J. Cent. South Univ. (2016) 23: 421−429
DOI: 10.1007/s11771-016-3087-z

Springer

A hybrid discrete particle swarm optimization-genetic algorithm for multi-task scheduling problem in service oriented manufacturing systems

WU Shan-yu(武善玉)[1], ZHANG Ping(张平)[1], LI Fang(李方)[1], GU Feng(古锋)[2], PAN Yi(潘毅)[3]

1. School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China;
2. Department of Computer Science, College of Staten Island, Staten Island, New York 10314, USA;
3. Department of Computer Science, Georgia State University, Atlanta, GA 30303, USA

Abstract: To cope with the task scheduling problem under multi-task and transportation consideration in large-scale service oriented manufacturing systems (SOMS), a service allocation optimization mathematical model was established, and then a hybrid discrete particle swarm optimization-genetic algorithm (HDPSOGA) was proposed. In SOMS, each resource involved in the whole life cycle of a product, whether it is provided by a piece of software or a hardware device, is encapsulated into a service. So, the transportation during production of a task should be taken into account because the hard-services selected are possibly provided by various providers in different areas. In the service allocation optimization mathematical model, multi-task and transportation were considered simultaneously. In the proposed HDPSOGA algorithm, integer coding method was applied to establish the mapping between the particle location matrix and the service allocation scheme. The position updating process was performed according to the cognition part, the social part, and the previous velocity and position while introducing the crossover and mutation idea of genetic algorithm to fit the discrete space. Finally, related simulation experiments were carried out to compare with other two previous algorithms. The results indicate the effectiveness and efficiency of the proposed hybrid algorithm.

Key words: service-oriented architecture (SOA); cyber physical systems (CPS); multi-task scheduling; service allocation; multi-objective optimization; particle swarm algorithm

1 Introduction

Manufacturing always attracts wide attention from industry, academia and the government due to its significant impact on economy, society and environment. With the development of such novel technologies and concepts as embedded systems, service-oriented architecture (SOA), Internet of Things (IoT), and cloud computing, they are applied in manufacturing and are continuously progressing towards deeply networked, service-oriented, and globalized systems. These manufacturing systems are called service-oriented manufacturing systems (SOMS), which are large-scale, completely resource sharing, and on-demand serving. For example, web services are implemented on devices as well as in software so that all devices could be seamlessly integrated into the whole manufacturing system with all high level information subsystems [1–2]. In addition, with the advance of cyber-physical systems (CPS) [3–4] in very recent years, it is possible to deeply integrate cyber systems with physical world by connecting the machines to distributed software applications, even people and networks of sensors, controls and processors into a whole system such as industrial internet [5]. Moreover, cloud manufacturing is proposed based on the concept of cloud computing and the idea of “Manufacturing as a Service (MaaS)”, in which multiple resources serve multiple tasks simultaneously [6–7].

In these SOMSs, everything is treated as a service. Each involved resource in the whole life cycle of a product, a piece of software or a hardware device, is encapsulated into a service. In a SOMS, there are soft-services and hard-services based on software and hardware devices, respectively. For a certain task, right services are chosen to form a composite service which can best perform the task without realizing the providers of these selected services. Therefore, boundaries between different factories or areas are blurred. However, transport links may be necessary during production of a task because the selected hard-services are possibly provided by various providers in different areas. Furthermore, since the same types of multiple tasks are

Foundation item: Project(2012B091100444) supported by the Production, Education and Research Cooperative Program of Guangdong Province and Ministry of Education, China; Project(2013ZM0091) supported by Fundamental Research Funds for the Central Universities of China
Received date: 2014−12−09; **Accepted date:** 2015−05−05
Corresponding author: ZHANG Ping, Professor, PhD; Tel: +86–20–87114170; E-mail: pzhang@scut.edu.cn

often performed simultaneously, it is a very important issue related to the system performance to schedule them or assign services in the same set. In recent years, many scholars have studied the task scheduling problems in various modern networked manufacturing systems. However, to the best of our knowledge, few works consider multiple tasks and transport links at the same time. To solve the multi-task scheduling problem considering transport links (MTSTL) in a SOMS, a hybrid discrete particle swarm optimization-genetic algorithm (HDPSOGA) is proposed in this work. Under the framework of the particle swarm optimization, the crossover and mutation operation of genetic algorithm are applied to each newly generated particle.

The main contributions of this work lie in the following three folds. Firstly, we analyzed the characteristics of MTSTL and developed its model. Secondly, we solved the model by proposing the HDPSOGA algorithm to incorporate the transport link between two services and multiple tasks simultaneously. Finally, we implemented the proposed algorithm and proved its efficiency and performance by the designed experiments.

2 Related work

In distributed networked manufacturing systems, the resource reconfiguration and optimal scheduling are characterized by single-task and partially collaborative machining, in which swarm intelligence algorithms, especially the genetic algorithm (GA) [8–9] and the particle swarm optimization (PSO) [10–11] algorithm are the most commonly applied and achieved satisfactory results. CAR et al [12] proposed a GA-based method to optimize the processes of surface grinding, mental cutting, electric discharge machining, and CNC turning. WEI et al [13] used an improved genetic algorithm based on Pareto multi-objective immune to solve the reconfiguration and optimal scheduling problem for networked collaborative manufacturing, in which the population ranking technique, the niche technique, and the Pareto solution set filter technology were adopted to ensure the groups variety and prevent the premature convergence problem. MA et al [14] presented the collaborative machining of parts, or some process sections of some parts, or processes of sections under circumstance of networked manufacturing, and then solved the optimization deployment of resources for single-task using binary-coding genetic algorithm. However, these solutions suffer from the limitations of the GA including low convergence efficacy, difficult finding of the appropriate parameters' values in the algorithms.

