Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2016, Article ID 3938679, 10 pages
http://dx.doi.org/10.1155/2016/3938679

Research Article

A Hybrid Algorithm Based on Particle Swarm Optimization and Artificial Immune for an Assembly Job Shop Scheduling Problem

Hui Du,[1,2] Dacheng Liu,[1] and Mian-hao Zhang[3]

[1]*Department of IE, Tsinghua University, Beijing 100084, China*
[2]*Department of Mechanical Electronic Engineering, Zaozhuang University, Zaozhuang 277160, China*
[3]*Mechanical and Electrical Technology Research Center, Zhejiang Normal University, Jinhua 321004, China*

Correspondence should be addressed to Hui Du; du78913@163.com

Received 30 April 2016; Accepted 10 July 2016

Academic Editor: Roberto Dominguez

To produce the final product, parts need to be fabricated in the process stages and thereafter several parts are joined under the assembly operations based on the predefined bill of materials. But assembly relationship between the assembly parts and components has not been considered in general job shop scheduling problem model. The aim of this research is to find the schedule which minimizes completion time of Assembly Job Shop Scheduling Problem (AJSSP). Since the complexity of AJSSP is NP-hard, a hybrid particle swarm optimization (HPSO) algorithm integrated PSO with Artificial Immune is proposed and developed to solve AJSSP. The selection strategy based on antibody density makes the particles of HPSO maintain the diversity during the iterative process, thus overcoming the defect of premature convergence. Then HPSO algorithm is applied into a case study development from classical FT06. Finally, the effect of key parameters on the proposed algorithm is analyzed and discussed regarding how to select the parameters. The experiment result confirmed its practice and effectiveness.

1. Introduction

A schedule is an allocation of the tasks to time intervals on the machines and the aim is to find a schedule that minimizes the overall completion time. In the job shop scheduling problem (JSSP) n jobs have to be processed on m different machines; its procedure is shown as Figure 1. Each job consists of a sequence of tasks that have to be processed during an uninterrupted time period of a fixed length on a given machine. Scheduling problem is one of the most studied areas in the manufacturing industry and plays a significant role in successful production planning and control. JSSP, as the classical scheduling problem, has been extensively addressed in the past years [1].

Very often, studies of job shop scheduling problem ignore assembly relationship. Assembly constraint of product, especially mechanical products, is mostly assembly type ones. Manufacturing systems producing multiple products are common in many industries, where products are made from several parts and/or subassemblies that require machining operations in first stage and assembly operations at later stage. In the discrete manufacturing production process, the assembly is the final stage of the whole production, and the problems occurring in production workshops will be accumulated in the assembly stage.

Assembly relationship between the assembly parts and components has not been considered in the JSSP model. These problems do not provide any relationship between the multiple manufactured parts. In an Assembly Job Shop Scheduling Problem (AJSSP), the completion time of a child component becomes the release time of the parent part; hence, the multiple jobs become connected in a tree structure defined by the bill of material.

In a scheduling problem with assembly operations, usually there is a preassembly stage followed by an assembly stage. When the preassembly stage is machining stage, the

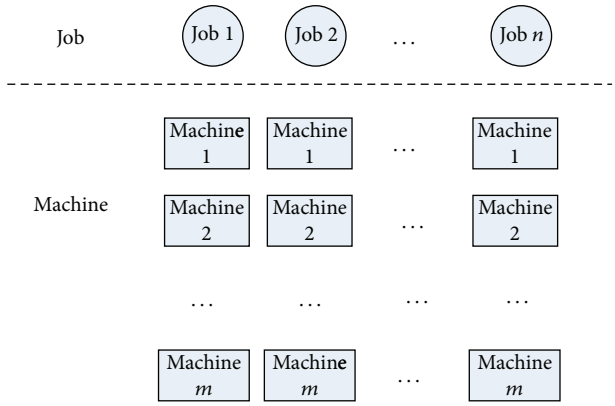

Figure 1: JSSP models.

parts are processed independently and then, they are assembled into products in the assembly stage. Scheduling for the parts machining and planning for the assembly operations have been independently studied; this may not lead to ideal results for the total production system. So, considering these two stages simultaneously in scheduling problems has received an increasing attention of researchers recently [2].

The first stage of two-stage assembly scheduling (TSAS) is fabrication stage with m parallel identical machines, while the second stage is the assembly stage utilizing n assembly machine [3]; the procedure is shown as Figure 2. There are n jobs to be scheduled and each job has $m + 1$ operations. For each job, the first m operations are conducted at the first stage by m machines in parallel and a final operation in the second stage by the assembly machine. Each machine can process only one job at a time. The last operation at the second stage may start only after all m operations at the first stage are completed.

As far as we have known, the two-stage assembly scheduling problem models appearing in the past years may be divided into four categories. The first type of TSAS is the first stage consisting of two independent machines, each of which produces its own type of components. The second stage consists of two identical and independent parallel machines. The processing of the second stage cannot begin until the processing of the first stage is finished [4].

This type is relatively simple. The second type is there are m machines at the first stage while there is only a single assembly machine at the second stage. There are n jobs to be scheduled and each job needs m operation, the m of which are done at the first stage by m machine in parallel and then an assembly operation utilizing a single machine in the second stage. This two-stage assembly scheduling model is the popular issue, and many researchers have studied this problem deeply [5–10]. The third type of TSAS is m machines in the fabrication stage with m machines and the assembly stage with two machines. The first stage consists of two independent machines, each of which produces its own type of components. The second stage consists of two identical and independent parallel machines.

The final type of TSAS is relatively complex and more realistic. This TSAS model is m machine in the first stage and more important at the second stage with n assembly machine [11]. The n machine of assembly stage may be work center, assembly worker, and so on. It means that the time needed to assemble a given job on various assembly machines is different. In this, a given job after being processed at the first stage can be assembled just by one of k nonidentical assembly machines at the second stage. A job in a given sequence which has been processed at the first stage must wait before being assembled if the job in the previous position is still being assembled.

