12

Hybrid particle swarm algorithm for job shop scheduling problems

Xiaoyu Song
School of Information and Control Enineering, Shenyang Jianzhu University
China

1. Introduction

Particle swarm optimization (PSO) algorithm is a kind of random optimization algorithm based on swarm intelligence. Swarm intelligence of PSO is produced by cooperation and competition between particles, which is used for guiding optimization search. PSO has been studied widely in many applications due to its good global searching ability. Currently PSO has been widely used in function optimization, neural network training, pattern classification, system control and other applications. The research on PSO in recent years indicates that PSO has fast convergence speed and good quality in solutions and fine robustness on optimization in multidimensional space functions or in dynamic objectives, which is suitable for project applications. In this chapter, we firstly introduce searching mechanism and algorithm processes of PSO. Then, some important problems are solved when PSO is used for job shop scheduling problems (JSSP), such as hybrid algorithms between particle swarm and other algorithms (HPSO), its deadlock issues, and the proof of PSO and HPSO convergence. This chapter can provide guides effectively for readers who apply particle swarm optimization algorithm.

2. Particle Swarm Optimization Algorithm for JSSP

Particle swarm optimization (PSO) is an evolutionary computation technique developed by Kennedy and Eberhart in 1995. The particle swarm concept was motivated by the simulation of social behaviors. PSO algorithm constitutes the simple conduct rules for each particle, remembers the best position of the particles, and shares the information between particles. That is, PSO algorithm achieves the optimization through cooperation and competition between the individuals of population. Comparing with other evolutionary algorithms, PSO algorithm retains the global search strategy based on population, and belongs to the simple model of movement and velocity. PSO algorithm can dynamically adjust the current search with unique memory. Considering the currency and validity of the algorithm, PSO algorithm has been studied in many applications.
Job shop scheduling problem (JSSP) is the simplification model of an actual problem, and among the most typical and hardest combinatorial optimization problems, which is a NP

complete problem. JSSP is often used to test the performance of the intelligent algorithms, which has important research and actual engineering meanings.

2.1 Introduction of PSO

PSO algorithm simulates the prey behavior of a bird flock. We can imagine a scene, a group of birds are random searching the food, and there is only a piece of food in this region. All the birds don't know the place of food, but they know distance from the current location to the place of food. What is the optimal strategy of searching the food? The most simple and effective strategy is to search the areas where are close to the birds.

PSO algorithm is motivated from the model, and is used to solve optimization problems. Each optimization is considered as a bird in the search space called a particle. Each particle has a fitness value that is decided by an optimization function, has a velocity to determine its flight direction and distance. PSO algorithm constructs an initial particle swarm (random solutions), then find the optimal solution through iterations. In each iteration, particles update their velocities and positions by tracking the two extreme values. An optimal solution is the individual extremum *pBest* that is found by the particle itself, and another optimal solution is the global individual extremum *gBest* that is found by the current population.

In traditional PSO algorithm, the particle swarm searches results in space of m*n dimension, each particle position means a result of the problem. The particle continuously adjusts itself position X to search new results. Let P_{id} denote the optimal result that the particle obtains. Let P_{gd} denote the optimal position that the particle swarm passed, the best total result in the search domain. Let V denote the speed of the particle.

$$V_{id}(t+1)= \omega \times V_{id}(t) + \eta_1 \times rand() \times (P_{id} - X_{id}(t)) + \eta_2 \times rand() \times (P_{gd} - X_{id}(t)) \quad (1)$$

Let $V_{id}(t)$ denote the speed of d dimension of particle i in generation t, ωdenote inertia weight, and '-' denote distance. Let η_1 and η_2 denote parameter, which can adjust P_{id} and P_{gd} respectively. *rand()* is the random number generation function. Therefore, we can get the next particle position.

$$X_{id}(t+1) = X_{id}(t) + V_{id}(t+1) \quad (2)$$

Considering the formula (1) and (2), we can find that the moving direction of particle is decided by three parts. That is, the initial speed $X_{id}(t)$ of the particle, and the optimum distance $P_{id} - X_{id}(t)$ that the particle passed, and the optimum distance $P_{gd} - X_{id}(t)$ that the particle swarm passed. The relative importance of three parts is decided by weighting coefficient ω, η_1, η_2。

The traditional PSO algorithm is described as follows.

STEP 1: Construct an initial particle swarm, that is randomly set the initial position X and the initial velocity V of each particle;

STEP 2: Calculate fitness value of each particle;

STEP 3: Compare each particle fitness value and its best position fitness value P_{id}, if better, update P_{id};

STEP 4: Compare each particle best position P_{id} and the best position of particle swarm P_{gd}, if better, update P_{gd};
STEP 5: adjust the velocity and position according the formula (1) and (2);
STEP 6: If termination conditions are satisfied (good enough position or the maximum number of iterations), then end; otherwise, go to 2.
PSO algorithm is a kind of evolutionary algorithm, which has several typical characteristics. First, the individual of population has been randomly initialized a random solution in the initialization process. Secondly, the better solutions of a new generation are obtained by searching the solution space. At last, a new generation of population is produced on the basis of the previous generation.

2.2 Convergence of PSO

The convergence of intelligence optimization algorithm is an important problem for the application of intelligent optimization algorithms. It is necessary that we discuss the convergence of PSO algorithm before solving a practical problem.

2.2.1 Convergence of Traditional PSO

It is a difficult problem to prove the convergence for an intelligent optimization algorithm. Two assumptions H1 and H2 proposed by Solis and Wet were introduced, which were used to prove the global convergence of the pure optimization algorithm with probability *1*. General requirements of stochastic optimization algorithm convergence are described as follows.
An optimization problem $\langle A, f \rangle$ and stochastic optimization algorithm D are given. x_k is the results of the *k-th* iterations, and results of the next iteration is x_{k+1} ($x_{k+1} = D(x_k, \zeta)$), where ζ is the solution that has been searched by algorithm D.
Condition H1: $f(D(x, \zeta)) \le f(x)$, if $\zeta \in \boldsymbol{A}$, set $f(D(x_k, \zeta)) \le f(\zeta)$, where A is the subset of the R^n, and A denotes the constraint space of the problem.
Conditions H1 random algorithm can guarantee the correctness; their objective is to ensure optimization of the solution to the fitness value of f (x) non-incremental.
A global convergence of the algorithm, which means sequence $\{f(x_k)\}_{k=0}^{\infty}$ can reach infimum $inf(f(x) : x \in \boldsymbol{A})$ in the feasible solution A. Because it is possible that the feasible solution A of optimization problem exist discontinuity spaces or isolated spots, infimum and other fitness value is incontinuous. Considering this potential problem, search infimum is defined in Lebesgue measure space as shown in formula 3, where $v[X]$ denotes Lebesgue measure in set X.