Although PSO was firstly introduced in social behavior of a bird flock by KENNEDY and EBERHART [10–11], it has been widely adopted in manufacturing area. NAVALERTPORN and AFZULPURKAR [15] did a detailed survey about PSO and its variants and proposed an integrated approach based on an artificial neural network (ANN) and bidirectional particle swarm optimization (BPSO) to solve multi-objective process parameter optimization problems. Moreover, they showed the performance using a case study involving the optimization of process parameters for cement roof-tile pressing. TAO et al [16–20] carried out a lot of research related to the resource service match, search, optimal-selection and composition problems in manufacturing grid systems. They presented a new method based on the PSO to solve the resource service composition and optimal-selection problem of manufacturing grid system [16] and proposed an algorithm which combined PSO with the non-dominated sorting technique, dynamically generating parameters in updating formulation and permutation-based and objective-based population trimming operators to solve the multi-objective resource service composition and optimal-selection problem [20].

As distributed manufacturing systems evolved into cloud manufacturing, characterized by full-scale service and computing oriented, new requirements of service composition appeared. The two most noticeable problems are transport links between different links and the multiple tasks. TAO et al [21] formulated the QoS model of the composition service for single-task in cloud manufacturing systems which considered both hardware resources and software resources as services, and then solved the model using a novel "full connection based parallel chaos optimization with reflex migration" algorithm. LIU et al [22] incorporated the assumptions and principles of multi-task multi-coalition generation problem into the multi-task oriented manufacturing cloud service composition (MO-MCSC) in cloud manufacturing systems, then took account for the possibility of resource-constrained, formulated the model using the methodology of single-task oriented manufacturing service composition and solved it by a proposed matrix real-code based genetic algorithm. From the above work, we notice that they only considered either the transport links or the multiple tasks. In our work, we will examine both of the problems in the SOMS.

3 Problem description

The service allocation problem is described as follows. Assume there are N productive tasks with the same type in a set of tasks T:

$$T=\{T_1,\ T_2,\ \cdots,\ T_i,\ \cdots,\ T_N\} \tag{1}$$

where T_i means the i-th task, which includes a sequence of n subtasks:

$$T_i=\{T_{i1},\ T_{i2},\ \cdots,\ T_{ij},\ \cdots,\ T_{in}\} \tag{2}$$

where T_{ij} refers to the j-th subtask of the task T_i.

T_j^* is used to include all the j-th subtasks for tasks T_1, T_2, $\cdots$, T_i, $\cdots$, T_N:

$$T_j^*=\{T_{1j},\ T_{2j},\ \cdots,\ T_{Nj}\} \tag{3}$$

The system selects some available services for T_j^* according to a certain service discovery and matching scheme and forms a set R_j:

$$R_j=\{S_{j,k}\mid k\in[1,m_j]\} \tag{4}$$

where $S_{j,k}$ is the k-th service and m_j is the number of services in the set R_j. Each subtask in T_j^* is considered to process on any service in R_j. The processing performance of the subtask T_{ij} on the service $S_{j,k}$ is denoted as p_{ijk} and the transportation performances between any two services of the adjacent subtasks are fixed. Therefore, the solution space Z is denoted:

$$Z=\prod_{j=1}^{n}[m_j!/(m_j-N)!] \tag{5}$$

The associated assumptions are described as follows: 1) All tasks have the same type and should be processed by the same manufacturing procedure; 2) The available services are either local or remote; 3) The relationship between any two adjacent subtasks of the same task is strictly sequential; 4) One service can process at most one subtask at a time; 5) Once being processed, one subtask cannot be interrupted.

Based on the above assumptions, the completion time t_i of the task T_i is composed of the processing time of all subtasks belonging to T_i, and the transportation time of materials or products between any two adjacent subtasks or between the user and production uses. The total cost c_i of the task T_i includes the processing cost and the transportation cost. Likewise, the overall quality q_i of the task T_i consists of the processing quality and the transportation quality.

On the basis of above study, the optimization algorithm is designed to assign all subtasks to the corresponding available services properly and aims to achieve the overall optimal schedule with several objectives: 1) T: the total completion time of all tasks; 2) C: the total completion cost of all tasks; 3) Q: the maximal reject ratio of all tasks.

The fitness value $F(X)$ of MTSTL is defined as

$$F(X)=\alpha T+\beta C+\gamma(f_{\mathrm{u}}\times Q) \tag{6}$$

where α, β and γ are scaling factors which indicate the significance of each objective and $\alpha+\beta+\gamma=1$, and f_{u} is the factor which normalizes the reject ratio Q to the close proximate magnitude as C and T.

4 HDPSOGA for MTSTL

To solve the model formalized in Section 3, a hybrid discrete particle swarm optimization-genetic algorithm (HDPSOGA) is proposed, which integrates some ideas of the two classical swarm intelligent algorithms PSO and GA. The proposed algorithm aims to improve the population initializing method and the updating strategy.

4.1 Encoding scheme

The position of the h-th particle $\boldsymbol{X}_h$ in the swarm representing a solution of a service composition is encoded into a $N\times n$ matrix called service sequence matrix:

$$\boldsymbol{X}_h=\left\{\begin{matrix} x_{h11} & x_{h12} & \cdots & x_{h1j} & \cdots & x_{h1n}\\ x_{h21} & x_{h22} & \cdots & x_{h2j} & \cdots & x_{h2n}\\ \vdots & \vdots & \vdots & \vdots & \cdots & \vdots\\ x_{hi1} & x_{hi2} & \cdots & x_{hij} & \cdots & x_{hin}\\ \vdots & \vdots & \vdots & \vdots & \cdots & \vdots\\ x_{hN1} & x_{hN2} & \cdots & x_{hNj} & \cdots & x_{hNn} \end{matrix}\right\} \tag{7}$$

where the i-th row of $\boldsymbol{X}_h$ corresponds to the service composition for the task T_i, and the j-th column is linked to the subtask set T_j^*. The element $x_{hij}\in[1,\ m_j]$ denotes the number of the service assigned to the subtask T_{ij}. For example, the second row $\boldsymbol{X}_{h2}$=(1,2,1,3,2) indicates that the subtasks T_{21}, T_{22}, T_{23}, T_{24}, T_{25} of the task T_2 will be processed by the services $S_{1,1}$, $S_{2,2}$, $S_{3,1}$, $S_{4,3}$ and $S_{5,2}$, respectively.