The JSSP has been proved as NP-hard problem; therefore, the job shop scheduling problem including assembly is even NP-hard [12]. In the past years, a variety of scheduling algorithms have been developed to this problem. Fattahi et al. [2] discussed a hierarchical branch and bound algorithms to solve a hybrid flow shop scheduling problem with assembly operations.

Tian et al. provided a discrete particle swarm optimization algorithm to solve the two-stage assembly scheduling problem [3]. Sung and Kim provided a branch and bound algorithm by using the solution properties to solve the two-stage multiple-machine assembly scheduling problem. Moreover, the objective of the problem they considered was the sum of completion times [4]. Shokrollahpour et al. [5] discussed a two-stage assembly flow shop scheduling problem with minimization of weighted sum of makespan and mean completion time as the objective. A metaheuristic named imperialist competitive algorithm has been proposed to solve it. Liao et al. considered an assembly scheduling problem of N products to minimize the makespan. The problem is formulated as a mixed integer programming model, and several properties for finding optimal solutions are developed [8]. Yan et al. dealt with a two-stage assembly flow shop scheduling problem for minimizing the weighed sum of maximum makespan, earliness, and lateness [9]. They proposed a hybrid variable neighborhood search-electromagnetism-like mechanism algorithm to solve the problem.

To solve JSSP in assembly environment, Wang et al. [12] put up a genetic algorithm based on feasible solution space searching. Wong and Ngan [13] proposed two hybrid evolutionary algorithms with Lagrangian relaxation technique to minimize the makespan in AJSSP. Al-Anzi and Allahverdi [14] proposed an artificial immune system for AJSSP with two machines in assembly stage. The objective aims to minimize total or mean completion time for all jobs. At another literature, they put up a hybrid Tabu Search (TS) heuristic to solve the AJSSP [15]. Xu and Nagi consider an assembly scheduling problem with tree-structured precedence constraints [16]. They propose a mixed integer linear programming formulation and solve the problem with a Lagrangian relaxation approach. Seidgar et al. considered a two-stage assembly flow shop problem where m parallel machines are in the first stage and an assembly machine is in the second stage [6]. The objective is to minimize a weighted sum of makespan and mean completion time for n available jobs. They employed an imperialist competitive algorithm as solution approach.

Komaki and Kayvanfar described minimizing the makespan of two-stage assembly flow shop. For the studied

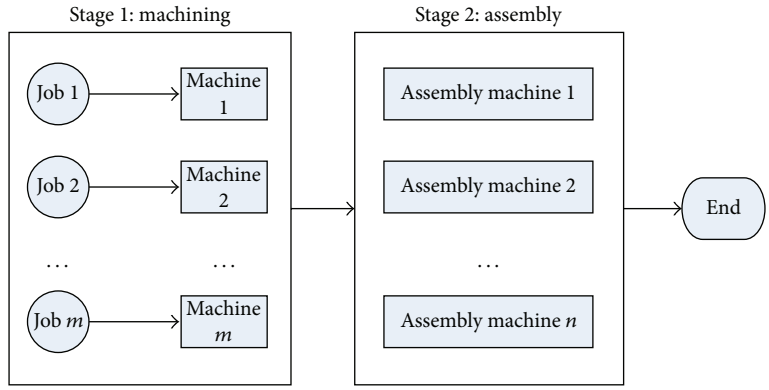

Figure 2: The two-stage assembly scheduling problem.

problem, a lower bound (LB), dispatching rules, and heuristic algorithms are proposed. Also, a novel metaheuristic algorithm called Grey Wolf Optimizer (GWO) algorithm is proposed [7]. Furthermore, Komaki et al. find the schedule which minimizes completion time of the last product. For the considered problem, lower bound, heuristic algorithms, and two met heuristic techniques based on artificial immune system are developed [17].

AJSSP has very important practical significance. Generally, the scheduling program must resolve the assembly constraints; otherwise, it is not realistic and has no practical engineering applications value.

The product of most mechanical products includes manufacturing and assembling, and the parts used in assembly should be in complete set. Even if some part has been manufactured ahead of time, the assembly cannot start until the parts in assembly constraint relations have been processed. As a result, manufacturing is not consistent with assembly and the production cycle is prolonged. The scheduling scheme without assembly constraint is not applicable to production. Only when manufacturing and assembly are synchronously optimized and scheduled can it be ensured that all parts of a product are finished or assembled at the time of the general assembly of tooling and can it be avoided that assembly cannot start due to lack of parts.

Product is composed of many parts and each part is processed and assembled separately. When processed parts are assembled into products, they are in assembly constraint relations. The key to scheduling is to control the production progress of part sets and to ensure the completeness and synchronization of production so that the parts in assembly constraint relations may be in complete sets.

In the discrete manufacturing production process, the assembly is the final stage of the whole production, problems occurring in the first few production workshop will be accumulated in the assembly stage. JSSP, as the classic scheduling problem, has been widely and deeply studied, but the model of JSSP without considering assembly relationship between the assembly parts and components. Overall, the research of scheduling problem with assembly constrain was relatively few in these years, and it has not received due attention and research.

The particle swarm optimization (PSO) algorithm is a popular global optimization method, which has been applied extensively in solving many optimization fields in these years. As a population-based searching technique it has high search efficiency by combining local search (by self-experience) and global one (by neighboring experience). PSO embodies some attractive characteristics: (1) compared with other evolutionary algorithms, it possesses memory; for example, the characteristics of the good solutions are retained by all particles. (2) PSO has constructive cooperation among the particles; particles in the swarm share their information. But, similar to evolutionary algorithms, PSO is easy to be a premature convergence so that exploration and exploitation should be enhanced and well balanced to achieve better performance [18].

In this paper, we provide the AJSSP with the objective of minimizing total completion time. Thereafter, a hybrid particle swarm optimization (HPSO) integrated PSO with Artificial Immune (AI) is proposed and developed to solve AJSSP. The selection strategy based on antibody density makes the particles of HPSO maintain the diversity during the iterative process, thus overcoming the defect of premature convergence of PSO. In order to test the effective and efficiency of proposed algorithm, a study case based on classical FT06 is developed. Furthermore, we completed comparative test between HPSO and other single algorithms. The experiment result confirmed its practice and effectiveness.