$$\Psi = inf(t : v[x \in A \mid f(x) < t] > 0) \tag{3}$$

Formula (3) implies that non-empty set of the search space is existent, where the fitness of its members infinitely are close to Ψ. The definition of $v[X]$and A guarantee that nonempty point does not exist in set A. So the algorithm can reach or be close to the infimum without searching all points of set A.
Therefore, the optimal region can be defined as the following formula, where $\varepsilon > 0$, $M \to \infty$.

$$R_{\varepsilon,M} = \begin{cases} \{x \in A \mid f(x) < \Psi + \varepsilon\}, & \text{finite}\Psi \\ \{x \in A \mid f(x) < -\mathrm{M}\}, & \Psi = -\infty \end{cases}$$

If the optimization algorithm find a point among $R_{\tau,\ M}$, the point is an acceptable global optimal or near to the global optimum.

Condition H2: $\prod_{k=0}^{\infty}(1 - v_k[B]) = 0$, $\forall B \subseteq A$, s.t.$v[B] > 0$

Where $v_k[B]$ is probability measure in set B, and B is the *kth* iteration set of algorithm D. Algorithm D satisfies condition H2. It means, it is impossible that algorithm D searches the points among set B, and let $v[B] > 0$. Because $R_{\tau,\ M} \subseteq$ A, it is possible that the global optimum can be found.

Theorem 1 (**Global Convergence):** Supposing that f is measurable and feasible solution space A is measurable subset of R^n, algorithm D satisfies condition H1 and H2. And algorithm D generates series $\{x_k\}_{k=0}^{\infty}$. Then

$$\lim_{k \leftarrow +\infty} P[x_k \in R_{\varepsilon,M}] = 1$$

$P[\ x_k \in R_{\tau,\ M}]$ is probability measure in $R_{\tau,\ M}$, and $R_{\tau,\ M}$ is the *kth* iteration set of algorithm D. Considering theorem 1, the global convergence of stochastic algorithms must satisfy the conditions H1 and H2. Because each iteration of PSO algorithm has kept the best position, conditions H1 must be satisfied. However, utilizing Markov chain theory and mathematical theory of real variable, Dr. Van den Bergh has proved that PSO algorithm does not satisfy conditions H2.

2.2.2 Convergence of Improved PSO

Because the traditional PSO algorithm does not guarantee global convergence, the position and velocity update equations are improved for solving JSSP. Considering the formula (1) and (2), although v_k and x_k is multidimensional variable, each dimension is independent. Therefore the convergence analysis can be simplified to the one-dimensional. In order to expand the solution space of PSO algorithm, we adopt the velocity update equation and position update equation of particle i as follows:

$$v_i(t+1) = \alpha(P_i - x_i(t)) + \beta(P_g - x_i(t)) \quad (4)$$

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad (5)$$

In formula (4) and (5), α, $\beta(\alpha, \beta \in [0, 1])$ are random numbers. The part $\alpha(P_i - x_i(t))$ represents that the best private distance experience of the particle i in group t is inhabited by probability α; And the part $\beta(P_g - x_i(t))$ represents that the best group distance experience of all the particles in group t is inhabited by probability β.

It can be obtained from formula (4) and (5) that the bigger the value of α is the larger impact of P_i is, and the greater possibility of particle's moving to the local optimum will become;

Similarly, the bigger the value of β is the larger impact of P_g is and the greater possibility of particle's moving to the global optimum will become.
The particle i will stop moving when $x_i(t) = P_i = P_g$. Namely $x_i(t+1) = x_i(t)$. In order to expand the solution space of PSO algorithm, we save P_g as historical global best position and regenerate position $x_i(t+1)$ of particle i randomly in the solution space, which will make the particle i continue to search.
Through the operation, equation (5) can be deformed as follows:

$$x_i(t+1) = (1-c)\, x_i(t) + c_1 p_i + c_2 p_g \tag{6}$$

When p_i, p_g fixed, equation (6) is a simple linear difference equation, when $x_i(0) = x_{i0}$, its solution is:

$$\begin{aligned} x_i(t) &= k + (x_{i0} - k)(1-c)^t \\ k &= \frac{c_1 p_i + c_2 p_g}{c}, \quad c = c_1 + c_2 \end{aligned} \tag{7}$$

Considering the formula (7), the formula (6) has convergence if $|1-c| < 1$. That is, if $t \to \infty$, then $x_i(t) \to \frac{c_1 p_i + c_2 p_g}{c}$. If $|1-c| < 1$, then $0 < c_1 + c_2 < 2$. That is, if $0 < c_1 + c_2 < 2$, the evolution equation of improved PSO algorithm is asymptotic convergence. The convergence region shown in Fig. 1.

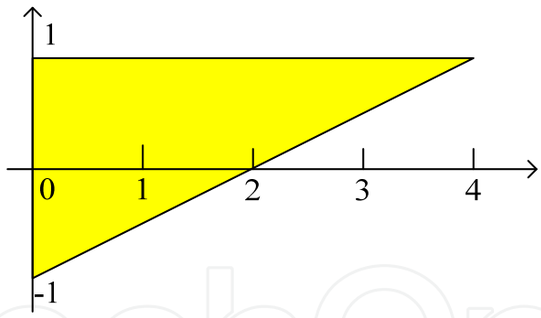

Fig. 1. Convergence region

Theorem 2: if $|1-c| < 1$, $\lim_{t\to\infty} x_i(t) = p_g$.
See the formula (7), if $|1-c| < 1$, then

$$\begin{aligned} \lim_{t\to\infty} x_i(t) &= k = \frac{c_1 p_i + c_2 p_g}{c} \\ x_i(t+1) &= x_i(t) - (c_1 + c_2)\, x_i(t) + c_1 p_i + c_2 p_g \end{aligned} \tag{8}$$

if $t \to \infty$,

$$\lim_{t\to\infty} x_i(t+1) = \lim_{t\to\infty} x_i(t) \tag{9}$$

Then

$$-(c_1+c_2)\lim_{t\to+\infty} x_i(t) + c_1 p_i + c_2 p_g = 0 \tag{10}$$

Considering theorem 1, if a random optimization algorithm satisfies condition H1 and H2, we can guarantee that the algorithm can converge to global optimal solution with probability *1*. We will discuss the problem whether the improved PSO algorithm is able to satisfy condition H1 and H2.