4.2 Decoding a particle to a schedule

Let t_{ij} denote the completion time of the subtask T_{ij}. We use $t_{i(j-1,j)}$ to represent the transportation time, $c_{i(j-1,j)}$ to stand for the transportation cost, and $q_{i(j-1,j)}$ to refer to the transportation quality of the material from the processing position of subtask $T_{i(j-1)}$ to the current subtask T_{ij}, respectively. We define $t^{\mathrm{p}}_{ij,k}$, $c^{\mathrm{p}}_{ij,k}$, and $q^{\mathrm{p}}_{ij,k}$ as the processing time, the cost, and the product qualification rate of T_{ij} on service $S_{j,k}$. In the same manner, we denote the completion time, the total cost, and the product qualification rate of the task T_i by t_i, c_i, and q_i, respectively.

The decoding procedure of a particle is presented as follows. Firstly, we parse the service sequence matrix $\boldsymbol{X}_h$ row by row from top to bottom. For the i-th row $\boldsymbol{X}_{hi}$, we pick the j-th element x_{hij} from left to right one after another, and determine all necessary properties relevant to the current subtask by repeating the steps below.

1) Identify the completion time $t_{i(j-1)}$. If j=0, meaning that the current subtask is the first one of the task T_i, set $t_{i(j-1)}$ to be 0.

2) Identify the transportation time $t_{i(j-1,j)}$, the cost $c_{i(j-1,j)}$, and the qualification rate $q_{i(j-1,j)}$ from the processing position of previous subtask to the current site of $S_{j,x_{hij}}$.

3) Identify the service $S_{j,x_{hij}}$ allocated to the subtask T_{ij} in set R_j, and its processing time $t^{\mathrm{p}}_{ij,x_{hij}}$, cost $c^{\mathrm{p}}_{ij,x_{hij}}$, and product qualification rate $q^{\mathrm{p}}_{ij,x_{hij}}$.

4) Calculate t_i, c_i, q_i and then compute T, C, and Q using following equations:

$$T=\sum_{i=1}^{N}t_i=\sum_{i=1}^{N}(\sum_{j=1}^{n}(t_{i(j-1,j)}+t^{\mathrm{p}}_{ij,x_{hij}})+t_{i(n,0)}) \tag{8}$$

$$C=\sum_{i=1}^{N}c_i=\sum_{i=1}^{N}(\sum_{j=1}^{n}(c_{i(j-1,j)}+c^{\mathrm{p}}_{ij,x_{hij}})+c_{i(n,0)}) \tag{9}$$

$$Q=1-\prod_{i=1}^{N}q_i=1-\prod_{i=1}^{N}(\prod_{j=1}^{n}q^{\mathrm{p}}_{ij,x_{hij}}q_{i(j-1,j)}q_{i(n,0)}) \tag{10}$$

4.3 Initialization of population

An initializing method is used to obtain more optimal initial population so that the proposed algorithm can achieve the better performance. In the proposed method, the position of a particle is initialized column by column regardless of the transportation relationship between them. For a particular column, its initialization is similar to that of the full flexible job-shop scheduling problem [23–24]. We obtain the inspiration from the assignment rule of the approach proposed by PEZZELLA et al [23] since they used the less complex algorithms to get quality initial population based on the approach by location [24]. It uses a mix of two rules for the machine resource assignment by searching the minimum processing time of all jobs on any machine and fixing that assignment. To avoid the effects of the jobs/machines order, it randomly permutes jobs and machines originally ordered.

In proposed algorithm, the processing times, costs, and product qualification rates for subtasks are considered. For each column of a particle, we record the processing time, the cost, and the product qualification rate of every subtask on each service within the corresponding service set. Then, we initialize a column according to one of the following three rules (the proportions of initial particles generated by Rule 1, Rule 2 and Rule 3 are set to be α, β and γ, respectively).

Rule 1: Initialize a column by repeating the following steps until all subtasks in the table obtain their service resources:

Step 1: Randomly select a subtask in above mentioned column and find its minimum processing time and determine that assignment (set the corresponding element in the service sequence matrix according to the order of the corresponding service) if the corresponding service is not marked as “unavailable”.

Step 2: Mark the allocated service as “unavailable”.

Rule 2 and Rule 3 use the same process for the processing cost and the product qualification rate. Note that Rule 3 tries to find the maximum product qualification rate instead of the minimal processing time or cost in Rule 1 and Rule 2.

After initializing the position of a particle, we set the initial individual best position $\boldsymbol{b}_k$ immediately. When the whole initial population is generated, we calculate the fitness of each particle to select the optimal individual as the global best position $\boldsymbol{G}$.

4.4 Update of particles

In this work, the updating method for basic PSO cannot be applied because the particle approach is a discrete solution. In previous studies, some discrete PSOs or hybrid discrete PSOs for discrete problems, such as job-shop scheduling problem (JSSP) [25] and flexible job-shop scheduling problem (FJSP) [26–27], have been proposed. In this work, we present a hybrid scheme that is suitable for the multitask services composition discrete problem described in Section 3. In the proposed scheme, each particle is updated according to four aspects as follows.

1) Update the new position of the particle h in the k-th iteration column by column depending on its previous position by

$$\boldsymbol{X}_h^{k(1)}=w\otimes\mathrm{reverse}(\boldsymbol{X}_h^k)=\begin{cases}\mathrm{reverse}(\boldsymbol{X}_h^k),\ \text{if rand()}>w\\ \boldsymbol{X}_h^k,\ \text{otherwise}\end{cases} \tag{11}$$

where w is the inertia weight, “reverse” is the function to reverse the order of elements between two randomly selected points of the current column of $\boldsymbol{X}_h^k$, and $\boldsymbol{X}_h^{k(1)}$ is an interim state between $\boldsymbol{X}_h^k$ and $\boldsymbol{X}_h^{k+1}$.

