained.

By using the SGP to control the search behavior, either exploitation or exploration would make the algorithm simplified and also reduce the execution time.

Meanwhile, high diversity in the early stage and low diversity in the later stage are preferred for searching in the solution space provided as indicated in **Figure 9**. Therefore, the proposed SGPSO with lower SGP is suggested for solving workflow class scheduling problem in the grid.

Unlike in [15], using heuristics to find initial solutions is not adopted in this work. Therefore, there is no need to consider which heuristic should be designed to increase performance, and hence the algorithm is thus easier to implement. In [15], the best result comes with the constriction factor $\chi = 0.5$ which was obtained after thorough testing. However, the best result can be yielded with the commonly suggested value $\chi = 0.72984$. Hence, the laborious work of finding the best constriction factor value is eliminated.

The experimental results show that the proposed method can effectively solve grid task scheduling problems and boost grid performance. In reality, there are many problems similar to grid task scheduling problems, such as the multimode resource-constrained project scheduling problems (MRCPSPs). In the future, the method proposed in this study will be applied to find solutions to MRCPSP-type problems.

Acknowledgements

This work was partly funded and supported by the Ministry of Science and Technology, Taiwan, under contract MOST 104-2221-E-167-011.

Conflicts of interest

The authors declare no conflict of interest.

Author details

Ruey-Maw Chen[1]* and Yin-Mou Shen[2]

*Address all correspondence to: rmchen@mail.ncut.edu.tw

1 Department of Computer Science and Information Engineering, National Chin-Yi University of Technology, Taiwan

2 Department of Information Management, Kun Shan University, Taiwan

References

[1] Beynon M, Sussman A, Catalyurek U, Kurc T, Saltz J. Performance optimization for data intensive grid applications. In: Proceedings of the Third Annual International Workshop on Active Middleware Services, San Francisco, CA, USA, Aug 2001, AMS'01. Los Alamitos, USA: IEEE Computer Society Press

[2] Goux JP, Kulkarni S, Linderoth J, Yoder M. An enabling framework for master-worker applications on the computational grid. In: Proceedings of The Ninth International Symposium on High-Performance Distributed Computing, Pittsburgh, USA, Aug 1–4, 2000. Los Alamitos, USA: IEEE Computer Society Press

[3] Xhafa F, Paniagua C, Barolli L, Caballé S. A parallel grid-based implementation for real time processing of event log data in collaborative applications. International Journal of Web and Grid Services. 2010;**6**(2):124-140

[4] Chang RS, Lin CY, Lin CF. An adaptive scoring job scheduling algorithm for grid computing. Information Sciences. 2012;**207**:79-89

[5] Fidanova, S. Simulated annealing for grid scheduling problem. In: Proceedings of the IEEE John Vincent Atanasoff 2006 International Symposium on Modern Computing (JVA'06), Sofia, Bulgaria, Oct 3–6, 2006. Los Alamitos, USA: IEEE Computer Society Press

[6] Gao Y, Rong H, Huang JZ. Adaptive grid job scheduling with genetic algorithms. Future Generation Computer Systems. 2005;**21**:151-161

[7] Liu H, Abraham A, Hassanien AE. Scheduling jobs on computational grids using a fuzzy particle swarm optimization algorithm. Future Generation Computer Systems. 2010; **26**(8):1336-1343

[8] Lorpunmanee S, Sap MN, Adbullah AH, Chai C. An Ant Colony Optimization for dynamic job scheduling in grid environment. Engineering and Technology. 2007;**1**(5): 314-321

[9] Salimi R, Motameni H, Omranpour H. Task scheduling using NSGA II with fuzzy adaptive operators for computational grids. Journal of Parallel and Distributed Computing. 2014;**74**(5):2333-2350

[10] Chen RM, Shen YM, Wang CT. Ant Colony Optimization inspired swarm optimization for grid task scheduling. In: Proceedings of the 2016 International Symposium on Computer, Consumer and Control (IS3C 2016), Xi'an, China, July 4–6, 2016. Los Alamitos, USA: IEEE Computer Society Press

[11] Chen RM, Shih HF. Solving university course timetabling problems using constriction particle swarm optimization with local search. Algorithms. 2013;**6**:227-244

[12] Li D, Deng N. Solving permutation flow shop scheduling problem with a cooperative multi-swarm PSO algorithm. Journal of Information & Computational Science. 2012;**9**(4): 977-987

[13] Chen RM, Shen YM. Novel encoding and routing balance insertion based particle swarm optimization with application to optimal CVRP depot location determination. Mathematical Problems in Engineering. 2015;**2015**:11

[14] Tao Q, Chang HY, Yi Y, Gu CQ, Li WJ. A rotary chaotic PSO algorithm for trustworthy scheduling of a grid work flow. Computers & Operations Research. 2010;**38**(5):824-836

[15] Chen RM, Wang CM. Project scheduling heuristics-based standard PSO for task-resource assignment in heterogeneous grid. Abstract and Applied Analysis. 2011;**2011**:20

[16] Dorigo M, Gambardella LM. Ant colony system: A cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation. 1997;**1**(1): 53-66

[17] Kennedy J, Ebrhart RC. Particle swarm optimization. In: Proceedings of the 4th IEEE International Conference on Neural Networks, 1995, Perth, Australia, Nov 27–Dec 1, 1995. Piscataway, NJ: IEEE Service Center; 1995

[18] Bratton D, Kennedy J. Defining a standard for particle swarm optimization. In: Proceedings of the IEEE Swarm Intelligence Symposium (SIS'07), Honolulu, UAS, Apr 1–5, 2007. Los Alamitos, USA: IEEE Computer Society Press

[19] Kennedy J, Ebrhart RC. A discrete binary version of the particle swarm Algorithm. In: Proceedings of The IEEE International Conference on Systems, Man, and Cybernetics, Orlando, USA, Oct 12–15, 1997. Vol. 5. Los Alamitos, USA: IEEE Computer Society Press

[20] Dorigo M, Maniezzo V, Colorni A. The ant system: Optimization by a colony of cooperating agents. IEEE transaction on system. Man and Cybernetics. 1996;**26**(1):1-13

[21] Hlaing ZCSU, Khine MA. Solving traveling salesman problem by using improved ant Colony optimization algorithm. International Journal of Information and Education Technology. 2011;**1**(5):404-409

[22] Chen D, Wang J, Zou F, Hou W, Zhao C. An improved group search optimizer with operation of quantum-behaved swarm and its application. Applied Soft Computing. 2012;**12**(2):712-725

IntechOpen