In the improved PSO algorithm, the solution sequence is$\{ p_{g,\ t} \}$, where t denotes evolutional generation, and $p_{g,\ t}$ denotes the best position of particle swarm in generation t. The function D is redefined by the improved PSO algorithm.

$$D(p_{g,t} x_i(t)) = \begin{cases} p_{g,t}, & f(p_{g,t}) \le f(x_i(t)) \\ x_i(t), & f(p_{g,t}) > f(x_i(t)) \end{cases} \tag{11}$$

It is easy to prove that the condition H1 is satisfied.

In order to satisfy the conditions H2, the sample space of particle swarm A must contain A. Namely,

$$A \subseteq \bigcup_{i=1}^{a} M_{i,t}, \tag{12}$$

Where $M_{i,\ t}$ is the support set of the sample space of particle i in generation t. Considering particle j, let $M_{i,\ t} = A$ when $x_j(\ t\ ) = p_i = p_g$. Let the other particle i:

$$M_{i,t} = x_i(t-1) + \varphi_1(p_i - x_i(t-1)) + \varphi_2(p_g - x_i(t-1)) \tag{13}$$

Because of $0 \le \varphi_1 \le c_1$ and $0 \le \varphi_2 \le c_2$, $M_{i,\ t}$ is a super rectangle with vertices, where $\varphi_1 = \varphi_2 = 0$, $\varphi_1 = c_1$, and $\varphi_2 = c_2$. Let $v[\ M_{i,\ t} \cap A\ ] < v(\ A\ )$, when $max(\ c_1 \mid p_i - x_i(\ t - 1\ )\ \mid,\ c_2 \mid p_g - x_i(\ t - 1\ )\ \mid\ ) < 0.5 \times \mathrm{diam}(A)$, where diam(A) denotes the length of A. Considering condition H2, the length of $M_{i,\ t}$ is near to *0* when $t \to \infty$. Therefore, the measure $v[\ M_{i,\ t}\ ]$ of each $M_{i,\ t}$ is decreasing with the growth of generation t. And the measure $v[\bigcup_{i\neq j} M_{i,t}\ ]$ of union $\bigcup_{i\neq j} M_{i,t}$ is decreasing. Therefore there is N, let $v[\ \bigcup_{i\neq j} M_{i,t} \cap A\ ] < v[\ A\ ]$ when $t > N$.

Because of $M_{i,\ t} \subseteq$ A, Let $\bigcup_{i=1}^{A} M_{i,t}$ = A. In order to satisfy the conditions H2,we define A is the Borel subset($S = M_{i,\ t}$), then $v[\ S\ ] > 0$ and $v_t[\ S\ ] = \sum_{i=0}^{a} v_{i,t}[S] = 1$. Considering theorem 1, the improved algorithm can converge to a global optimal solution with probability *1*.

There is almost identical convergence between the improved PSO algorithm and the traditional PSO algorithm. That is, the parameter $x_i(t)$ can converge to the best location within the finite range. The traditional PSO algorithm does not guarantee global convergence, but the improved PSO algorithm can converge to a global optimal solution with probability *1* when generation t is near to ∞.

2.3 Convergence of PSO

Job shop scheduling problems (JSSP) is an important part of production scheduling of an enterprise, which is one kind of the most typical and hardest combinatorial optimization problems, an NP complete problem. The main task in scheduling, in terms of production targets and constraints, is to determine the precise process route, time, machine and operation for every process object. JSSP is often used to test the performance of the intelligent algorithms, which is significant for research and actual engineering.

Particle swarm optimization (PSO) algorithm is a kind of random optimization algorithm based on continuous optimization problems. PSO algorithm is less studied to solve JSSP. The PSO algorithm design of solving JSSP is difficult, and the efficient PSO algorithm design of solving JSSP is more difficult.

Leticia etc. construct the single machine scheduling algorithm based on random coding of JSSP, and the algorithm is a kind of retardation minimum time algorithm. The algorithm utilizes the dynamic mutation operators to ensure the diversity of particle populations. The algorithm has been tested respectively with 40 jobs and 50 jobs, and the algorithm achieves good results. Lina etc. construct PSO algorithm based on operation code to solve JSSP. They apply the crossover and mutation operation of GA in place of the update operations of velocity and position of PSO algorithm.

In the hybrid particle swarm optimization, Jerald.J etc. apply GA, SA and PSO algorithm to solve scheduling problems of flexible manufacturing systems. The hybrid algorithm optimizes machine idle time and reduces the cost of production tardiness. Liu etc. combine PSO algorithm and VNS. The hybrid algorithm minimizes the makespan of the flexible JSSP. Xia etc. design the hybrid PSO algorithm based on SA local search algorithm. The hybrid algorithm can solve multi-objective flexible JSSP. In order to minimize the makespan, Sha etc. construct the hybrid algorithm based on Hash table to solve JSSP. In the hybrid algorithm, Giffler-Thompson (G&T) algorithm is adopted to construct the feasible solution from the particle location of Hash table, and SWAP operation updates the particle velocity. The hybrid algorithm combines with TS algorithm based on block structure.

2.3.1 JSSP Description

Each instance of the problem J/ $/C_{max}$ is defined by a set of jobs, a set of machines and a set of operations. Each job consists of a sequence of operations, each of which has to be performed on a given machine for a given time. A schedule is an allocation of the operations to time intervals on the machines. The problem is to find the schedule that minimizes the

makespan subject to the following constraints: (i) the precedence of operations given by each job are to be respected, (ii) each machine can perform at most one operation at a time and (iii) the operations cannot be interrupted.
Let:

- J = {1, … ,n} denote the set of jobs;
- M = {1, … ,m} denote the set of machines;
- V = {0,1, … , $\tilde{n}$ +1} denote the set of operations, where 0 and $\tilde{n}$ +1 represent the dummy start and finish operations, respectively;
- A be the set of pair of operations constrained by the precedence relations, as in (i);
- V_k be the set of operations to be performed by the machine k;
- $E_k \subset V_k \times V_k$ be the set of pair of operations to be performed on the machine k and which therefore have to be sequenced, as specified in (ii);
- p_v and t_v denote the (fixed) processing time and the (variable) start time of the operation v, respectively. The processing time of the 0 and $\tilde{n}$ +1 operations is equal to zero, i.e., $p_0 = p_{\tilde{n}+1} = 0$.