Update the position generated by the last step column by column from left to right depending on the local best position of the h-th particle. We formalize the changing as

$$\boldsymbol{X}_h^{k(2)}=d_h^b\otimes\mathrm{col\_cross}(\boldsymbol{X}_h^{k(1)},\ \boldsymbol{b}_h^k) \tag{12}$$

where the crossover probability d_h^b is proportional to the Hamming distance between $\boldsymbol{X}_h^{k(1)}$ and $\boldsymbol{b}_h^k$. The Hamming distance H is calculated as the number of positions at which the corresponding values are different for two particles. Then, the crossover probability d_h^b is obtained from

$$d_h^b=H_h^b/(N\times n) \tag{13}$$

The function "col_cross" is implemented by replacing parts of $\boldsymbol{X}_h^{k(1)}$ with the corresponding parts of $\boldsymbol{b}_h^k$ with probability d_h^b. For each column of $\boldsymbol{X}_h^{k(1)}$, we generate a random number between 0 and 1, rand(), and then randomly select a segment from the current column. If rand()< d_h^b, we replace it with the corresponding segment in the local best position $\boldsymbol{b}_h^k$, and then proceed to the next column and repeat the same process until all columns are finished.

Once all the columns have been updated, we adjust the elements of $\boldsymbol{X}_h^{k(2)}$ to correct the repeated ones in the same column using the method of elements correction which will be provided in Section 4.5.

3) Update the interim position $\boldsymbol{X}_h^{k(2)}$ depending on the overall best position using

$$\boldsymbol{X}_h^{k(3)} = d_h^G \otimes \text{col\_cross}(\boldsymbol{X}_h^{k(2)},\ \boldsymbol{G}_h^k) \tag{14}$$

$$d_h^G = H_h^G \big/ (N \times n) \tag{15}$$

Then we correct elements which have been set to same values in each same column using the method of elements correction.

4) In order to avoid the premature convergence, a disturbance mechanism is applied in the proposed algorithm to mutate the particles which incline to stagnation state. We judge a stagnant particle by the limited number of consecutive generations denoted as $g_{\max}$, during which the local best position has not been refreshed. In this work, the interim position $\boldsymbol{X}_h^{k(3)}$ is mutated into $\boldsymbol{X}_h^{k+1}$ using

$$\boldsymbol{X}_h^{k+1} = (g_h / g_{\max}) \otimes \text{mutate}(\boldsymbol{X}_h^{k(3)}) \tag{16}$$

where g_h stands for the number of consecutive generations that the local best position of the h-th particle has not been refreshed, $(g_h/g_{\max})$ is the mutation probability, and "mutate" is the function implemented by replacing randomly selected elements of the third interim particle with their opposition values, i.e., x_{hij} is replaced by $(m_j - x_{hij})$. Finally, we correct the elements which have the same values in the same column.

4.5 Elements correction

Updating a particle using the function "col_cross" or "mutate" could cause the same values to be found in the same column, which means that more than one subtasks have been assigned to the same service. So, it is necessary to correct the repeated elements in the same column right after updating a particle. Each particle is adjusted column by column and concrete steps of the correction operation are as follows.

1) Divide the service set related to the current column into two sets $\boldsymbol{U}_1$ and $\boldsymbol{U}_2$. $\boldsymbol{U}_1$ consists of all services found in the current column, and $\boldsymbol{U}_2$ is composed of other services.

2) Read elements of the current column one by one from top to bottom, and collect the row numbers of elements with equal values into a collection $\boldsymbol{S}_a$. All such collections are denoted as $\boldsymbol{S}_1, \boldsymbol{S}_2, \cdots, \boldsymbol{S}_v\ (v \le N)$.

3) Keep all collections $\boldsymbol{S}_1, \boldsymbol{S}_2, \cdots, \boldsymbol{S}_v$ in random order and process them one by one.

4) For any collection, select the element whose processing performance on the assigned service is the best and delete it from this collection. Then, randomly assign services in $\boldsymbol{U}_2$ to all elements in the current collection, modify their values of corresponding elements in the current column of the particle, and delete the assigned services from $\boldsymbol{U}_2$.

5) Repeat the above Step 1) to 4) until all collections $\boldsymbol{S}_1, \boldsymbol{S}_2, \cdots, \boldsymbol{S}_v$ are processed.

4.6 Procedure of HDPSOGA

The procedure of the proposed HDPSOGA is illustrated as follows:

1) Define the population size "SwarmSize", the maximum iterations "MaxT" and all related parameters;

2) Generate initial population $\boldsymbol{P}(0)$ according to the approach described in Section 4.3;

3) Decode each particle and calculate the fitness using Eq. (6);

4) Refresh the local optimal position $\boldsymbol{b}$ and the global optima position $\boldsymbol{G}$;

5) Repeat the following steps for "MaxT" times:

(1) Update $\boldsymbol{P}(g-1)$ to yield $\boldsymbol{P}^{(1)}(g)$ according to Eq. (11);

(2) Update $\boldsymbol{P}^{(1)}(g)$ to yield $\boldsymbol{P}^{(2)}(g)$ according to Eq. (12);

(3) Adjust elements of $\boldsymbol{P}^{(2)}(g)$ using elements correction approach;

(4) Update $\boldsymbol{P}^{(2)}(g)$ to yield $\boldsymbol{P}^{(3)}(g)$ according to Eq. (14);

(5) Adjust elements of $\boldsymbol{P}^{(3)}(g)$ using the elements correction approach;

(6) Update $\boldsymbol{P}^{(3)}(g)$ to yield $\boldsymbol{P}(g)$ according to Eq. (16);

(7) Adjust elements of $\boldsymbol{P}(g)$ using the elements correction approach;

(8) Decode each particle and calculate the fitness using Eq. (6);

(9) Refresh the local optimal position $\boldsymbol{b}$ and the global optimal position $\boldsymbol{G}$;

6) Output $\boldsymbol{G}$ and its corresponding solution as the near optimal schedule.

