1

Application of Discrete Particle Swarm Optimization for Grid Task Scheduling Problem

Ruey-Maw Chen
National Chin-Yi University of Technology, Taichung, 411, Taiwan, R. O. C.

1. Introduction

Many applications involve the concepts of scheduling, such as communications, packet routing, production planning [Zhai et al., 2006], classroom arrangement [Mathaisel & Comm, 1991], aircrew scheduling [Chang, 2002], nurse scheduling [Ohki et al., 2006], food industrial [Simeonov & Simeonovova, 2002], control system [Fleming & Fonseca, 1993], resource-constrained scheduling problem [Chen, 2007] and grid computing. There are many different types of scheduling problems such as real-time, job-shop, permutation flow-shop, project scheduling and other scheduling problems have been studied intensively. However, in this work, the studied grid task scheduling problem is much more complex than above stated classic task scheduling problems. Restated, a grid application is regarded as a task scheduling problem involving tasks with inter-communication and distributed homogeneous or heterogeneous resources, and can be represented by a task interaction graph (TIG).

Grid is a service for sharing computing power and data storage capacity over the Internet. The grid systems outperform simple communication between computers and aims ultimately to turn the global network of computers into one vast computational resource. Grid computing can be adopted in many applications, such as high-performance applications, large-output applications, data-intensive applications and community-centric applications. These applications major concern to efficiently schedule tasks over the available multi-processor environment provided by the grid. 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 [Foster et al., 2001]. The efficiency and effectiveness of grid resource management greatly depend on the scheduling algorithm [Lee et al., 2007]. Generally, in the grid environment, these resources are different over time, and such changes will affect the performance of the tasks running on the grid. In grid computing, tasks are assigned among grid system [Salman, 2002]. The purpose of task scheduling in grid is to find optimal task-processor assignment and hence minimize application completion time (total cost). Most scheduling problems in these applications are categorized into the class of *NP-complete* problems. This implies that it would take amount of computation time to obtain an optimal solution, especially for a large-scale scheduling problem. A variety of approaches have been applied to solve scheduling problems, such as simulated annealing (SA) [Kirkpatrick, 1983], neural network [Chen, 2007], genetic algorithm (GA) [Holland, 1987], tabu search (TS) [Glover, 1989; Glover, 1990], ant colony optimization (ACO) [Dorigo & Gambardella, 1997] and particle swarm optimization [Kennedy & Eberhart, 1995]. Among above stated schemes, many PSO-based approaches were suggested for solving different scheduling application problems including production scheduling [Watts & Strogatz, 1998], project scheduling [Chen, 2010], call center scheduling [chiu et al., 2009], and others [Behnamian et al., 2010; Hei et al., 2009].

In light of different algorithms studied, PSO is a promising and well-applied meta-heuristic approach in finding the optimal solutions of diverse scheduling problems and other applications. The particle swarm optimization (PSO) is a swarm intelligent inspired scheme which was first proposed by Kennedy and Eberhart [Kennedy & Eberhart, 1995]. In PSO, a swarm of particles spread in the solution search space and the position of a particle denotes a solution of studied problem. Each particle would move to a new position (new solution) determined by both of the individual experience (particle individual) and the global experience (particle swarm) heading toward the global optimum. However, many PSO derivatives have been studied, and one of them was named "discrete" particle swarm optimization (DPSO) algorithm proposed by Kennedy et al. [Kennedy & Eberhart, 1997] representing how DPSO can be used to solve problems. Hence, this study focuses on applying discrete particle swarm optimization algorithm to solve the task scheduling problem in grid computing.

To enhance the performance of the applied DPSO, additional heuristic was introduced to solve the investigated scheduling problem in grid. Restated, simulated annealing (SA) algorithm was incorporated into DPSO to solve task assignment problem in grid environment. Moreover, the resulting change in position is defined by roulette wheel selection rule rather than the used rule in [Kennedy & Eberhart, 1997]. Furthermore, the dynamic situations are not considered; the distributed homogeneous resources in grid environment are considered in this investigation.

2. The task scheduling problem in grid computing

There are different grid task scheduling problems exist including homogeneous and heterogeneous architectures. This section gives a class of task scheduling problem in homogeneous grid. Definition, limitation and objective of a grid computing system are presented. The introduced grid task scheduling problem can be represented as a task interaction graph (TIG) proposed by Salman *et.al.* [Salman, 2002], as displayed in Fig. 1. Figure 1 presents available memory in homogeneous grid, task processing time, memory requirement of each task, data exchange between tasks, and communication cost between grids. Meanwhile, two possible solutions are displayed in Fig. 2.