Given the above assumptions, the problem can be stated as searching minimize $t_{\tilde{n}+1}$
subject to

$$\begin{aligned} t_j - t_i &\geq p_i, && (i,j) \in A, \\ t_j - t_i \geq p_i \vee t_i - t_j &\geq p_j, && (i,j) \in E_k, k \in M, \\ t_i &\geq 0, && i \in V. \end{aligned} \tag{14}$$

The first set of constraints represents the precedence relations among the operations of the same job, whereas the second set of constraints describes the sequencing of the operations on the same machine. These constraints impose that either $t_j - t_i \geq p_i$ or $t_i - t_j \geq p_j$,. Any feasible solution of the problem (1) is called a schedule.
In this framework, it is useful to represent the job shop scheduling problem in terms of a disjunctive graph $G:=(V,A, E)$, where V is the set of nodes, A the set of ordinary arcs (conjunctive) and E the set of disjunctive arcs. The nodes of G correspond to operations, the directed arcs correspond to precedence relations, and the disjunctive arcs correspond to operations to be performed on the same machine. More precisely, $E = \bigcup_{k=1}^{m} E_k$, where E_k is the subset of disjunctive arcs is related to a machine k; each disjunctive arc of E can be considered as a pair of opposite directed arcs. The length of an arc $(i,j) \in A$ is p_i, the length of an disjunctive arc $(i,j) \in E$ is either p_i or p_j depending on its orientation. The selection of a processing order on each machine involves the orientation of the disjunctive arcs, in order to produce an acyclic directed graph. A schedule on a disjunctive graph G consists in finding a set of orientations that minimizes the length of the longest path (*critical path*) in the resulting acyclic directed graph.
According to the Adams et al. method, the graph G can be decomposed into one direct subgraph $D=(V , A)$, by deleting disjunctive arcs, and in m cliques $G_k=(V_k , E_k)$, obtained from G by deleting both the conjunctive arcs and the dummy nodes 0 and $\tilde{n}+1$. A selection S_k in E_k contains exactly one arc between each pair of opposite arcs in E_k . A selection is acyclic since it does not contain any directed cycle. Moreover, sequencing the operations on

the machine k is equivalent to choosing an acyclic selection in E_k . A complete selection S is the union of selections S_k , one for each E_k , $k \in M$. S generates the directed graph D_S =(V, $A \cup S$);S is acyclic if the associated directed graph D_S is acyclic. An acyclic complete selection S infers a schedule, i.e., a feasible solution of Problem.

In order to solve the job shop scheduling problem the best acyclic complete selection S^* that minimizes the value of the length of the longest critical path in the direct graph D_{S^*}=(V,$A \cup S^*$)must be determined.

The neighbourhood of the current solution can be formed by the solutions generated by inverting the direction of a disjunctive arc in the critical path of D_S . To this end, as stated by other authors, it is useful to decompose the critical path into a sequence of r blocks (B_1,B_2, . . . ,B_r). Each block contains the operations processed on the same machine; for each pair of consecutive blocks B_j,B_{j+1} with $1 \le j \le r$ the last operation of B_j and the first of B_{j+1} belong to the same job but are performed on different machines.

2.3.2 PSO for Solving JSSP

As for applying PSO to the job shop scheduling problem, the problem can be described as that n jobs are processed by m machines. A certain list such as $S_m = (O_i)$, $i = 1, \ldots, n$, demonstrates the list of jobs processed on a machine, then the amount of possible lists is $n!$, list set S = { S_k | $k = 1, 2, \ldots, m$} is used to express the process that n jobs done by m machines, the whole possibility of solutions is $(n!)^m$. As job shop scheduling problem, when all the operations in the solution is configured, the best processed list that satisfies the efficiency index can be seeked. Therefore, for solving job shop scheduling problem by PSO, we only need to change m encoding of each particle to seek optimal solution. According to the above, definition of operating list in job shop problem is given here.

Definition 1: exchanging operation. In the operation list, operation O_i on position i and operation O_j on position j change their positions each other. This behavior is called exchanging operation, the operator is denoted as ⊗. For the list S, the exchange of O_i and O_j is expressed as $S \otimes (O_i,O_j)$, where, (O_i,O_j) denotes the operation exchange, which can be simply expressed as $O_{i \otimes j}$. Then $S' = S \otimes (O_i,O_j) = S \otimes O_{i \otimes j}$, S' denotes the list which has been disposed.

Example 1: with regard to the job shop problem in which 6 jobs are processed on m machines, the list done on machine m is S_m =(*2 4 6 1 3 5*), for the list S_m, if operation 2 and operation 6 exchanges, their position are respectively 1 and 3, the exchange process can be described as following formula.

Here, $S'_m = S_m \otimes (O_1,O_3)$ =(2 4 6 1 3 5) ⊗ (2,6) =(6 4 2 1 3 5).

Definition 2: exchange list. The operation list composed of no less than one exchanges among operations is named as exchange list, which is denoted as CS, and $CS = (O_{1i \otimes 1j}, O_{2k \otimes 2l}, \ldots, O_{np \otimes nq})$. When the list only have one time exchange operate, $CS = (O_{1i \otimes 1j},)$, where the sequence $O_{1i \otimes 1j}$, $O_{2k \otimes 2l}$, …, $O_{np \otimes nq}$, denotes the sequence of exchanging operations in the list S.

Exchange list acts on certain fraction of S_m, and it means that all the exchange operation in the list acts on S_m one by one, namely $S'_m = S_m \otimes CS = S_m \otimes (O_{1i \otimes 1j}, O_{2k \otimes 2l}, \ldots, O_{np \otimes nq}) = [[S_m \otimes (O_{1i},O_{1j})] \otimes (O_{2k},O_{2l})] \ldots \otimes (O_{np},O_{nq})$.

Definition 3: Equal set of exchange list. Different exchange list acts on the same solution, maybe the same solution is obtained. All the exchange lists which products the same solution is called the equal set.

Definition 4: united operation of exchanged list. When more than two exchanging lists such as CS_1, CS_2, …, CS_n, which act on one list according to the sequence is named as united operate, moreover, the operator is denoted as ⊕, the unit of exchange list is expressed as HS, HS= $CS_1 \oplus CS_2 \ldots \oplus CS_n$. Through the principle stated above, it can be described as $S' = S \otimes HS = S \otimes (CS_1 \oplus CS_2 \ldots \oplus CS_n)$, where S' denotes the new operation list that S had been exchanged according to the exchange list.

Through definition 3 and definition 4, a new solution S' can be obtained after acting on solution S with CS_1 and CS_2. Supposed that there is another exchange list that acts on the solution S, if a same solution S' can be obtained, then the unite of CS_1 and CS_2 is equal to CS, which can be expressed as $CS = CS_1 \oplus CS_2$, CS and $CS_1 \oplus CS_2$ belong to the same equal set, generally speaking, CS is not sole.