The rest of this paper is organized as follows: in Section 2, the problem is described completely and a numbering example is presented. The solving approach and solution methodology will be expressed in Section 3, and the standard PSO and HPSO algorithm are presented in detail in this section. In Section 4, a design of the problems and experiments is described and then comparisons of the results and performance evaluation of the presented algorithm are done. Finally, concluding remarks and summary of the work and direction for the future research are given in Section 5.

2. Problem Description

In the presence of an assembly stage, all completed jobs from the BOM of the product need to be assembled to form

the final products. Assembly can be seen as the process to construct a final product from its root components. AJSSP, as the extension of JSSP, includes assembly constraint conditions. In AJSSP, the part manufacturing process of different units should be strictly restricted by assembly, but the part processes of the same unit may be paralleled. Both mechanical manufacturing process and assembly process are called procedures; and both the machinery necessary for manufacturing and the equipment (assemblers) needed for assembly are called equipment. Thus AJSSP may be described as the follows:

(1) Part set $L = \{L_1, L_2, \ldots, L_n\}$, and L_i is part i, and the number of parts to be scheduled is n.

(2) Constraint set $R = \{R_1, R_2, \ldots, R_v\}$, and R_v is assembly constraint condition number v.

(3) Manufacturing machine set $M = \{M_1, M_2, \ldots, M_m\}$, and there are a total of m machines available, and M_m is number m manufacturing machine.

(4) Process set: $O = \{O_1, O_2, \ldots, O_n\}$; O_{il} stands for process number l of part i; M_{il} is the machines needed by the process, and $M_{il} \subset M$.

(5) The start time for process O_{il} is S_{ik} and t_{il} is corresponding manufacturing time; then

$$S_{ik} + t_{ik} < S_{i(k+1)}, \quad S_{ik} > 0. \tag{1}$$

(6) Assembly constraint relations may be described as follows: let P be an assembly, representing a unit or product; the direct subpart-set B of P is $B = \{B_1, B_2, \ldots, B_k\}$; the relation set between P and B is Q; then $Q \subset R$, and the start time S_{P1} of the first assembly process of P meets such condition: $S_{P1} > (S_{bl} + t_{bl})$; S_{P1} is the last process of any subpart and t_{bl} is the corresponding manufacturing time.

In order to simplify AJSSP model and make the model meet actual demands, the following assumptions should be made about AJSSP:

(1) During the whole manufacturing process, each manufacturing procedure can be finished only with the named equipment (multiple pieces of equipment available), and the assembly can be carried out only by the designated assembly team (several assembly teams available).

(2) The next process of any operation cannot start until the previous process has finished, and with the same machine the next manufacturing task cannot start until the previous one has finished, so that the machine may be available all the time.

(3) Each operation must be manufactured on a machine according to process route and in designated order, and at one moment with one machine only one operation can be manufactured. Furthermore, the manufacturing cannot be interrupted.

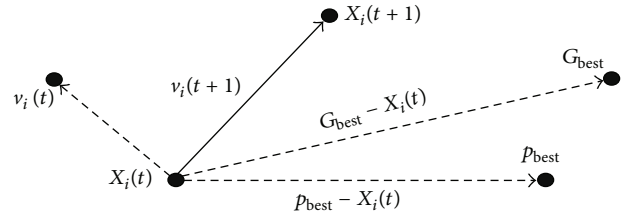

Figure 3: Motion of the particles in the PSO.

The above description of model shows that AJSSP mainly involves three constraint conditions: process constraint, resources constraint, and assembly constraint. To solve AJSSP means after the three constraint conditions are met determining the start time of all processes and tasks, the start time of unit assembly, and the start time of the final general assembly and optimizing the goals.

The researchers have considered several optimization objectives for AJSSP, like minimizing the makespan, maximum tardiness/tardiness, total (weighted) completion time, and total (weighted) tardiness/tardiness. As to the scheduling objective, different industry environment has different optimization objective. In this paper, the scheduling objective of AJSSP is to get a feasible schedule with the minimizing complete time.

3. The Proposed Solving Approach

An assembly stage should be appended to JSSP for the final product; the problem then becomes Assembly Job Shop Scheduling Problem. The JSSP has been proved NP-hard, so the AJSSP is also strongly NP-hard [12]. Because of the NP-hard characteristics of job shop scheduling, it is usually very hard to find its optimal solution, and an optimal solution in the mathematical sense is not always necessary in practice. Researchers turned to search its near-optimal solutions with all kinds of heuristic algorithms. In this paper, we provide a hybrid particle swarm optimization integrated PSO with Artificial Immune (AI) to solve AJSSP.

3.1. Particle Swarm Optimization. Particle swarm optimization (PSO), which was first developed by Kennedy and Eberhart, is an evolutionary algorithm that simulates the social behavior of bird flocking to desired places [19]. Similar to a bird that flies to the food, one particle moves its position to a better solution with a velocity which is dynamically adjusted according to its own flying experience and its companions' flying experience [20]. The motion of particles in the PSO is shown in Figure 3.

PSO starts by initializing a population of random solutions and searches for optima by updating generations. In the PSO method, the position of a particle represents a solution of a dedicated problem, particles spread in the solution space, and each particle would move to a new position for the global optimal solution based on the global experience of the swarm and the individual experience of the particle. In contrast to general heuristic methods, PSO has advantage of simple principle and parameters and fast convergence speed [21].

The PSO consists of a swarm of particles in the space; the position of a particle is indicated by a vector which presents a solution. PSO is initialized with a population of randomly positioned particles and searches for the best position with best fitness. In each iteration, every particle flies to a new position and this new position is guided by a velocity vector. Each particle is updated iteratively by the following formula [22]:

$$v_{i+1} = \chi \cdot v_i + c_1 \cdot \text{rand}\,() \left(p_{\text{best}} - x_i\right) + c_2 \cdot \text{rand}\,() \left(G_{\text{best}} - x_i\right), \quad (2)$$
$$x_{i+1} = x_i + v_{i+1}.$$

The notations used in the proposed PSO are as follows: v_i represents the distance to be traveled by particle i from its current position, x_i is the position of particle i, p_i is the best previous position of particle i, p_{best} represents the best position that the particle i has experienced, G_{best} is the best position among all particles in the population, c_1, c_2 are acceleration coefficients, to control the maximum step size, and χ is inertia weight, to control the impact of previous velocity on current one.