From the above algorithm, we know that although the HDPSOGA algorithm adopts the framework of PSO with the ideas of the crossover and mutation, it extends PSO by integrating updating and initialization methods.

5 Simulation results and analysis

The HDPSOGA has been implemented in Matlab2009a on a personal computer with 2.00 GHz Intel Pentium Dual CPU, 2 GB RAM and tested on two problem instances with 4 tasks, each of which has 10 subtasks. For each stage of the subtasks, there is a service set containing some corresponding services. Two cases have been considered. Every service set contains 10 and 20 candidate services in Case 1 and Case 2, respectively.

5.1 Performance comparison with PSO and GA

In this experiment, our algorithm is compared to two classical optimization algorithms: the standard particle swarm optimization (PSO) and the genetic algorithm (GA). The encoding approaches used in these three algorithms are the same as those described in Section 4.1. Some parameters and weight coefficients of all algorithms are set to the most commonly used values in PSOs or GAs, and others are randomly set. All the processing and transportation times, costs and qualities are randomly generated within a pre-determined range. More specifically, the processing and transportation times are randomly generated between 1 and 120, the costs are random integers between 50 and 1000, and the qualities are originally expressed by qualification rates between 90% and 100%. The values of parameters applied in our experiment are: population size: 200; number of generations: 500; rate of initial population with Rule 1: 1/3; rate of initial population with Rule 2: 1/3; rate of initial population with Rule 3: 1/3; inertia weight w of HDPSOGA and PSO: 0.7298; limited number of consecutive generations $g_{\max}$: 10; scaling factors α, β, γ: 1/3, 1/3, 1/3; learning factor c_1, c_2 of PSO: 2, 2; mutation probability p_m of GA: 0.03; crossover probability p_c of GA: 0.8.

Figure 1 shows the fitness and objective values of three algorithms in 20 runs for Case 1. From Fig. 1(a), we can see that all the optimal fitness values of GA are smaller than those of PSO and greater than those of HDPSOGA. More specifically, the HDPSOGA has the uniform optimal fitness value for all the 20 runs. This is because the calculated optimal fitness values are the same for each run. From Fig. 1(b), we know that GA has slightly greater total completion time than HDPSOGA, whose total completion time is the same for all the runs (952). However, the total completion time of PSO has a wide range from 707 to 1246 for the 20 different runs. This may be resulted from the fitness value used in this work as well as the features of PSO. The fitness value $F(X)$ is the weighting sum of three sub-objectives and each sub-objective has its contribution. Although two or

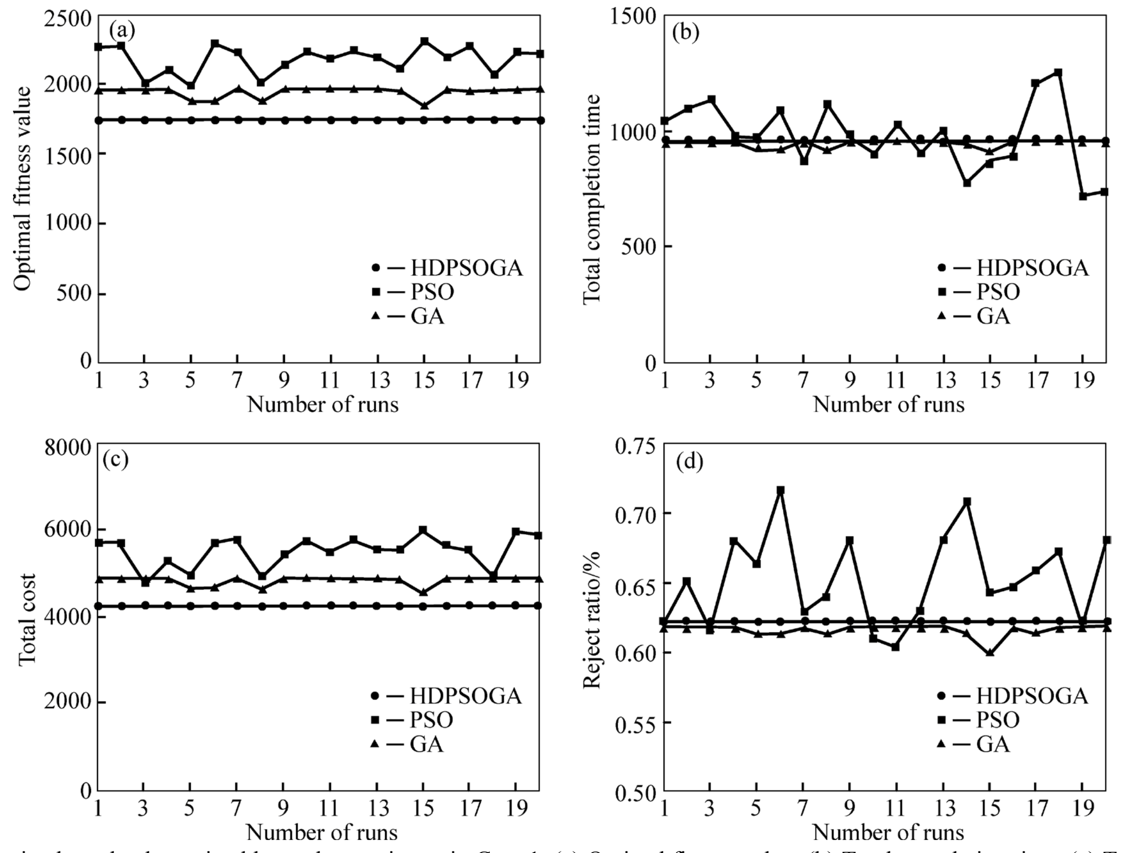

Fig. 1 Optimal results determined by each experiment in Case 1: (a) Optimal fitness value; (b) Total completion time; (c) Total cost; (d) Reject ratio

more fitness values are similar, their sub-objectives have large differences. In Fig. 1(c), HDPSOGA has the smallest total cost compared to PSO and GA. Compared to GA, PSO has more total costs for most of the runs and exhibits the instable manner. From Fig. 1(d), GA maintains the lowest reject ratio although HDPSOGA has very small differences from GA. For most of runs, PSO has a relatively higher reject ratio than HDPSOGA and GA. From Fig. 1, we notice that HDPSOGA has uniform values for all the metrics including the optimal fitness value, the total completion time, the total cost, and the reject ratio for all 20 runs. However, all the values of PSO display the instable pattern and those of GA slightly fluctuate. Therefore, the stability of HDPSOGA is the best and that of GA is the worst.