Definition 5: Basic exchange list. In the equal set $\{CS_i\}$ of exchanging list, exchange list BS with least exchange operators is called basic exchange list of this equal set. Supposed X and Y are operation list on machine m, constructing an exchange list BS which satisfies $X = Y \otimes BS$, if BS is of least exchange operation, then BS is a basic exchange list, which is denoted as $BS = Y \rightarrow X$.

According to the following method, a basic exchange list can be constructed, supposed that two solutions of problem FT06 are given, the lists on machine m are respectively X and Y.

Eg: X= (1 2 3 4 5 6), Y= (2 6 3 1 5 4).

It can be seen that, in the operation X, $O_1 = 1$. and in operation Y, $O_4 = 1$, let Y's first operate exchanging $O_{1i \otimes 1}$ be $Y \otimes (O_1, O_4)$, then $Y1 = Y \otimes (O_1, O_4)$, there exists $Y1 = (1\ 6\ 3\ 2\ 5\ 4)$; in X, $O_2 = 2$, and in $Y1$, $O_4 = 2$, let the second exchange operate $O_{2k \otimes 2l}$ be $Y1 \otimes (O_2, O_4)$, then $Y2 = Y1 \otimes (O_2, O_4)$, there exists $Y2 = (1\ 2\ 3\ 6\ 5\ 4)$. Similarly, the third exchange operate $O_{3p \otimes 3q}$ is $Y2 \otimes (O_4, O_6)$, there exists $Y3 = Y2 \otimes (O_4, O_6) = X$. Spontaneously, $BS = (O_{1i \otimes 1j}, O_{2k \otimes 2l}, O_{3p \otimes 3q})$ is of the minimal exchange operations, which is named as a basic exchange list, namely, $BS = Y \rightarrow X$. Here, $BS = Y \rightarrow X = (O_{1i \otimes 1j}, O_{2k \otimes 2l}, O_{3p \otimes 3q}) = ((O_1, O_4) \oplus (O_2, O_4) \oplus (O_4, O_6))$.

Aiming at PSO used to solve job shop problem, formula of basic PSO is not fit for this new type algorithm, so the formulas are recreated as follows:

$$V_{id} = \alpha(X_{id} \rightarrow P_{id}) \oplus \beta(X_{id} \rightarrow P_{gd}) \quad (15)$$

$$X'_{id} = X_{id} \otimes V_{id} \quad (16)$$

Where α, β are random number and (α, $\beta \in [0, 1]$). $\alpha(X_{id} \rightarrow P_{id})$ expresses that all the exchange operations of basic exchange list $(X_{id} \rightarrow P_{id})$ are withheld by the probability α. Similarly, $\beta(X_{id} \rightarrow P_{gd})$ expresses that all the exchange operations of basic exchange list $(X_{id} \rightarrow P_{gd})$ are withheld by the probability β.

According to the formula (15) and (16), it can be seen that, the greater α is, the stronger P_{id} affects, the probability of moving towards to the local optimization is magnified. In the same way, the greater β is, the stronger P_{gd} effect, the probability of moving towards to the global optimization is magnified.

Due to the regularity of object functions, the optimal solution must be in the active scheduling set, so PSO uses the solution produced with G&T as the initial solution. For the random and widespread searching ability, the exchanging list based PSO is used to search globally. In the process of running PSO algorithm, if any infeasible solution appears, it must be adjusted. When there exists $P_i(t) = P_{id} = P_{gd}$ for the particle $P_i(t)$ of generation t, then recreate this particle, so that PSO algorithm for job shop problem is constructed.
The steps of solving JSSP by PSO are described as following:
Step1: Use G&T algorithm to produce an initial solution, initialize P_{id} with the initial solution, initialize P_{gd} with the best P_{id};
Step2: If the end condition is satisfied, go to Step6;
Step3: According to the position of X_{id}, calculate X_{id} 's next position X'_{id}, namely new solution;

a) $A = X_{id} \rightarrow P_{id}$ denotes that A acts on X_{id} to get P_{id} , where, A is a basic exchange list,;
b) $B = X_{id} \rightarrow P_{gd}$, where B is also a basic exchange list;
c) Calculate validity V_{id} of particle according to formula (15);
d) Calculate new position X'_{id} (solution) according to formula (16);

Step4: Adjust infeasible solution;
Step5: Calculate fitness:

a) If a better solution is got, then update P_{id};
b) If a better solution of the whole swarm is searched out, then update P_{gd}, simultaneously adopt G&T to recreate a new particle instead. Go Step2;

Step6: Show the optimal solution obtained by this algorithm (P_{gd}).
Adjustment of infeasible solution is described in hybrid PSO algorithm.

2.4 Summary

Particle swarm optimization (PSO) is an evolutionary computation technique developed by Kennedy and Eberhart in 1995. The particle swarm concept was motivated by the simulation of social behaviors. The original intent was to graphically simulate the graceful but unpredictable choreography of bird flock. In the section, we introduce search mechanisms and processes of PSO, and analyze the convergence of PSO theoretically. A new PSO algorithm is proposed based on exchanged factors and exchanged lists, which is put the PSO idea into the discrete field of JSSP.

3. Hybrid Particle Swarm optimization Algorithm for JSSP

Recently, the theorem of No Free lunch (NFL) is proposed for evaluating optimization algorithms by professor Wolpert and Macready of Stanford University. It is shown that there isn't a single solution that adapts to all problems effectively. Radcliffe and Surry have the same conclusion.
For example, if GA algorithm is better than SA algorithm when solving the problem set A, then SA algorithm must be better than GA algorithm when solving the problem set B. Considering all the circumstances, two algorithms have the same performance. Therefore, there is no kind of intelligent optimization algorithm better than the other intelligent optimization algorithms. That is, every method has its corresponding application circumstances.

In theory and practice, adopting a single intelligent algorithm is not enough for solving JSSP. The hybrid algorithm is an effective method, which enlarges the application domain and improves their performance. A hybrid algorithm combines effectively some features of several algorithms, such as optimization mechanism, process, search behavior, operation, and so on. The hybrid algorithm will have better optimization efficiency.

3.1 HSPSO

If adopting a single algorithm to solve job shop problems, it is hard to improve the local optimization after some running time of the algorithm, it is necessary to find out a method to escape from this local optimization. Therefore, a hybrid PSO algorithm based on exchanging list is proposed.