Grid #	1	2	3	4
Memory available	25	40	30	25

Task #	1	2	3	4	*5*
Process time	15	10	20	30	*15*
Memory requirement	20	30	25	20	*10*

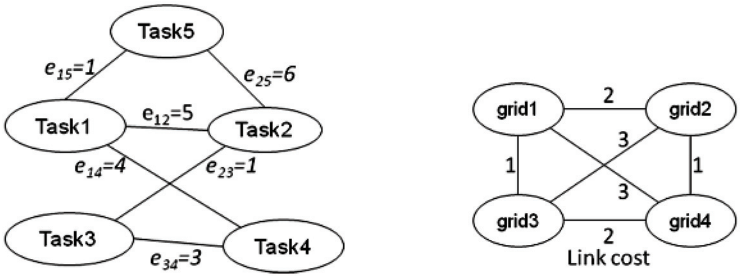

Fig. 1. Grid task scheduling problem representation [Salman et al., 2002]

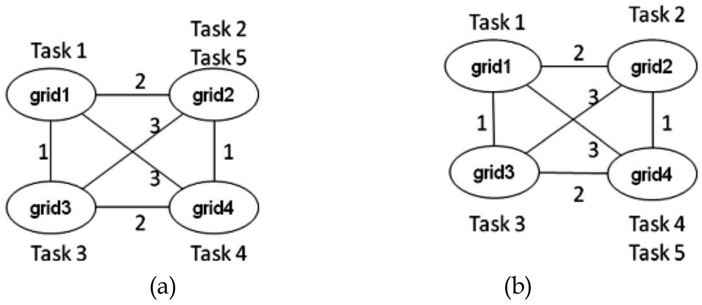

Fig. 2. Possible solutions

A meta-heuristic algorithm that is based on the principles of discrete particle swarm optimization (PSO) is proposed for solving the grid scheduling problem. The traditional assignment problems are only concerned to minimize the total processing cost, and there is no communication cost between these tasks and grids.

In homogeneous system, a grid environment can be represented as task interaction graph (TIG), G (V, E), where $V \in \{1, 2, \ldots, M\}$ is the set of the tasks and E are the interactions between these tasks as in Fig. 1. The M and N are the total number of tasks and the total number of grids (resources), respectively. The total amount of transmission data weight e_{ij} denotes the information exchange (interactive) data between tasks i and j. The p_i is the processing time (cost) corresponding to the work load to be performed by task i on grid. In the example of TIG problem shown in Figure 1, the tasks 2 and 5 are processed on the grid 2. Restated, no communication cost exists between these two tasks (task 2 and 5). Additionally, each task i has memory requirement m_i to be processed on one grid, and each grid requires enough memory to run their tasks.

For example, the processing time of task 1 is 15 and scheduled on grid 1. The task 1 has to exchange data with the tasks 2, 4 and 5. However, the tasks 2, 4 and 5 are on different grids, this means that there are communication costs with task 1. Furthermore, tasks 4 and 5 are on the same grid and there is no communication cost required between them. Therefore, the total cost for grid 1 of possible solution case (a) is (15) + (5 × 2 + 1 × 2 + 4 × 3) = 39. Moreover, task 1 satisfies the memory constraint; that is, the memory requirement is 20 for task 1, which is less than the memory available of 25 for grid 1. The communication cost is computed by the communication cost (link cost) multiplies the edge weight (exchange data). The total cost for grids of different possible solutions as demonstrated in Fig. 2 are determined as follows.

Grids	Total cost – case (a)	Total cost – case (b)
Grid 1	(15)+(5×2+1×2+4×3)=39	(15)+(5×2+4×3+1×3)=40
Grid 2	(10+15)+(5×2+1×3)+(1×2)=40	(10)+(5×2+1×3+6×1)=29
Grid 3	(20)+ (1×3+3×2)=29	(20)+ (1×3+3×2)=29
Grid 4	(30)+ (4×3+3×2)=48	(30+15)+(4×3+3×2)+(1×3+6×1)=72

A grid application completion time is defined as the latest time that grid finishes all scheduled tasks processing and communication. According to above total cost calculation, different task assignment in grid would obtain different application completion time. For example, case (b) solution of Fig. 2 yields application completion time 72; case (a) solution of Fig. 2 has less application completion time 48. Restated, the resulting schedule of case (a) is better than that of case (b).

The grid system can be represented as a grid environment graph (GEG) G (P, C), where $P = \{1, 2, . . . , N\}$ is the set of grids in the distributed system. The C represents the set of communication cost between these grids. The d_{ij} between grids i and j represents the link cost between the grids. The problem of this study is to assign these tasks in V to the set of grids P. The objective function is to minimize the maximum total execution time required by each grid and the communication cost among all the interactive tasks that satisfies the memory constraint on different grids. The problem can be defined as:

$$\text{Minimize } \{\max (C_{exe}(k)+C_{com}(k))\},\ k\in\{1, 2, . . . , N\} \quad (1)$$

Where

$$C_{exe}(k) = \sum_{i\in A_k} p_i \quad , A_k \text{ is the set of tasks assigned to grid } k \quad (2)$$

$$C_{mem}(k) = \sum_{i\in A_k} m_i \quad , A_k \text{ is the set of tasks assigned to grid } k \quad (3)$$

$$C_{com}(k) = \sum_{i\in A_k} \sum_{j\notin A_k} d_{kp}\cdot e_{ij} \quad ,\text{for all grids } p \neq k,\ p=1 \text{ to } N;\ i, j=1 \text{ to } M \quad (4)$$

Subject to

$$C_{mem}(k) \leq MemAvail(k) \quad (5)$$

Where $C_{exe}(k)$ is the total execution time of all tasks assigned to grid k and $C_{com}(k)$ is the total communication cost between tasks assigned to grid k. Those relative tasks are assigned to other grids in an assignment. The $C_{mem}(k)$ is the total memory requirement of all tasks assigned to grid k, for which the value of $C_{mem}(k)$ have to less than or equal than the total available memory of grid k; *MemAvail(k)* as listed in Eq. (5). The objective of the task assignment problem is to find an assignment schedule that the cost is minimized of one grid for a given TIG on a given GEG. In this study, the penalty function is adopted in the proposed algorithms.

$$\text{Penalty(k)} = \text{C}_{\text{mem}}\text{(k)} - \text{MemAvail(k)} \quad (6)$$

In Eq. (6), the *penalty(k)* is set to zero if the constraint of Eq. (5) is satisfied.