Figure 2 displays the fitness and sub-objective values of three algorithms in 20 runs for Case 2. From Fig. 2(a), we know that HDPSOGA has the smaller optimal fitness values than PSO and GA, and PSO and GA have similar optimal fitness values for all the 20 runs. Therefore, HDPSOGA has better results than PSO and GA for larger scale systems like Case 2, and the difference between PSO and GA is smaller with the increase of the problem's scale. From Fig. 2(b), we can see that there are not remarkable differences in the total completion time among HDPSOGA, PSO, and GA for all the 20 runs. In Fig. 2(c), the HDPSOGA costs much less than PSO and GA. In Fig. 2(d), it is hard to see the difference in the reject ratio for all the 20 runs for all the three algorithms. The possible reasons are the used fitness value in this work and the magnitude of the total cost in the proposed algorithm. From Fig. 2, we conclude that the metrics including the optimal fitness value, the total completion time, the total cost, and the reject ratio have slight oscillation in all the runs for the second case. It is maybe because of the larger number of services. Although the difference between HDPSOGA and other two algorithms becomes less obvious and PSO and GA have the stability with little difference, HDPSOGA still has the best stability for the larger scale of Case 2.

Table 1 shows the average fitness values of all the 20 runs and the improvement achieved by HDPSOGA for both of Case 1 and Case 2. We know that the average optimal fitness values of HDPSOGA for Case 1 are reduced by 19.89% and 10.08%, respectively, compared to PSO and GA. The corresponding reduced rates are 25.81% and 25.78, respectively, for Case 2. Therefore, more improvements of HDPSOGA compared to both of PSO and GA are made for larger scale cases.

5.2 Efficiency of HDPSOGA

In order to validate the efficiency of HDPSOGA, two experiments were performed. In the first one, we set the population size to be 200, 300, 400, 500 and 700,

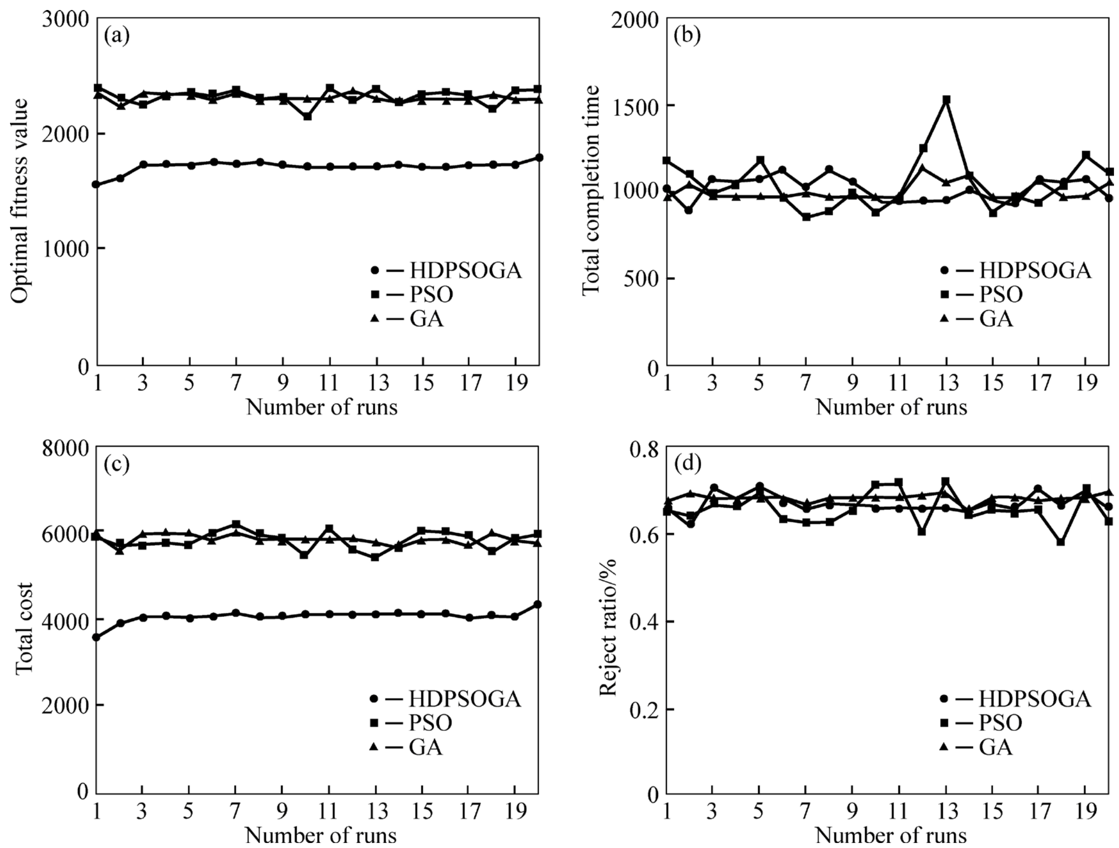

Fig. 2 Results determined by each experiment in Case 2: (a) Optimal fitness value; (b) Total completion time; (c) Total cost; (d) reject ratio

Table 1 Average optimal fitness values for two cases and improvement by HDPSOGA

Case	Parameter	HDPSOGA	PSO	GA
1	Optimal fitness value	1746.7	2180.3	1942.5
	Improvement/%	—	19.89	10.08
2	Optimal fitness value	1711.3	2306.8	2305.4
	Improvement/%	—	25.81	25.78

respectively, and other parameters are as the same as Section 5.1. We perform the algorithm 20 times for every population size. Figure 3 displays the average results and average time consumptions for 20 runs for different population sizes. From Fig. 3(a), we can see that the used time is almost proportional to the population size for both cases. More specifically, Case 2 consumes more time since it contains more service candidates for the subtasks. In Fig. 3(b), the average fitness values are the same for Case 1 despite of the population size. However, after the population size reaches 400, the average fitness values become stable with the increase of the population size. This is because the larger scale systems need bigger population sizes to maintain the stability of the algorithm. Because the used time is approximately proportional to the population size, the algorithm can effectively solve both of the cases with different scales.