In the above formulas, the first one is used to calculate the new velocity according to its previous velocity and to the distance of its current position from both its own best historical position and its neighbor's best position.

The standard PSO algorithm is applied for AJSSP including the follow steps [21].

Step 1. Parameter initialization: generate pop initial schedule: x_i, $i = 1, \ldots,$ pop. If each of generated schedules is infeasible, regenerate it until all i schedules are feasible.

Step 2. Find the objective value for each particle.

Step 3. Compare particle's fitness evaluation with particle's p_{best}; if current value is better than p_{best}, then set p_{best} value to the current value and the p_{best} location equal to the current location. In the same way, compare fitness evaluation with the population's overall previous best. If current value is better than G_{best}, then set G_{best} value to the current value.

Step 4. Update the position and velocity of the particles according to (2). Find the best position that the particle i has experienced and the best position among all particles in the population.

Step 5. Loop to Step 2 until a criterion is met, usually a sufficiently good fitness or a maximum number of generations.

3.2. The Proposed HPSO for AJSSP. Similar to other evolutionary algorithms, the standard PSO is often easy to be a premature convergence so that exploration and exploitation of PSO should be enhanced and well balanced to achieve better performance. On the other hand, Artificial Immune is a stochastic searching algorithm with a jumping property. Artificial immune algorithm uses an antibody diversity itself and self-regulation, to make the population maintain diversity and avoid falling into local optimization. So, we employ the jumping mechanism of AI into PSO in order to achieve the results with higher quality.

Artificial immune algorithm, inspired by the biological immune system, has emerged as a novel and a maturing computational paradigm [23]. Inspired by the underlying characteristics of biological immune system, AI gleaned ideas from immunology in order to explore, derive, and apply different biologically inspired immune mechanisms aim at computational problem solving. AI has been applied successfully to a variety of optimization problems, and studies have shown that they possess several attractive immune properties that allow evolutionary algorithms to avoid premature convergence and improve local search [24]. In the process of solving practical problems, the scheduling problem is seen as antigen, and the antibody is the feasible solution of scheduling problem. Then, the final solution can be attained through interaction between antigen and antibody. In the HPSO algorithm, each particle represents a feasible solution.

During the iterative process of standard PSO, the premature convergence of particles decreases the algorithm's searching ability. By introducing the selection strategy based on antibody density, an immunity based PSO model was proposed, and its application to AJSSP was given. The selection strategy based on antibody density makes the particles of HPSO maintain the diversity during the iterative process, thus overcoming the defect of premature convergence. The flowchart of proposed HPSO algorithm for AJSSP is shown in Figure 4.

Step 1. Parameter initialization is such as c_1, c_2, m, n, and $T_{\max}$.

Step 2. Randomly generate n particles as the initialized particles, which are the antibody.

Step 3. Calculate the fitness of particles $f(X_i)$, and compare particle's fitness evaluation with particle's p_{best}; if current value is better than p_{best}, then set p_{best} value to the current value and the p_{best} location equal to the current location. In the same way, compare fitness evaluation with the population's overall previous best. If current value is better than G_{best}, then set G_{best} value to the current value.

Step 4. Update the position and velocity of the particles according to (2). Find the best position that the particle i has experienced and the best position among all particles in the population. By this way, a new generation of particle swarm is created.

Step 5. Measure the updated N particles to find out whether they meet the diversity condition. If they meet the diversity condition, turn to the Step 7. If not, then turn to the Step 6.

Step 6. Randomly generate M new particle, and create $(m+n)$ particles swarm. After this, select the new n particle according to the probability $p_d(x_i)$.