3. Particle swarm optimization

The particles swarm optimization (PSO) was first proposed by Kennedy and Eberhart in 1995. The original PSO is applied in real variable number space. There are a lot of task-resource assignment related works have been introduced in recent years [Kuo et al. 2009; Sha & Hsu, 2006; Bokhari, 1987; Chaudhary & Aggarwal, 1993; Norman & Thanisch, 1993]. These works indicated that the problems are set in a space featuring of continuous. However, the combinatorial problems are most of discrete or quantitative variables [Liao et al., 2007].PSO schematic diagram is displayed as in Fig. 3. The introduced grid task scheduling problem as in Fig. 1 can be regarded as a task-grid assignment problem in a graph as in Fig. 2.

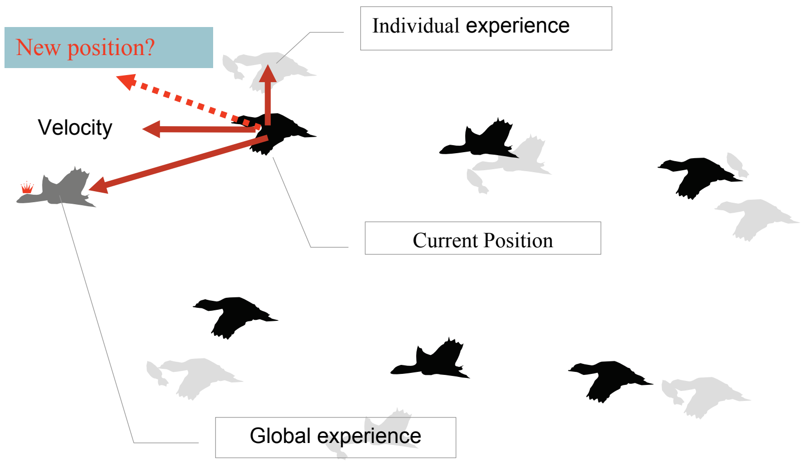

Fig. 3. PSO schematic diagram

The particle swarm optimization is a multi-agent general meta-heuristic method, and can be applied extensively in solving many NP-complete or combinatorial problems. The PSO consists of a swarm of particles in the search space; the position of a particle is indicated by a vector which presents a potential solution of the problem. PSO is initialized with a population of particles (randomly assigned or generated by heuristic) and searches for the best position (solution or schedule) with the best fitness. In every generation or iteration, the local bests and global best are determined through evaluating the performances, i.e., the fitness values of current population of particles. A particle moves to a new position obtaining a new solution guided by the velocity (a vector). Hence, the velocity plays an important role in affecting the characters of creating new solution. There are two experience positions are used in the PSO; one is the global experience position of all particles, which memorizes the global best solution obtained from all positions (solutions) of all particles; the other is the individual experience position of each particle, which memorizes the local best solution acquired from the positions (solutions) of the corresponding particle has been at. These two experience positions and the inertia weight of the previous velocities used to determine the impact on the current velocity. The velocity retains part of prior velocity (the inertia) and driving particle toward the direction based on the global experience position and the individual experience position. Thus, the particles can derive new positions (solutions) by their own inertia and experience positions.

In traditional PSO, the search space (solution space) is D dimension space (the number of dimension is corresponding to the parameters of solutions) and the population consists of N_p particles. For the ith particle ($i = 1, \ldots, N_p$), the position consists of M components $X_i = \{X_{i1}, \ldots, X_{iM}\}$, X_{ij} is the jth component of the ith position. The velocity $V_i = \{V_{i1}, \ldots, V_{iM}\}$, where V_{ij} is the velocity component corresponding to the component of X_{ij}, and the individual experience is a position $L_i = \{L_{i1}, \ldots, L_{iM}\}$ which is the local best solution for the ith particle. Additionally, $G = \{G_1, \ldots, G_M\}$ represents the global best experience shared among all particles achieved so far. The mentioned parameters above are used to calculate the updating of the jth component of the position and velocity for the ith particle, as shown in Eq. (7).

$$\begin{cases} V_{ij}^{new} = wV_{ij} + c_1 r_1 (L_{ij} - X_{ij}) + c_2 r_2 (G_j - X_{ij}) \\ X_{ij}^{new} = X_{ij} + V_{ij}^{new} \end{cases} \tag{7}$$

Where w is an inertia 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 either closes to the individual experience position or global experience position respectively. Furthermore, the r_1 and r_2 are the random numbers uniformly distributed in [0, 1], influencing the tradeoff between the global exploitation (based on swarm's best experience) and local exploration (based on particle's best experience) abilities during search.