The design ideas of hybrid optimization algorithm HPSO are as follows: (1) Due to the regularity of object function, the optimal solution must be in the active scheduling set, so HPSO uses the solution produced with G&T as the initial solution. (2) For the randomly and widespread searching ability, the exchanging list based PSO is used to search globally. (3) In the process of running PSO algorithm, if an infeasible solution appears, it must be adjusted. (4) In order to avoid algorithm falling in a local optimization too early, TS exploiting strategy embedded critical operations based on exchanging neighbors is adopted to realize local parallel search, simultaneously improve the local search ability.

When there exists $P_i(t) = P_{id} = P_{gd}$ for the particle $P_i(t)$ of generation t, then adopt G&T algorithm to regenerate the particle, so that hybrid PSO algorithm for solving JSSP is constructed. The arithmetic frame is shown as Fig. 2.

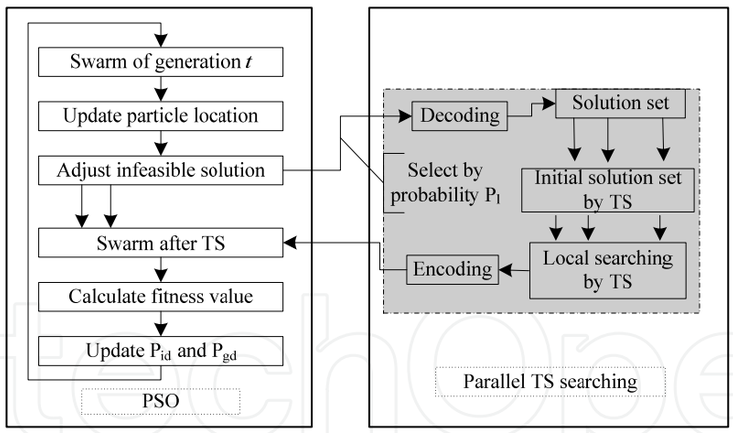

Fig. 2. Frame of the hybrid PSO algorithm

The steps of solving job shop problem by HPSO are described as following:

Step1: Use G&T algorithm to produce initial solution, initialize P_{id} with an initial solution, initialize P_{gd} with the best P_{id};

Step2: If the end condition is satisfied, go to Step7;

Step3: According to the position of X_{id}, calculate X_{id} 's next position X'_{id}, namely a new solution;

a) $A = X_{id} \rightarrow P_{id}$ denotes that A acts on X_{id} to get P_{id} , where, A is a basic exchange list,;
b) $B = X_{id} \rightarrow P_{gd}$, where B is also a basic exchange list;

c) Calculate validity V_{id} of particle according to formula (8);
d) Calculate new position X'_{id} (solution) according to formula (9);

Step4: Adjust infeasible solutions;
Step5: Select some solutions by the probability P_l to perform TS;
Step6: Calculate fitness:

a) If a better solution is gotten, then update P_{id};
b) If a better solution of the whole swarm is searched out, then update P_{gd}, simultaneously adopt G&T to recreate a new particle instead. Go Step2;

Step7: Show the optimal solution obtained by this algorithm (P_{gd}).

3.2 TS based on neighbor exchanging of critical operation

Taboo search(TS) algorithm is one of the best algorithms for solving job shop scheduling problem. So far, its running speed is faster, and it may provide a better induct within the whole searching field compared with other algorithms.

In order to obtain better searching results and higher efficiency, neighbors must be highly constrained and can be rapidly assessed. The possibility of moving to high quality solutions should be increased.

The local searching function is TS algorithm. To improve the efficiency of the local searching, we modify the TS algorithm. Firstly, the algorithm reduces the maximum that doesn't evolution. Secondly, a new exchanging strategy of neighbors is proposed based on critical operations so that TS algorithm can rapidly assess neighbors. We firstly indicate the neighbor exchanging based on the critical operation.

The feasible solution of job shop scheduling is usually denoted by the gantt graph. The gantt graph of 6×6 problem is illustrated in Fig. 3. In the figure, x-axis denotes the process time, y-axis denotes the machines, and every rectangular block marked (i, j) denotes the operation j of task i, and it is denoted as O_{ij}.

The optimized result of job shop scheduling problem is related to the length of the critical paths. The critical path is that the longest path without time intervals between operations in an available schedule. A solution always has many critical paths. For example, in Fig. 3, there are two critical paths. The first one is (4,1) (3,2) (5,1) (5,2) (4,2) (4,3) (5,3) (3,4) (3,5) (6,4) (5,5) (5,6) and another one is (4,1) (3,2) (5,1) (5,2) (4,2) (3,3) (3,4) (3,5) (6,4) (5,5) (5,6).

Furthermore, operations of the critical path can be decomposed into blocks. A block is a set of consecutive operations in a critical path in one machine. For example, operation (5,2), (4,2) and operation (4,3), (5,3), (3,4) in the first critical path forms the block respectively. Operation (5,2), (4,2), (3,3) and (3,4) in the second critical path forms the block respectively. For the two consecutive blocks, the last operation of the anterior block and the first operation of the latter block are always in the same task. For example the operation (3,3) and (3,4) of task 3 are in the same task.

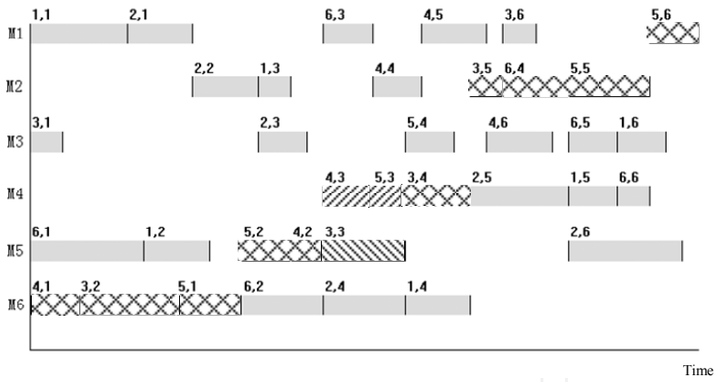

Fig. 3. 6×6 problem solution Gantt figure

Let $J_p(v)$ represent the previous operation of operation v in the same job, and $M_p(v)$ denote the previous operation of operation v processed in the same machine, $S_t(v)$ and $E_t(v)$ denote the start time and the end time of the operation v respectively.

Definition 6: Critical operation. In critical path, if operation satisfy the condition that $S_t(v)= E_t(M_p(v))=E_t(J_p(v))$, then v is called a critical operation.