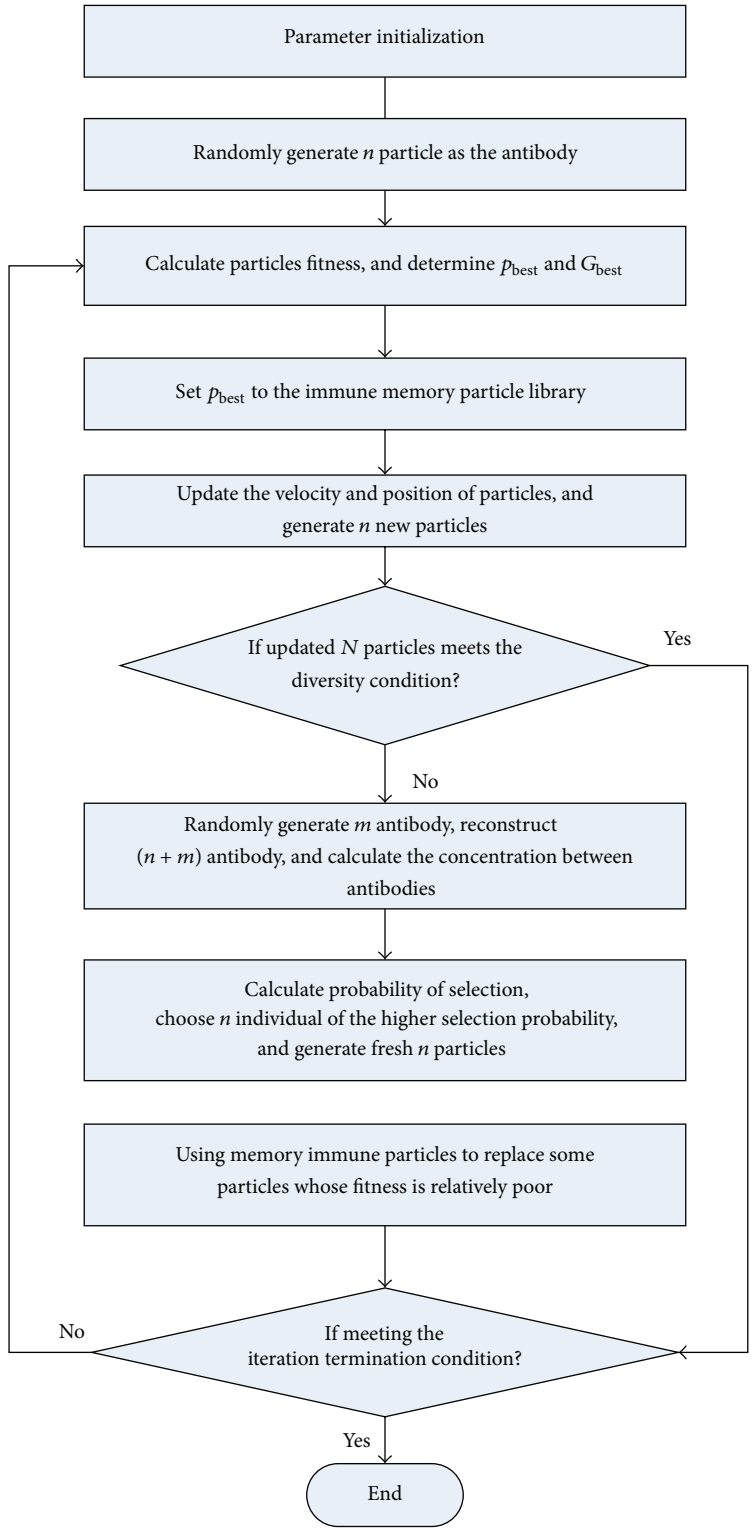

Figure 4: Flowchart of HPSO algorithm for *assembly* job shop scheduling problem.

Step 7. Perform vaccination and vaccine selection operation, using memory immune particles to replace some particles whose fitness is relatively poor in the current particle swarm.

Step 8. Loop to Step 3 until a criterion is met, usually sufficiently good fitness or a maximum number of generations.

In the above step of HPSO algorithm, the target of scheduling and optimization is to make the manufacturing duration of tooling the shortest. So let us adopt the following fitness formula:

$$f(X_i) = \frac{\delta}{\Delta T} = \frac{\partial}{t_c - t_s}, \tag{3}$$

where $f(X_i)$ is the fitness formula of particles, ΔT is the longest activity duration of all parts and units, t_c is the completion time of parts, t_s is the start time of parts, and δ is the coordination coefficient of fitness formula.

In particle immunity, the selection mechanism based on antibody concentration is adopted to select individuals. Particles with relative higher probability are selected to form new particles, which are used to be immunization. If the fitness value of children particles after Vaccination is lower than that of their parents, the particles will be refused; else, the particles will be kept and form a new generation of subparticle swarm. The antibody concentration is an important indicator for measuring the diversity of antibodies, and selection probability p_d based on antibody concentration is calculated by [21]

$$p_d(x_i) = \frac{1/D(x_i)}{\sum_{i=1}^{N+M}(1/D(x_i))}. \tag{4}$$

In formula (4), n is the total number of antibodies; $D(x_i)$ is the concentration of antibody x_i, which is the ratio of the number of antibodies whose similarity is less than predetermined value to the total number. $D(x_i)$ may be calculated as

$$D(x_i) = \frac{1}{\sum_{j=1}^{N+M}\left|f(x_i) - f(x_j)\right|}, \quad i = 1, 2, \ldots, N+M. \tag{5}$$

According to formula (5), the chance of being chosen of particle is inverse to the similarity. So, as to particles (antibody), the smaller the similarity, the higher the probability of being selected. This selected mechanism based on probability ensures the diversity of antibody, so, by this way, the contractility and convergence capability of proposed HPSO algorithm are improved to a large extent.

Based on formulas (4) and (5), selection probability p_d may be calculated according the following formula:

$$p_d(x_i) = \frac{1/D(x_i)}{\sum_{i=1}^{N+M}(1/D(x_i))} = \frac{\sum_{j=1}^{N+M}\left|f(x_i) - f(x_j)\right|}{\sum_{i=1}^{N+M}\sum_{j=1}^{N+M}\left|f(x_i) - f(x_j)\right|}, \quad i = 1, 2, \ldots, N+M. \tag{6}$$

HPSO optimizes particle swarm algorithm by introducing immunization and immune selection. The optimal scheduling solution meeting constraint conditions is called antigen in immune system and the feasible scheduling solution antibody in immune system. In the iterative evolution of algorithm, the selection mechanism of artificial immune is used to update and replace particles to make them diverse in update and iteration. The particles of each fitness level are kept at certain concentration, to prevent particles from reaching partial optimized values too early, thus improving the comprehensive search capability of algorithm.

4. Computational Experiments and Results

4.1. The Example of a AJSSP. In order to verify the effectiveness and superiority of the proposed algorithm, an example of AJSSP is put forward in this section. The structure of product P is showed in Figure 5(a). P is composed of four units, namely, A, B, C, and D. In fact, the construct of P in this case is similar to the product in Figure 1. The only difference in product construction is part B. In the case study, to make it a general case, part B is added.

In this, B is outsourced and the other three are manufactured. A is assembled with parts E and F and C with parts G, H, and I. The manufacturing of unit D and parts E, F, G, H, and I involves six processes ($O_{i1} \sim O_{i6}$). The AJSSP is shown in Figure 5(b).

The relations between product structures and nodes in AJSSP are, respectively, $(P \rightarrow A_{21})$, $(A \rightarrow A_{11})$, $(C \rightarrow A_{12})$, $(D \rightarrow O_{66})$, $(E \rightarrow O_{16})$, $(F \rightarrow O_{26})$, $(G \rightarrow O_{36})$, $(H \rightarrow O_{46})$, and $(I \rightarrow O_{56})$. As B is purchased part and its manufacturing needs not to be considered, there is no node in AJSSP corresponding to B. It is just necessary to take into account the purchase lead time of B when formulating a demand plan for complete sets of materials. To simplify the model, let us assume the man-hour for assembly is too little to be considered here. The operation time and require equipment are shown in detail in Table 1. $N(p, q)$ means that the Nth job, including processing and assembly operations, requires equipment q, and the corresponding processing time is q unit. In Table 1, for example, the array $1(2, 1)$ means that job 1 (also the first job of part E) requires equipment 2 to complete process, and the processing time is 1 unit. In the same way, $37(7, 20)$ mean that the thirty-seventh job requires equipment 7 to complete assembly and the processing time is 1 unit. It should be noted that the machines are numbered starting with 0.