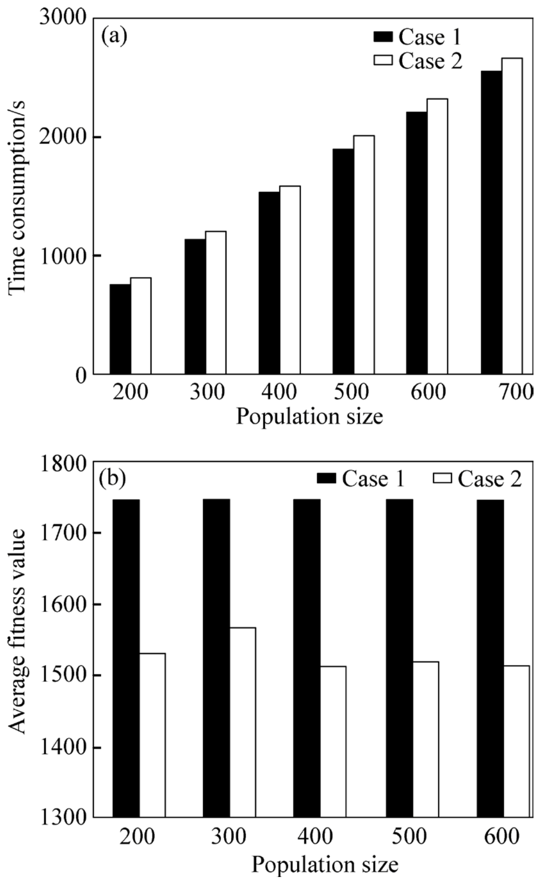

Fig. 3 Effects of population size: (a) Average time consumptions; (b) Average fitness values

To test the parameter of the needed maximum number of iterations, we conduct the same experiment in Section 5.1 by applying different number of iterations from 500 to 1000. The average consumed time and fitness values are plotted in Fig. 4. From Fig. 4(a), we know that the consumed time increases linearly with the increase of the number of iterations for both of cases. From Fig. 4(b), we conclude that after the maximum number of iterations exceeds 500, it has little impact on the average fitness values for Case 1 and Case 2.

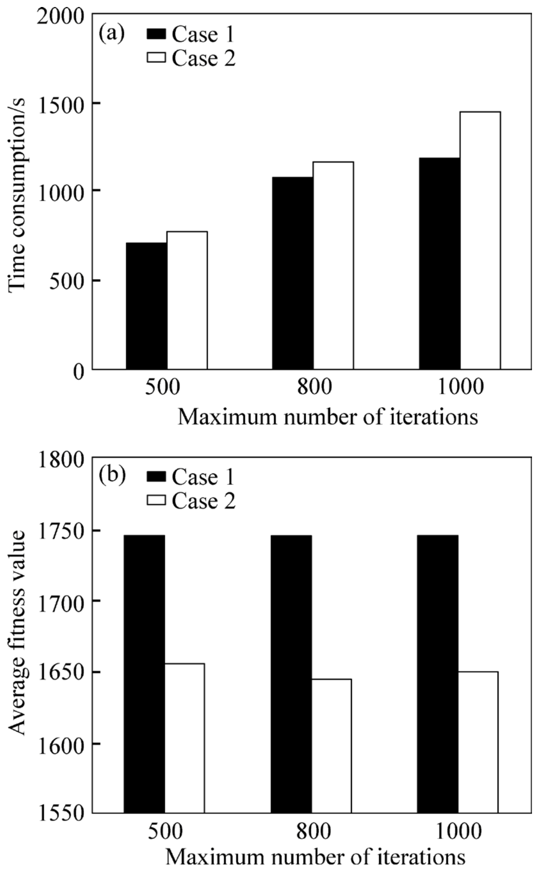

Fig. 4 Effect of maximum number of iterations: (a) Average time consumptions; (b) Average fitness values

Overall, HDPSOGA has stronger searching ability and stability than PSO and GA. It obtains more optimal results than traditional swarm intelligent algorithms. Moreover, HDPSOGA can effectively solve the problems with different scales.

6 Conclusions

We present a hybrid discrete particle swarm optimization-genetic algorithm for the multi-task scheduling problem of large-scale service oriented manufacturing systems. Because the services in this kind of systems include hard-services as well as soft-services, the transport link between different subtasks is an important factor affecting the performance of the entire system. The most distinctive feature of the problem is that we consider the transport links between different subtasks and the multiple tasks with the same type simultaneously. The solution spaces of the multi-task scheduling problem are much larger than those of the single-task scheduling since it includes the transport links. To improve searching capability of the algorithm, we combine the ideas of both PSO and GA, and adjust the updating method to adapt the discrete problem. In addition, some initialization strategies are introduced to provide more optimal initial populations. Compared to traditional swarm intelligent algorithms, the searching capability is remarkable in the experiments tested on two cases with different scales.

References

[1] KARNOUSKOS S, BAECKER O, SOUZA L M S D, SPIESS P. Integration of SOA-ready networked embedded devices in enterprise systems via a cross-layered web service infrastructure [C]// 12th IEEE Conference on Emerging Technologies and Factory Automation. Patras, Greece: IEEE Press, 2007: 25–28.

[2] TAISCH M, COLOMBO A W, KARNOUSKOS S, CANNATA A. SOCRADES road map [EB/OL]. [2014–08–18]. http://www.socrades.eu/Documents, 2010/.