4. Simulated annealing algorithm

Other meta-heuristics are usually combined into PSO to increase the problem solving performance. SA is one of the popular algorithms to be combined with other meta-heuristic schemes. Simulated annealing (SA) was first introduced by Metropolis in 1953 [Metropolis et al., 1953]. Meanwhile, SA is a stochastic method for combinatorial problem optimization. Furthermore, SA is one of the efficient methods applied to solve widely complex problems [Kirkpatrick, 1983]. The original SA procedure is listed as shown in Fig. 4.

```
Initial solution $S$, compute corresponding energy $E$
Set the initial temperature ($T$), cooling rate ($r$)
While $E$ <> 0
        $S'$ = Generate the new solution
        Compute new energy $E'$corresponding to $S'$ and calculate $\Delta E = E' - E$
        If $\Delta E<0$ then accept $S = S'$, $E = E'$
        Else Compute the $\delta = e^{-(\Delta E / T)}$
            Accept the new solution when random number < $\delta$
   Decrease the temperature $T = T \times r$
```

Fig. 4. Simulated annealing algorithm

In Fig.4, the energy E is corresponding to solution S, and energy E' is correlated to solution S'. However, energy definition is determined by the studied problem. Hence, E is defined as $\{\max(C_{exe}(k)+C_{com}(k))\}+Penalty(k)$, $k \in \{1, 2, \ldots, N\}$ in this investigation. The temperature, T, is the magnitude of fluctuation; it is a key parameter in controlling the search direction as well as the step size toward the global minimum. The applied cooling schedule is controlled by

T=*T*×*r*. The acceptance criterion of worse solution is based on the probabilistic process which is dependent on the temperature and energy difference between two states. Restated, the probability is determined by $\delta=exp(-\Delta E/T)$.

5. Discrete particle swarm optimization method

Kennedy and Eberhart developed a discrete version of PSO in 1997 [Kennedy & Eberhard, 1997]. The discrete PSO essentially differs from the original PSO in two characteristics. Firstly, the particle is composed of the binary variable. Secondly, the velocity represents the probability of the binary variable taking the value of one, i.e., the probability of task *i* is assigned to grid *k* in this study ($i\in \{1, 2, . . . , M\}$; $k\in\{1, 2, . . . , N\}$). The discrete PSO is adopted by generating solutions for updating the particle's position and velocity vectors to solve the task scheduling problem in parallel machines [Kashan & Karimi, 2009; Kashan et al., 2008; Lee et al., 2006]. Another similar to the discrete PSO optimization technique developed by Laskari et al. [Laskari et al., 2002], which is based on the truncation of the real values to their nearest integer. In this study, employed discrete PSO equations were introduced by Kennedy and Eberhard for solving the task assignment problem. The discrete PSO was also applied to solve the flowshop scheduling problem, and performed well in the computation result. This study conducts the discrete PSO method introduced by [Liao et al., 2007] and combines the SA algorithm for solving the task assignment problems in grid. The task-grid assignment problem will be then introduced.
Assumes there are N_p particles, and each particle searches for $D = M\times N$ dimension space (the number of tasks and grids). For the hth particle ($h =1, . . . , N_p$), the position consists of $M\times N$ components $X_h = \{X_{h11}, . . . , X_{hMN}\}$, $X_{hij} \in\{0,1\}$is the ith task assigned to grid j for particle h ($i =1, . . . , M$; $j =1, . . . , N$). The velocity $V_h = \{V_{h11}, . . . , V_{hMN}\}$, where V_{hij} is the velocity associated with component X_{hij}, and the individual experience for particle h is $L_h = \{L_{h11}, \ldots, L_{hMN}\}$, the local best solution for the hth particle. Additionally, $G = \{G_{11},\ldots, G_{MP}\}$ represents the global best experience obtained and shared among all the population of particles. Above stated parameters are then used to update all components of the V_h. The velocity components updating for the hth particle is shown as in Eq. (8).

$$V_{hij}^{\;new} = wV_{hij} + c_1r_1(L_{hij} - X_{hij}) + c_2r_2(G_{ij} - X_{hij}) \tag{8}$$

According to Eq. (8), each particle moves to new position according to its new velocity. However, the new position generation is not the same as in original PSO, Eq. (7). Kennedy and Eberhart claim that the higher velocity component value is more likely to choose 1 for the corresponding position component, while lower velocity component value favors the position component value of 0. Hence, a probability function is used as shown in Eq. (9).

$$s(V_{hij}) = \frac{1}{1+\exp(-V_{hij})} \tag{9}$$

Equation (9) is the sigmoid function as displayed in Fig. 5, where $s(V_{hij})$ is defined as representing the probability of X_{hij} to be set to 0 or 1. To avoid the value of $s(V_{hij})$ approaching 0 or 1, a constant V_{max} is used to limit the range of V_{hij}. In practice, V_{max} is often set at 4, i.e., $V_{hij} \in [-V_{max}, +V_{max}]$. After transformation via Eq. (9), $s(V_{hij})$ is mapped to a value between 0 and 1, i.e., $s(V_{hij}) \in (0, 1)$. For example, if V_{max} =4 then probabilities will be