The manufacturing processes of parts of the same unit are in conflict in use of equipment. The first process O_{11} of unit E, for example, needs the same manufacturing equipment 3 as needed by process O_{31}. And different part processes of the same unit that are in assembly constraint relations are in conflict in the use of resources. The third process O_{13} of E, for example, needs the same manufacturing equipment 2 as needed by process O_{21}.

4.2. Parameter Setting and Compute Results. As to the example AJSSP, the objective is minimizing complete time. In this, δ is the coordination coefficient of fitness formula; set $\delta = 100$. Parameters of HPSO are as follows: particle number $P = 100$; the max number of iterations $n = 100$; the learning coefficient $c_1 = c_2 = 100$; the similarity threshold between

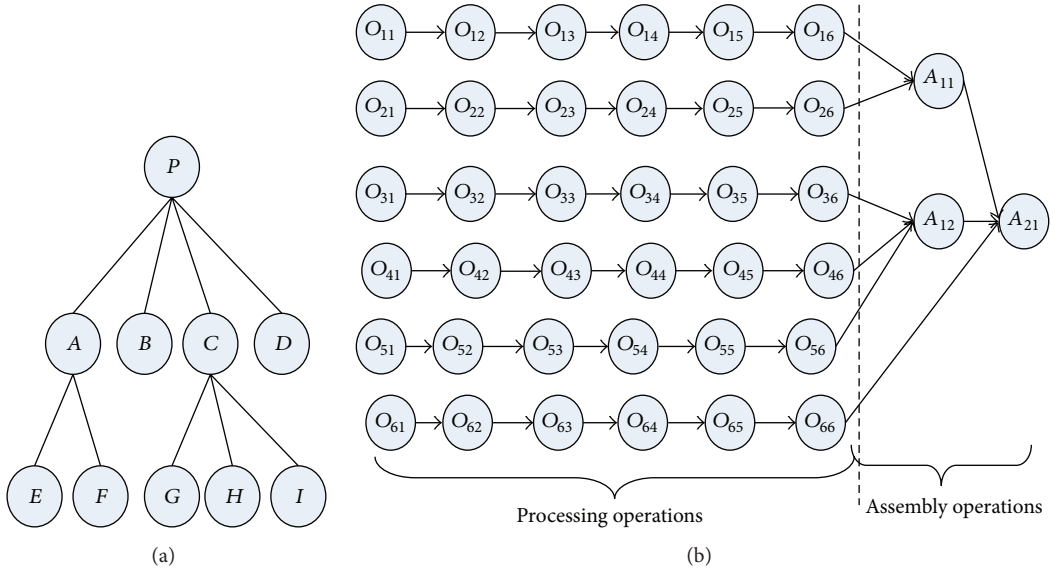

Figure 5: A case of AJSSP.

Table 1: Operations, equipment, and processing time of the case.

	Processing 1	2	3	4	5	6	Assembly
E	1 (2, 1)	2 (0, 3)	3 (1, 6)	4 (3, 7)	5 (5, 3)	6 (4, 6)	—
F	7 (1, 8)	8 (2, 5)	9 (4, 10)	10 (5, 10)	11 (0, 10)	12 (3, 4)	—
G	13 (2, 5)	14 (3, 4)	15 (5, 5)	16 (0, 9)	17 (1, 1)	18 (4, 7)	—
H	19 (1, 5)	20 (0, 5)	21 (2, 5)	22 (3, 3)	23 (4, 8)	24 (5, 9)	—
I	24 (2, 9)	26 (1, 3)	27 (4, 5)	28 (5, 4)	29 (0, 3)	30 (3, 1)	—
D	31 (1, 3)	32 (3, 3)	33 (6, 9)	34 (0, 10)	35 (4, 4)	36 (2, 1)	—
A	—	—	—	—	—	—	37 (7, 20)
C	—	—	—	—	—	—	38 (7, 30)
P	—	—	—	—	—	—	39 (7, 15)

particles (antibodies) distance = 1; the selection probability based on antibody concentration $p_d = 0.6$. And inertia weight $\gamma = 0.7$ and $M = 30$.

Use proposed HPSO hybrid algorithm to seek the solution to this AJSSP as showed in Figure 5 in MATLAB 2010b soft environment. The computer should have AMD CPU-2.0 GHz, 512 M memory, and Windows XP operation system. Algorithm should be carried out 50 times. Figure 6 displays the algorithm convergence curve corresponding to the optimal solution obtained from 50 times of algorithm.

In addition, Figure 7 is the task scheduling Gantt chart corresponding to the optimal solution result. The results of the 20 operations show that the results of AJSSP scheduling example obtained with PSOAI are steady and are basically converged on the 21st generation, with max fitness $f(X_i)$, and minimum T, $\min T = 106$.

In fact, the processing operation of six parts in the example AJSSP is the classic F6 scheduling problem. Here FT6, the manufacturing problem of AJSSP, is used to verify whether proposed hybrid algorithm is effective in solving such scheduling problems. The optimal time span of FT6 is 55 [25], and the best result obtained with PSOAI algorithm is also 55, proving it is effective in solving scheduling problem with PSO.

The scheduling result of FT06 which takes into consideration of assembly constraint is 74, more than the result of FT06 (55), and the total duration is 106. This displays that assembly constraint makes scheduling complicated to a great extent, so, a scheduling scheme will surely be infeasible if not considering assembly constraint. On the other hand, if assembly is considered only after the optimized scheduling of mechanical manufacturing, or manufacturing and assembly are considered separately, and according to the Longest Processing Time first Rule (LPT), the total duration of the example AJSSP will be 120 ($T = 55 + 30 + 20 + 15$), which is much longer than the result drawn in this paper.

As to some complex mechanical products, assemblies require relatively long time. For some large machinery and equipment, their assembly jobs will spend even to ten days or months. In this assembly environment, the AJSSP is extremely important in minimizing the complete time; therefore, the proposed HPSO algorithm will attain even better scheduling result. And the advantages of PSOAI would be more prominent.