[3] LEE E A. Cyber physical systems: Design challenges [C]// Proceeding of the 11th IEEE International Symposium on Object Oriented Real-Time Distributed Computing. Los Alamitos, CA: IEEE Computer Society, 2008: 363–369.

[4] KLESH A T, CUTLER J W, ATKINS E M. Cyber-physical challenges for space systems [C]// 2012 IEEE/ACM Third International Conference on Cyber-Physical Systems. Beijing, China: IEEE/ACM, 2012: 45–54.

[5] Industrial Internet [EB/OL]. [2014–08–18]. http://www.ge.com/stories/industrial-internet, 2014/.

[6] LI Bo-hu, ZHANG Lin, WANG Shi-long, TAO Fei, CAO Jun-wei, JIANG Xiao-dan, SONG Xiao, CAI Xu-dong. Cloud manufacturing: A new service-oriented networked manufacturing model [J]. Computer Integrated Manufacturing Systems, 2010, 16(1): 1–7. (in Chinese)

[7] LI Bo-hu, ZHANG Lin, REN Lei, CHAI Xu-dong, TAO Fei, LUO Yong-liang, WANG Yong-zhi, YIN Chao, HUANG Gang, ZHAO Xin-pei. Further discussion on cloud manufacturing [J]. Computer Integrated Manufacturing Systems, 2011, 17(3): 449–457. (in Chinese)

[8] HOLLAND J H. Adaptation in natural and artificial system [M]. Ann Arbor: The University of Michigan Press, 1975: 141−153.

[9] GOLDBERG D E. Genetic algorithms in search, optimization and machine learning [M]. Reading, MA: Addison-Wesley, 1989: 89−145.

[10] EBERHART R C, KENNEDY J. A new optimizer using particle swarm theory [C]// Proceedings on 6th International Symposium on Micromachine and Human Science. Nagoya: IEEE Service Center, 1995: 39–43.

[11] KENNEDY J, EBERHART R C. Particle swarm optimization [C]// Proceedings of the IEEE International Conference on Neural Networks. Perth, Australia: IEEE Press, 1995: 1942–1948.

[12] CAR Z, BARISIC B, IKONIC M. GA based CNC turning center exploitation process parameters optimization [J]. Metallugica, 2009, 48(1): 47–50.

[13] WEI Yun, LI Dong-bo, TONG Yi-fei. Multi-objective reconfiguration and optimal scheduling of service-oriented networked collaborative manufacturing resource [J]. Transactions of the Chinese Society for Agricultural Machinery, 2012, 43(3): 193–199. (in Chinese)

[14] MA Xue-fen, DAI Xu-dong, SUN Shu-dong. Optimization deployment of networked manufacturing resources [J]. Computer Integrated Manufacturing Systems, 2004, 10(5): 523–527. (in Chinese)

[15] NAVALERTPORN T, AFZULPURKAR N V. Optimization of tile manufacturing process using particle swarm optimization [J]. Swarm and Evolutionary Computation, 2011, 1(2): 97–109.

[16] TAO Fei, ZHAO Dong-ming, HU Ye-fa, ZHOU Zu-de. Resource service composition and its optimal-selection based on particle swarm optimization in manufacturing grid system [J]. IEEE Transactions on Industrial Informatics, 2008, 4(4): 315–327.

[17] TAO Fei, ZHANG Lin, LU K, ZHAO Dong-ming. Study on manufacturing grid resource service optimal-selection and composition framework [J]. Enterprise Information Systems, 2012, 6(2): 237–264.

[18] TAO Fei, HU Ye-Fa, ZHOU Zu-de. Study on manufacturing grid & its resource service optimal-selection system [J]. International Journal of Advanced Manufacturing Technology, 2008, 37(9/10): 1022–1041.

[19] TAO Fei, ZHAO Dong-ming, ZHANG Lin. Resource service optimal-selection based on intuitionistic fuzzy set and non-functionality QoS in manufacturing grid system [J]. Knowledge and Information Systems, 2010, 25(1): 185–208.

[20] TAO Fei, ZHAO Dong-ming, HU Ye-fa, ZHOU Zu-de. Correlation-aware resource service composition and optimal-selection in manufacturing grid [J]. European Journal of Operational Research, 2010, 201(1): 129–143.

[21] TAO Fei, LAILI Yuan-jun, XU Li-da, ZHANG Lin. FC-PACO- RM: A parallel method for service composition optimal-selection in cloud manufacturing system [J]. IEEE Transactions on Industrial Informatics, 2013, 9 (4): 2023–2033.

[22] LIU Wei-ning, LIU Bo, SUN Di-hua. Study on multi-task oriented service composition in cloud manufacturing [J]. Computer Integrated Manufacturing Systems, 2013, 19(1): 200–209. (in Chinese)

[23] PEZZELLA F, MORGANTI G, CIASCHETTI G. A genetic algorithm for the flexible job-shop scheduling problem [J]. Computers & Operations Research, 2008, 35(10): 3202–3212.

[24] KACEM I, HAMMADI S, BORNE P. Approach by localization and multi-objective evolutionary optimization for flexible job-shop scheduling problems [J]. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, 2002, 32(1): 1–13.

[25] WANG Wan-liang, ZHANG Jing, XU Xin-li, JIE Jing, WANG Hai-yan. A hybrid discrete particle swarm optimization for Job shop Scheduling [C]// 2010 International Conference on Computational Aspects of Social Networks. Taiyuan, China: IEEE CS Press, 2010: 303–306.

[26] SHAO Xin-yu, LIU Wei-qi, LIU Qiong, ZHANG Chao-yong. Hybrid discrete particle swarm optimization for multi-objective flexible job-shop scheduling problem [J]. The International Journal of Advanced Manufacturing Technology, 2013, 67(12): 2885–2901.

[27] ZHANG Guo-hui, SHAO Xin-yu, LI Pei-gen, GAO Liang. An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem [J]. Computers & Industrial Engineering, 2009, 56(4): 1309–13l8.

(Edited by YANG Bing)