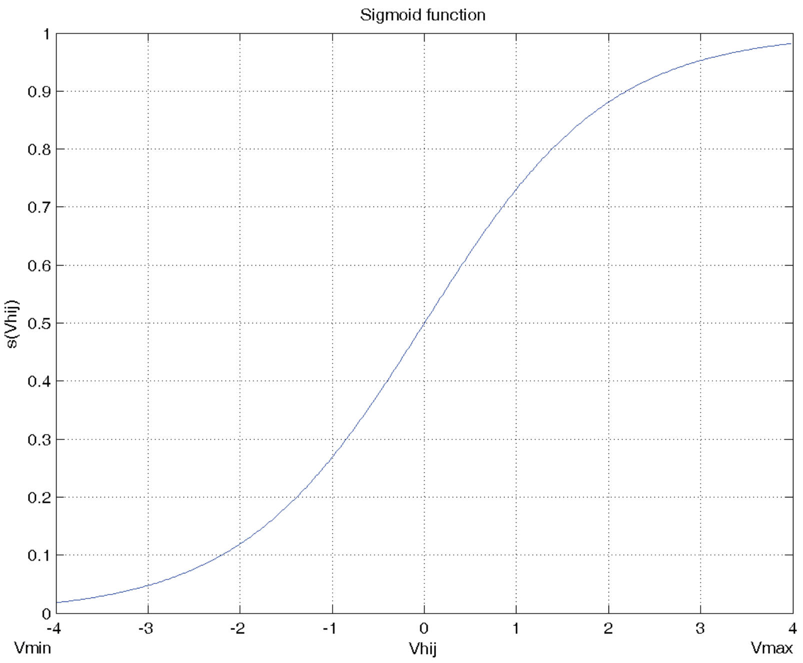

Fig. 5. Sigmoid function

limited to *s*(V_{hij}), between 0.9820 and 0.018. In [Kennedy & Eberhart, 1997], the resulting change in position is defined by the following rule (Eq. (10)).

$$\begin{cases} X_{hij} = 1, & \text{if } rand(\ ) < s(V_{hij}) \\ X_{hij} = 0, & \text{else} \end{cases} \tag{10}$$

Where the *rand( )* is a quasi-random number selected from a uniform distribution in [0.0, 1.0]. For task-grid assignment problem, each task can only be assigned to one grid. Therefore, in the proposed algorithm, each particle *h* places the unscheduled task *i* to grid *j* according to the following normalized probability [Liao et al., 2007]:

$$q_h(i,j) = \frac{s(V_{hij})}{\sum_{j \in U} s(V_{hij})}, \ U \text{ is the set of grids} \tag{11}$$

Restated, the determination of which grid to be assigned to an unscheduled task in the study is based on the roulette wheel selection rule which is well applied in genetic algorithm. Hence, according to roulette wheel selection rile, grid *j* is randomly selected from *U* for task *i* based on the probability distribution given by Eq. (11) and a generated random number. Based on the pseudo code of discrete PSO given by Kennedy et al. [Kennedy & Eberhard, 1997], the proposed algorithm is modified and showed in Fig. 6. The computation steps of the proposed algorithm in the simulation system can be summarized as:

1. Initialize the parameters and input the problem data.

2. Generate the initial particle solution, including velocity matrix (V_{N_pMN}), and then transform the velocity to a matrix of $s(V_{hij})$, and use Eq. (11) to generate the matrix (X_{N_pMN}), and update the local best and global best solution.
3. Use Eq. (8) to generate new velocity of particles for the next generation until a specified stopping criterion is reached.

```
Initialize and generate each particle solution of $X_h$ matrix and velocity $V_h$
Set $L_h = X_h$, $h=1,\dots, N_p$, $G = X_1$
Loop
//find the global best solution
For h= 1 to $N_p$
        If Z(Xh)<Z(G) then        // Z( ) objective function
                g = h  //g is the index of the global best G
End if
Next h
    For h= 1 to Np
         Update the velocity matrix Vh based on Eq. (8)
               subject to Vhij$\in$ [−Vmax,+Vmax]
               map Vhij to s(Vhij) based on Eq. (9)
               calculate normalized probability qh(i, j) using Eq. (11)
               select grid j for task i (Xhij) by roulette wheel selection rule
          Update the assignment matrix Xh based on Simulated annealing
        $\Delta E$ = Z(Xh)-Z(Lh)
        if $\Delta E$<0 then
              Lh = Xh
else
              Compute the $\delta = e^{-(\Delta E/T)}$
              Lh = Xh  when a generated random number Pa<$\delta$
  End if
Next h
// find the local best solution
    For h = 1 to Np
             If Z(Xh)<Z(Lh) then           // Z( ) objective function
                  Lh = Xh  // Lh is the best so far for particle h
End if
Next h
     decrease the temperature T
Until the end of criterion is reached
```

Fig. 6. The proposed discrete PSO combined with SA

5.1 DPSO encoding representation

Encoding the task assignment problem in grid into the position vector of particle is necessary. Hence, encoding is illustrated by an example as follows. For example, there are 5 tasks to be distributed to 3 grids; the initial velocity for particle h (V_{hij}) is

Task \ Grid	1	2	3
1	-1.2	-3.9	3.1
2	1.1	-2	-2.8
3	1.2	-2.6	3.2
4	-0.2	-1.9	-2
5	3.8	-0.4	0.4

According to Eq. (8), the updated velocity (V_{hij}) becomes

Task \ Grid	1	2	3
1	-1.1	-3.5	2.8
2	1	-1.8	-2.5
3	1.1	-2.3	2.9
4	-0.2	-1.7	-1.8
5	3.4	-0.4	0.4

Then, the corresponding probability $s(V_{hij})$ is determined by Eq. (9) as follows.