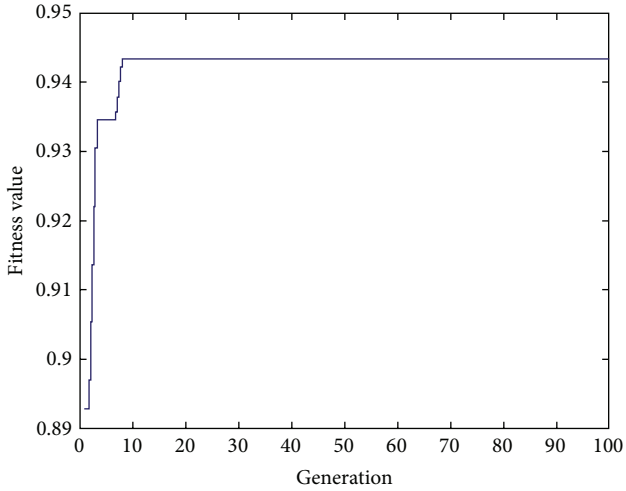

Figure 6: Convergence of HPSO algorithm to minimize makespan for the example AJSSP.

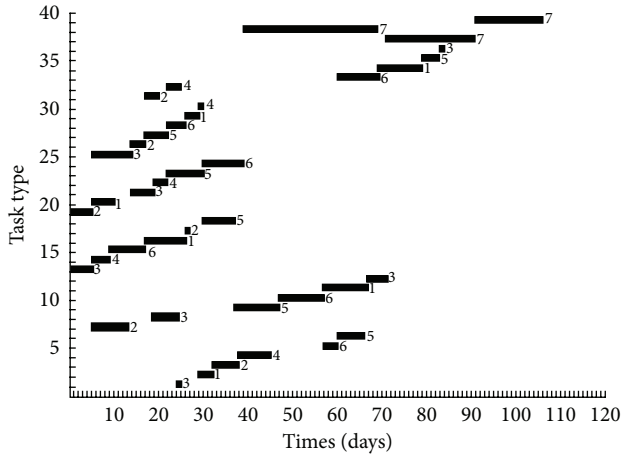

Figure 7: Gantt chart for the example AJSSP.

4.3. Parameter Sensitivity Analysis. The proposed hybrid algorithm is integrated PSO and AI, so the algorithm relates to several parameters from particle swarm and artificial immune. Related parameters the algorithm will directly affect its performance, and the parameter selection is important in solving scheduling problem. In this, we analyze the effect of some key parameters on the performance of the algorithm.

N denotes the number of particles. Generally, and the value of N is set as 20 to 30. The maximum value of N can fully meet the requirements of common scheduling problems. But as to the much more difficult problems, the number of particles may be relatively large, even near to hundreds.

Parameters c_1 and c_2 are acceleration coefficients, used to control the maximum step size.

χ is inertia weight, to control the impact of previous velocity on current particle. Inertia weight w is an important parameter to search ability of PSO algorithm. A large inertia weight facilitates searching new area while a small inertia weight facilitates fine-searching in the current search area.

Social learning rate c also has an important influence on the search ability of PSO algorithm; large c_1 make the particle strongly attracted by G_{best} and quickly converge to G_{best} while small c_1 weaken the attraction of G_{best} and particle can search in a big area. Suitable parameters provide a balance between global exploration and local exploitation and result in less iterations on average to find a sufficiently optimal solution.

In order to strike a balance between global and local searching ability, the inertia weight with cyclical attenuation was presented to improve the effectiveness of PSO in the proposed hybrid method. According to the literature result [26], cyclical attenuation of inertia weight is adopted based on

$$\chi = \left|\chi_{\max} \exp\left(-0.05t \cos t\right)\right| + \chi_{\min}, \tag{7}$$

where $\chi_{\max}$ and $\chi_{\min}$ are the maximum and minimum value of inertia weight, respectively. In this paper, their value is set to 0.7 and 0.1, and the change cycle of χ is π. From the cyclical attenuation curve, we can learn the χ cyclical fluctuations as well as constant attenuation. In the initial stage of every cycle, the value of χ is smaller relatively, so particles have strong local search ability; in the intermediate stage of every cycle, PSO has an appropriate inertia weight, thus, it has a strike balance between the global and local search; At the same time, in the late stage of every cycle, the particles have adequate global search ability. By this way, the PSO with adaptive inertia weight overcome the decline capabilities in the late evolution and accelerate algorithm convergence speed.

5. Conclusion

In this paper, a new particle swarm algorithm based on artificial immune is employed to solve the AJSSP. The complete time minimization is the objective of scheduling. The principle of particle swarm optimization algorithm and steps of proposed HPSO method are described in detail, and hybrid algorithm is applied into a case study development from classical FT06. The experiment result confirmed its practice and effectiveness.

Just like other stochastic optimization algorithms, HPSO is also sensitive to its parameters. In this paper we select the parameters based on literature suggested. And this may be negative to the effective of HPSO at a certain extent. So, how to optimize these parameters and to further improve its efficiency of solving AJSSP is still an interesting valuable subject in the future.

On the other hand, setup times were not considered in the above AJSSP model. Setup time is needed whenever the machining machine starts processing components, or the item of component is switched on the machine. Therefore, it is also worthwhile to study the AJSSP with setup times.

Finally, future research may consider different objective function such as minimizing the total tardiness, minimization of weighted sum of makespan, and mean completion time as the scheduling objective since they are often to be considered in the practical environment. So, the proposed scheduling method can be expected to be practically applied in the real-life production systems in the future.

Competing Interests

The authors declare that they have no competing interests.

Acknowledgments

The authors gratefully acknowledge financial support from the China Postdoctoral Science Foundation (no. 2016M591193) and the Doctoral Fund Support Program of Zaozhuang University, China (no. 20141020703).

References

[1] J. W. Barnes and J. B. Chambers, "Solving the job shop scheduling problem with tabu search," *IIE Transactions*, vol. 27, no. 2, pp. 257–263, 1995.