When critical path is not unique, not all the neighbor exchanges can shorten the critical path. For example, Fig. 3 describes a gantt graph which shows the 6×6 problem, the exchange of (4,2) and (3,3) is unable to shorten the critical path. Because St(3,4) = MAX(Et(5,3), Et(3,3)), operation (3,4) is a critical operation, due to the dependency of critical operation (3,4) on operation (3,3), (5,3), although operation (3,3) is shortened, the neighbor exchange before the critical operation is unable to shorten the critical path.

The method of choosing neighbors based on the critical operations is as follows: when the critical path is sole, exchangeable neighbors in the critical path is considered as a set for neighbor selection; when the critical path is not sole, the exchangeable neighbors between the last critical operation and the last operation is viewed as a set for neighbor selection; TS algorithm selects an exchangeable neighbor (usually the best neighbor) from the above neighbors set to commute. If the set described above is null, then stop the current search with TS.

When TS algorithm search process runs for certain times, the quality of solution can not be improved, then TS algorithm stops.

Because of adopting new exchanging strategy of neighbors based on critical operations, TS algorithm reduces invalid neighbor exchanges, enhances searching efficiency, increases the possibility of escaping from the local optimization, and expands the searching range. Simultaneity, when there is no exchangeable neighbor, it indicates that the cost of improving the solution is too large, or the current solution is already the optimal solution. Then the searching is terminated.

3.3 HPSO Convergence

Dr. Van den Bergh has proved that PSO algorithm diverges both in the local region and the global region with the criteria presented by Solis and Wets, under which stochastic search algorithms can be considered as a global search algorithms, or merely locally search algorithms. We analyze the convergence of PSO algorithm with an optimum keeping

strategy and TS algorithm by Markov chain theory at a different aspect in this book, and HPSO algorithm based on PSO and TS algorithm is proved to be convergent. First of all, we give an introduction of Markov chain theory as follows.

Definition 7 (Markov chain) A stochastic sequence $\{X_n, n\in T\}$ and a discrete temporal series $T=\{0,1,2,\ldots\}$ are given, all state values corresponding to each X_n constitute the set of discrete state $S=\{s_0,s_1,s_2,\ldots\}$. The stochastic sequence $\{X_n, n\in T\}$ is called Markov chain as soon as the conditional probability satisfies the formula (17) as for each integer $n\in T$ and any $s_0,s_1,s_2,\ldots,s_{n+1}\in S$.

$$P\{X_{n+1}=s_{n+1} \mid X_0=s_0, X_1=s_1,\ldots,X_n=s_n\}=P\{X_{n+1}=s_{n+1} \mid X_n=s_n\} \tag{17}$$

Definition 8 (Transit ionprobability matrix) The conditional probability $p_{i,j}=P\{X_{n+1}=j \mid X_n=i\}$ is called transition probability of Markov chain $\{X_n, n\in T\}$, where $i,j \in S$. The matrix $\{P_{i,j}: i,j=1,\ldots,k\}$ is called $k\times k$ transition probability matrix.

Definition 9 (Finite homogeneous Markov chain) Markov chain is called finite homogeneous Markov chain if conditional probability $p_{i,j}(n)$ of Markov chain $\{X_n, n\in T\}$ has nothing to do with n and its set of state $S=\{s_0,s_1,s_2,\ldots s_k\}$ is finite, where $i,j \in I$. Then $p_{i,j}(n)$ is always regarded as $p_{i,j}$.

Lemma 1 Markov chain $\{X_n, n\in T\}$ with transition probability matrix P is irreducible if and only if the conditional probability satisfies formula (11) for any $s_i,s_j\in S$, where the set of state is $S=\{s_0,s_1,s_2,\ldots s_k\}$.

$$P\{X_{m+n}=s_j \mid X_m=s_i\}>0 \tag{18}$$

Lemma 2 Transition probability matrix P is irreducible if P can be turned into the form $\begin{pmatrix} C & 0 \\ R & T \end{pmatrix}$ by the same line and row transformation, where C is a strict positive irreducible stochastic matrix with dimension m, $R,T\neq 0$.Then the matrix.

$$P^{\infty}=\lim_{n\to\infty} P^n=\begin{pmatrix} C^{\infty} & 0 \\ R^{\infty} & 0 \end{pmatrix} \tag{19}$$

is a stable stochastic matrix, where $R^{\infty}=\lim_{k\to\infty}\sum_{i=0}^{k-1} T^i R C^{k-i}$, $P^{\infty}=1'p^{\infty}$, $p^{\infty}=p^0P^{\infty}$ has nothing to do with the initial distribution, and $p_i^{\infty}>0 (1\le i\le m)$, $p_i^{\infty}=0 (m\le i\le k)$.

In this book, we set the change of the group made up of social collaboration S, self adapting A and competition C three basic evolution operations, where social collaboration S means that the particle adjusts its movement by cooperating with the best position P_g of the group; the self adapting A indicates that the particle adjusts its movement at the next moment by cooperation between cognition part $\alpha(P_i - x_i(t))$ and social collaboration part $\beta(P_g - x_i(t))$; All old particles $x_i(t)$ are totally replaced by new particles $x_i(t+1)$ with optimum keeping strategy to update their self best position and group position. Therefore, the course of transformation can be presented respectively by stochastic matrix P_S, P_A and P_C, and the transition probability matrix of TS algorithm is presented by stochastic matrix P_T.

Theorem 3 The hybrid algorithm HPSO based on PSO and TS algorithm is finite homogeneous Markov chain.

Proof: Since the probability of group in next state rests with the current state, which is independent of the past state, HPSO algorithm with the set of finite state $S=\{s_0,s_1,s_2,\ldots,s_k\}$ is Markov chain. Suppose that P_S, P_A, P_C and P_T are independent of time intervals, then the searching course of HPSO can be noted by a transition probability matrix with one step $P=P_T[P_C(P_SP_A)]$, which is independent of time intervals as well. Therefore, the whole search course of HPSO is finite homogeneous Markov chain.

The design of the neighborhood is the key factor to impact on the quality and efficiency of algorithm as for the neighborhood search algorithm TS. Therefore, we first give two assumptions about the neighborhood structure as follows to ensure the convergence of TS algorithm.

Assumption 1: The neighborhood structure is supposed to be symmetrical. That is, $\forall s_i,s_j\in S, s_i\in N(s_j)\Leftrightarrow s_j\in N(s_i), i,j=0,\ldots,k$;

Assumption 2: On the point view of the graph theory, the graph G_N is supposed to be strongly connected. Namely, there must be a path from s_i to s_j for any $s_i,s_j\in S$, where $i,j=0,\ldots,k$.