Task \ Grid	1	2	3
1	0.25	0.03	0.94
2	0.73	0.14	0.08
3	0.75	0.09	0.95
4	0.45	0.15	0.14
5	0.97	0.4	0.6

Hence, the normalized probability $q_h(i, j)$ (based on Eq. (11)) for applying roulette wheel selection rule is

Task \ Grid	1	2	3
1	0.20	0.02	0.77
2	0.77	0.15	0.08
3	0.42	0.05	0.53
4	0.61	0.20	0.19
5	0.49	0.20	0.30

Where, $q_h(1, 1)=0.25/(0.25+0.03+0.94) \fallingdotseq 0.20$; $q_h(2, 1)=0.73/(0.73+0.14+0.08) \fallingdotseq 0.77$ and so forth. Finally, task-grid assignment based on roulette wheel selection rule will be

Task \ Grid	1	2	3
1	0	0	1
2	1	0	0
3	0	0	1
4	0	1	0
5	0	0	1

Restated, grid 1 would execute task 2; grid 2 services task 4; grid 3 is responsible for executing tasks 1, 3, and 5.

6. Experimental results

To verify the performance of the presented algorithm (SA + discrete PSO), some simulation cases will be tested. Simulations use the cases of 5 and 10 tasks in 4 grids and 5 grids respectively to verify the performance of the proposed algorithm. According to Fig. 1, the 5-task case only uses 4 grids and suffers the heavy loading from the computation effort in which the processing time is much more than the communication cost. For the 10-task case, the setting of processing time is in the range of [5, 10], and it needs more communications than the processing cost. Tables 1-4 demonstrate simulation data for 5-task case. Simulation data for 10-task case are listed in Tables 5-8. The other parameters used in this study are set as following: The temperature T was set to 100 and cooling scheduling was set to 0.99 ($r=0.99$), i.e., $T = T \times 0.99$, $w = 0.7$, $c_1 = c_2 = 1$ and $V_{max} = 4$. There are 10 particles involved in simulation tests. The interactive matrix is symmetrical matrix as the grid distance matrix.

Task #	1	2	3	4	5
Process time	15	10	20	30	15
Memory requirement	20	30	25	20	10

Table 1. Simulation data with 5 tasks

Grid #	1	2	3	4
Memory available	25	40	30	25

Table 2. Memory available for 4 grids

Task #	1	2	3	4	5
1	0	5	0	4	1
2	5	0	1	0	6
3	0	1	0	3	0
4	4	0	3	0	0
5	1	6	0	0	0

Table 3. Interaction cost matrix for 5 tasks

Task #	1	2	3	4
1	0	2	3	1
2	2	0	1	3
3	3	1	0	2
4	1	3	2	0

Table 4. Distance cost matrix for 4 grids

Task #	1	2	3	4	5	6	7	8	9	10
Process time	10	9	8	7	6	5	5	9	8	10
Memory requirement	10	15	10	20	10	10	20	15	20	10

Table 5. Simulation data with 10 tasks

Grid #	1	2	3	4	5
Memory available	100	90	130	90	100

Table 6. Memory available for 5 grids

Task #	1	2	3	4	5	6	7	8	9	10
1	0	1	2	3	4	5	5	4	3	4
2	1	0	5	1	2	3	4	3	5	4
3	2	5	0	3	2	1	0	1	2	3
4	3	1	3	0	4	5	4	3	2	1
5	4	2	2	4	0	1	2	3	4	5
6	5	3	1	5	1	0	4	3	2	1
7	5	4	0	4	2	4	0	2	3	1
8	4	3	1	3	3	3	2	0	5	1
9	3	5	2	2	4	2	3	5	0	2
10	4	4	3	1	5	1	4	1	2	0

Table 7. Interaction cost matrix for 10 tasks

Grid #	1	2	3	4	5
1	0	2	3	1	5
2	2	0	1	3	3
3	3	1	0	2	4
4	1	3	2	0	6
5	5	3	4	6	0

Table 8. Distance cost matrix for 5 grids

In the example of 5-task in 4 grids environment, the best solution can be found in Table 9. Table 9 indicates that task 4 is assigned to grid 4; task 1 is assigned to grid 1, and so on. The total processing cost for task 4 on grid 4 is 30. Based on Tables 3 and 4, task 4 has interaction with tasks 1 and 3, and the communication cost to tasks 1 and 3 is 1×4 + 2×3 = 10. Thus, the total cost for grid 4 is 48. Similarly, the obtained best solution of 10-task case is displayed in Table 10.