[2] P. Fattahi, S. M. Hosseini, F. Jolai, and R. Tavakkoli-Moghaddam, "A branch and bound algorithm for hybrid flow shop scheduling problem with setup time and assembly operations," *Applied Mathematical Modelling. Simulation and Computation for Engineering and Environmental Systems*, vol. 38, no. 1, pp. 119–134, 2014.

[3] Y. Tian, D. Liu, D. Yuan, and K. Wang, "A discrete PSO for two-stage assembly scheduling problem," *International Journal of Advanced Manufacturing Technology*, vol. 66, no. 1–4, pp. 481–499, 2013.

[4] C. S. Sung and H. A. Kim, "A two-stage multiple-machine assembly scheduling problem for minimizing sum of completion times," *International Journal of Production Economics*, vol. 113, no. 2, pp. 1038–1048, 2008.

[5] E. Shokrollahpour, M. Zandieh, and B. Dorri, "A novel imperialist competitive algorithm for bi-criteria scheduling of the assembly flowshop problem," *International Journal of Production Research*, vol. 49, no. 11, pp. 3087–3103, 2011.

[6] H. Seidgar, M. Kiani, M. Abedi, and H. Fazlollahtabar, "An efficient imperialist competitive algorithm for scheduling in the two-stage assembly flow shop problem," *International Journal of Production Research*, vol. 52, no. 4, pp. 1240–1256, 2014.

[7] G. M. Komaki and V. Kayvanfar, "Grey Wolf Optimizer algorithm for the two-stage assembly flow shop scheduling problem with release time," *Journal of Computational Science*, vol. 8, pp. 109–120, 2015.

[8] C.-J. Liao, C.-H. Lee, and H.-C. Lee, "An efficient heuristic for a two-stage assembly scheduling problem with batch setup times to minimize makespan," *Computers and Industrial Engineering*, vol. 88, pp. 317–325, 2015.

[9] H.-S. Yan, X.-Q. Wan, and F.-L. Xiong, "A hybrid electromagnetism-like algorithm for two-stage assembly flow shop scheduling problem," *International Journal of Production Research*, vol. 52, no. 19, pp. 5626–5639, 2014.

[10] A. Mozdgir, S. M. T. Fatemi Ghomi, F. Jolai, and J. Navaei, "Two-stage assembly flow-shop scheduling problem with non-identical assembly machines considering setup times," *International Journal of Production Research*, vol. 51, no. 12, pp. 3625–3642, 2013.

[11] J. K. Lenstra, A. H. Rinnooy Kan, and P. Brucker, "Complexity of machine scheduling problems," in *Annals of Discrete Mathematics*, vol. 1 of *Studies in Integer Programming*, pp. 343–362, North-Holland, Amsterdam, Netherlands, 1977.

[12] F.-J. Wang, G.-K. Zhao, Z.-Y. Jia, X.-H. Lu, and L.-P. Wang, "Assembly job shop scheduling based on feasible solution space genetic algorithm," *Computer Integrated Manufacturing Systems*, vol. 16, no. 1, pp. 115–120, 2010.

[13] T. C. Wong and S. C. Ngan, "A comparison of hybrid genetic algorithm and hybrid particle swarm optimization to minimize makespan for assembly job shop," *Applied Soft Computing Journal*, vol. 13, no. 3, pp. 1391–1399, 2013.

[14] F. S. Al-Anzi and A. Allahverdi, "An artificial immune system heuristic for two-stage multi-machine assembly scheduling problem to minimize total completion time," *Journal of Manufacturing Systems*, vol. 32, no. 4, pp. 825–830, 2013.

[15] F. S. Al-Anzi and A. Allahverdi, "Better heuristics for a two-stage multi-machine assembly scheduling problem to minimize total completion time," *International Journal of Operations Research*, vol. 9, no. 2, pp. 66–75, 2012.

[16] J. Xu and R. Nagi, "Solving assembly scheduling problems with tree-structure precedence constraints: a Lagrangian relaxation approach," *IEEE Transactions on Automation Science and Engineering*, vol. 10, no. 3, pp. 757–771, 2013.

[17] G. M. Komaki, E. Teymourian, and V. Kayvanfar, "Minimising makespan in the two-stage assembly hybrid flow shop scheduling problem using artificial immune systems," *International Journal of Production Research*, vol. 54, no. 4, pp. 963–983, 2015.

[18] R. C. Eberhart and J. Kennedy, "New optimizer using particle swarm theory," in *Proceedings of the 1995 6th International Symposium on Micro Machine and Human Science*, pp. 39–43, IEEE Service Center, Nagoya, Japan, October 1995.

[19] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the 1995 IEEE International Conference on Neural Networks*, pp. 1942–1948, Washington, DC, USA, December 1995.

[20] A. Jamili, M. A. Shafia, and R. Tavakkoli-Moghaddam, "A hybrid algorithm based on particle swarm optimization and simulated annealing for a periodic job shop scheduling problem," *International Journal of Advanced Manufacturing Technology*, vol. 54, no. 1–4, pp. 309–322, 2011.

[21] W.-J. Xia and Z.-M. Wu, "A hybrid particle swarm optimization approach for the job-shop scheduling problem," *International Journal of Advanced Manufacturing Technology*, vol. 29, no. 3-4, pp. 360–366, 2006.

[22] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *Proceedings of the IEEE World Congress on Computational Intelligence*, pp. 69–73, Anchorage, Alaska, USA, May 1998.

[23] L. N. De Castro and J. Timmis, *Artificial Immune Systems: A New Computational Intelligence Approach*, Springer, New York, NY, USA, 2002.

[24] E. Hart and J. Timmis, "Application areas of AIS: the past, the present and the future," *Applied Soft Computing Journal*, vol. 8, no. 1, pp. 191–201, 2008.

[25] Benchmark Models, http://amsterdamoptimization.com/benchmarkmodels.html.

[26] J. Xu, S.-M. Fei, S.-Y. Zhang, and Y.-D. Shi, "Adaptive particle swarm optimization for the project scheduling problem with dynamic allocation of resource," *Computer Integrated Manufacturing Systems*, vol. 17, no. 8, pp. 1790–1797, 2011.