Task #	Grid #			
	1	2	3	4
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1
5	0	1	0	0

Table 9. The assignment results for 5 tasks in 4 grids with cost of 40

Task #	Grid #				
	1	2	3	4	5
1	0	1	0	0	0
2	0	0	1	0	0
3	0	0	1	0	0
4	0	1	0	0	0
5	0	1	0	0	0
6	0	0	1	0	0
7	0	0	1	0	0
8	0	0	1	0	0
9	0	1	0	0	0
10	0	0	1	0	0

Table 10. The assignment results for 10 tasks in 5 grids with cost of 138

Table 10 shows the result of 10-task case that more grids may not decrease the total cost due to the communication cost. Furthermore, extra tests were simulated; the total numbers of tasks were set 20 to 50, the processing times of tasks are uniform distribution in [5, 10] and the memory requirement is also uniform distribution in [50, 100]. The numbers of grids are from 6 to 15, the total available memory is uniform distribution in [200, 400]. The interactive data between of tasks are varying from 1 to 10, and the communications between grids are varied by uniform distribution from 1 to 10. Simulation results demonstrate that more iterations or number of particles obtain the better solution since more solutions were generated as displayed in Table 11.

(Task, Grid)	Number of particles				
	10		20		
	# of iterations		# of iterations		
	100	300	100	300	500
20,6	1967	1888	1946	1872	1818
20,8	1619	1611	1611	1608	1569
20,12	1308	1249	1263	1241	1163
30,8	3347	3325	3329	3325	3315
30,12	2514	2359	2470	2359	2359
30,15	1998	1998	1998	1957	1957
40,12	4276	4276	4276	4276	4276
40,15	4008	3808	3808	3562	3562
50,15	5157	5157	5741	5157	5156

Table 11. Simulation results

7. Summary

This study introduces the discrete PSO algorithm for solving task-grid assignment problem in a distributed grid environment. Experiment results indicate that the discrete version of PSO combining simulated annealing is effective for solving task-grid assignment problems. However, more complicated cases can be considered in-depth, such as more realistic examples and more tasks involved. For example, a grid system consists of heterogeneous grids (with different process capabilities) is given. Restated, more complicated scheduling problem can be further solved using discrete PSO. Moreover, to further improve the discrete PSO performance, some other heuristics are suggested to be included, for example, insertion, 2-opt or others. Further research encourages that the extension of the discrete PSO by incorporating other meta-heuristics for solving different scheduling problems is recommended.

8. References

Behnamian, J.; Zandieh, M. & Ghomi, S. M. T. F. (2010). Due windows group scheduling using an effective hybrid optimization approach. *International Journal of Advanced Manufacturing Technology*, Vol. 46, No. 5-8 (2010), pp. 721-735, ISSN (printed): 0268-3768.

Bokhari, S. H. (1987). *Assignment Problems in Parallel and Distributed Computing*, Kluwer Academic Publishers, ISBN: 0-89838-240-8, Boston (1987).

Chang, S.C. (2002). A new aircrew-scheduling model for short-haul routes. *Journal of Air Transport Management,* Vol. 8, No. 4 (July 2002), pp. 249-260, ISSN: 0969-6997.

Chaudhary, V. & Aggarwal, J. K. (1993). A generalized scheme for mapping parallel algorithms. *IEEE Transactions on Parallel and Distributed Systems*, Vol. 4, No. 3 (March 1993), pp. 328-346, ISSN: 1045-9219.

Chen, R. M.; Lo, S. T. & Huang, Y. M. (2007). Combining competitive scheme with slack neurons to solve real-time job scheduling problem. *Expert Systems with Applications,* Vol. 33, No. 1 (July 2007), pp. 75–85, ISSN: 0957-4174.

Chen, R. M.; Wu, C. L.; Wang, C. M. & Lo, S. T. (2010). Using novel particle swarm optimization scheme to solve resource-constrained scheduling problem in PSPLIB. *Expert systems with applications*, Vol. 37, No. 3 (March 2010), pp. 1899-1910, ISSN: 0957-4174.

Dorigo, M. & Gambardella, L. M. (1997). Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation,* Vol. 1, No. 1 (April 1997), pp. 53-66, ISSN: 1089-778X.

Fleming, P. J. & Fonseca, C. M. (1993). Genetic algorithms in control systems engineering: a brief introduction. *Proceedings of IEE Colloquium on Genetic Algorithms for Control Systems Engineering*, London UK, 28 May 1993, pp. 1/1 - 1/5.

Foster, I.; Kesselman, C. & Tuecke, S. (2001). The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *Int'l J. of High Performance Computing Applications*, Vol. 15, No. 3 (Fall 2001), pp. 200-222, ISSN (printed): 1094-3420.

Glover, F. (1989). Tabu Search – Part I, *ORSA Journal on Computing*, Vol. 1, No. 3 (Summer 1989), pp. 190–206, ISSN: 0899-1499.

Glover, F. (1990). Tabu Search – Part II, *ORSA Journal on Computing*, Vio. 2, No. 1 (Winter 1990), pp. 4–32, ISSN: 0899-1499.

Hei, Y. Q.; Li, X. H.; Yi, K. C. & Yang, H. (2009). Novel scheduling strategy for downlink multiuser MIMO system: Particle swarm optimization. *Science in China - Series F: information Sciences,* Vol. 52, No. 12 (2009), pp. 2279-2289, ISSN (printed): 1009-2757.

Holland, J. H. (1987). Genetic algorithms and classifier systems: foundations and future directions. *Proceedings of 2nd international conference on genetic algorithms and their application*, pp. 82-89, ISBN:0-8058-0158-8, Cambridge, Massachusetts, United States, 1987, L. Erlbaum Associates Inc., Hillsdale, NJ, USA.

Kashan, A. H. & Karimi, B. (2009). A discrete particle swarm optimization algorithm for scheduling parallel machines. *Computers & Industrial Engineering*, Vol. 56, No. 1 (February 2009), pp. 216-223, ISSN: 0360-8352.

Kashan, A. H.; Karimi, B. & Jenabi, M. (2008). A hybrid genetic heuristic for scheduling parallel batch processing machines with arbitrary job sizes. *Computers and Operations Research,* Vol. 35, No. 4 (April 2008), pp. 1084-1098, ISSN: 0305-0548.

Kennedy, J. & Eberhart, R. C. (1995). Particle swarm optimization. *Proceedings of IEEE Int'l. Conf. on Neural Networks*, pp. 1942-1948, ISBN: 0-7803-2768-3, Perth, WA , Australia, Nov/Dec 1995.

Kennedy, J., Eberhard, R.C. (1997). A discrete binary version of the particle swarm algorithm. *Proceedings of IEEE Conference on Systems, Man, and Cybernetics*, pp. 4104--4109, ISBN: 0-7803-4053-1, Orlando, FL , USA, Oct. 1997, Piscataway, NJ.

Kirkpatrick, S.; Gelatt, C. D. & Vecchi, M. P. (1983). Optimization by Simulated Annealing. *Science*, Vol. 220, No. 4598 (May 1983), pp. 671 – 680, ISSN: 0036-8075.

Kuo, I. H.; Horng, S. J.; Kao, T. W.; Lin, T. L.; Lee, C. L.; Terano, T. & Pan, Y. (2009). An efficient flow-shop scheduling algorithm based on a hybrid particle swarm optimization model. Expert systems with applications, Vol. 36, No. 3 (April 2009), pp. 7027-7032, ISSN: 0957-4174.

Laskari, E. C.; Parsopoulos, K. E. & Vrahatis, M. N. (2002). Particle swarm optimization for integer programming. *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 1582-1587, ISBN: 0-7803-7282-4, Honolulu, HI , USA, May 2002.

Lee, L. T.; Tao, D. F. & Tsao, C. (2007). An adaptive scheme for predicting the usage of grid resources. *Computers & Electrical Engineering,* Vol. 33, No. 1 (Jan. 2007), pp. 1-11, ISSN:0045-7906.

Lee, W. C.; Wu, C. C. & Chen, P. (2006). A simulated annealing approach to makespan minimization on identical parallel machines. *International Journal of Advanced Manufacturing Technology,* Vol. 31, No. 3-4 (2006), pp. 328-334, ISSN (printed): 0268-3768.

Liao, C. J.; Tseng, C. T. & Luarn, P. (2007). A discrete version of particle swarm optimization for flowshop scheduling problems. *Computers and Operations Research*, Vol. 34, No. 10 (October 2007), pp. 3099-3111, ISSN: 0305-0548.

Mathaisel, D. & Comm, C. (1991). Course and classroom scheduling: an interactive computer graphics approach. *Journal of Systems and Software*, Vol. 15 (May 1991), pp. 149-157, ISSN: 0164-1212.

Metropolis, N.; Rosenbluth, A. W.; Rosenbluth, M. N.; Teller, A. H. & Teller, E. (1953). Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics*, Vol. 21, No. 6 (June 1953), pp. 1087-1092, ISSN (printed): 0021-9606.

Norman, M. G. & Thanisch, P. (1993). Models of machines and computation for mapping in multicomputers. *ACM Computing Surveys*, Vol. 25, No. 3 (September 1993), pp. 263-302, ISSN: 0360-0300.

Ohki, M.; Morimoto, A. & Miyake, K. (2006). Nurse Scheduling by Using Cooperative GA with Efficient Mutation and Mountain-Climbing Operators. *Proceedings of 2006 3rd International IEEE Conference on Intelligent System*, pp. 164–169, London, Sept. 2006.

Salman, A.; Ahmad, I. & Al-Madani, S. (2002). Particle swarm optimization for task assignment problem. *Microprocessors and Microsystems,* Vol. 26, No. 8 (November 2002), pp. 363-371, ISSN: 0141-9331.

Sha, D. Y. & Hsu, C. Y. (2006). A hybrid particle swarm optimization for job shop scheduling problem. *Computers and Industrial Engineering*, Vol. 51, No. 4 (December 2006), pp. 791-808, ISSN:0360-8352.

Simeonov, S. & Simeonovova, J. (2002). Simulation Scheduling in Food Industry Application, Mathematical and statistical methods. *Food processing and preservation,* Vol. 20, No. 1 (Jan. 2002), pp. 31-37, ISSN: 1212-1800.

Watts, D. J. & Strogatz, S. H. (1998). Collective dynamics of 'small-world' networks. Nature, Vol. 393, No. 6684 (June 1998), pp. 440–442, ISSN (printed): 0028-0836.

Zhai, X.; Tiong, R.L.K.; Bjornsson, H.C. & Chua, D.K.H. (2006). A Simulation-Ga Based Model for Production Planning in Precast Plant, *Proceedings of Winter Simulation Conference*, pp. 1796 – 1803, ISBN: 1-4244-0500-9, Monterey, CA, Dec. 2006.