Theorem 4 HPSO algorithm with the optimum keeping strategy is global asymptotic convergence when time is endless, namely the algorithm will converge to the optimal group.

Proof: Compared with the standard PSO velocity update equation, the equation has abandoned the previous velocity $\omega v_i(t)$ of particle i, which will make at least one particle of the particle swarm stop evolution of each generation due to its best history position. The optimal strategy algorithm is adopted in this hybrid algorithm. For convenience, the optimal individual reserved from each generation is saved in the left side of the population, but it does not participate in the evolutionary process. The state which contains the same optimal solution is arranged in order as same as which in the original state space, and the one which contains the different optimal solution is arranged in order according to the fitness value. Then new social collaboration transition probability matrix, self adapting transition probability matrix and competition transition probability matrix can be presented respectively as $P_S^*=diag(P_S,P_S,\ldots,P_S)$,$P_A^*=diag(P_A,P_A,\ldots,P_A)$, and $P_C^*=diag(P_C,P_C,\ldots, P_C)$. After the competition, we'll compare the optimal solution of the current population with the optimal solution reserved from the former generation, such an operation is presented by $U=(u_{ij})$. Set $Z^t=max\{f(pop_i^{t+1}),i=1,2,\ldots,N\}$ be the optimal fitness, then the transition probability from $Pop^t=[Z^{t-1},pop_1^t,pop_2^t,\ldots,pop_N^t]$ to $Pop^{t+1}=[Z^t,pop_1^{t+1},pop_2^{t+1},\ldots,pop_N^{t+1}]$ is presented as follows:

$$P(Pop^{t+1}=s_j \mid Pop^t=s_i)=\begin{cases}1 & \text{if } \max\{f(pop_i^t), i=1,2,\cdots,N\}=f(Z^t) \\ & \text{and } (pop_1^t,pop_2^t,\cdots,pop_N^t)=(pop_1^{t+1},pop_2^{t+1},\cdots,pop_N^{t+1}) \\ 0 & \text{otherwise}\end{cases} \tag{20}$$

Thus, there is unique element 1 in every line of the U, the others are 0. Meanwhile, U is lower triangular matrix considering that the individual or is replaced by better or remains unchanged. Therefore, U is noted as follows:

$$U=\begin{pmatrix} U_{11} & & & \\ U_{21} & U_{22} & & \\ \vdots & \vdots & \ddots & \\ U_{k1} & U_{k2} & \cdots & U_{kk} \end{pmatrix} \tag{21}$$

Where U_{ij} is $k\times k$ matrix, and U_{11} is unit matrix. That is to say that the transition probability matrix with one step of PSO algorithm is lower triangular matrix.

$$\begin{aligned} P^* &= P_C^* (P_S^* P_A^*)U \\ &= diag(P_C(P_SP_A), P_C(P_SP_A),\ldots, P_C(P_SP_A))U \\ &= \begin{pmatrix} P_C(P_SP_A)U_{11} & & & \\ P_C(P_SP_A)U_{21} & P_C(P_SP_A)U_{22} & & \\ \vdots & \vdots & \ddots & \\ P_C(P_SP_A)U_{k1} & P_C(P_SP_A)U_{k2} & \cdots & P_C(P_SP_A)U_{kk} \end{pmatrix} \\ &= \begin{pmatrix} C & 0 \\ R & T \end{pmatrix} \end{aligned} \tag{22}$$

Obviously, P^* is an irreducible stochastic matrix.

In theory, it has been proved that if the search space S of TS is limited, and neighborhood structure satisfies the above assumption 1 and assumption 2, TS algorithm will converge to optimal solutions inevitably. Then the transition probability matrix with one step of TS algorithm is irreducible stochastic matrix as well. Apparently, the transition probability matrix of HPSO algorithm $P=P_TP^*$ is irreducible stochastic matrix. This shows that the probability of individual staying in the non-global optimal solution tends to 0, therefore HPSO algorithm with the optimum keeping strategy will converge to the optimal group when time is endless. Namely, $\lim_{t\to\infty} P(Z_t \in S_{opt})=1$, where S_{opt} is the optimal solution set.

3.4 Experiments and Analysis

According to the above analysis, the global asymptotic convergence of HPSO algorithm can be guaranteed theoretically. However, the proof is based on perfect operation situations such as sufficiently large taboo list, infinite time and so on. Considering the reality of computer limitations and the limited time, we just take the convergence theory as the guidance in the specific computational experiments, some relaxations are made in accordance with the actual conditions on aspects of taboo length, search steps. Therefore, the solutions of some problems we obtained can just go nearly to rather than reach the optimal solution.

In this experiment, we apply the HPSO algorithm to 13 typical benchmark job-shop scheduling problems including FT10, LA02, LA21, LA24, LA25, LA27, LA29, LA36, LA37, LA38, LA39 and LA40. The experimental results are shown in Table 1.

In PSO algorithm, the swarm size is set to |O|*200%, where |O| is the number of operations, and maximum of iterative generations is set to |O|*200%; In HPSO algorithm, the swarm size is set to |O|*70% and maximum of iterative generations is set to |O|*70%. We choose the unfeasible solution of PSO algorithm by probability P_l (P_l=20%) as the initial solution of TS algorithm, after 50 search steps, algorithm will end if it couldn't find a better solution in TS algorithm. The swarm size of PGA is set to |O|*70% and maximum of iterative generations is set to |O|*70%, where the crossover probability is 0.85, the mutation probability is 0.05.

Problem	Optimum Makespan	PGA			PSO			HPSO		
		Optimum	*Average value*	*time/S*	*Optimum*	*Average value*	*time/S*	*Optimum*	*Average value*	*time/S*
FT10(10×10)	**930**	943	963.0	60.09	977	996.9	24.88	**930**	945.2	37.49
LA02(10×5)	**655**	**655**	682.4	14.43	702	734.2	3.71	**655**	668.2	5.05
LA19(10×10)	**842**	**842**	842.0	54.38	874	884.0	10.22	**842**	842.6	26.21
LA21(15×10)	**1046**	1058	1068.0	171.18	1254	1281.6	28.50	1078	1099.0	200.77
LA24(15×10)	**935**	945	949.0	165.86	1130	1149.3	27.81	947	959.4	218.16
LA25(15×10)	**977**	1020	1026.5	177.18	1174	1197.0	30.43	999	1018.5	217.70
LA27(20×10)	**1235**	1442	1464.9	577.59	1502	1530.1	457.35	1257	1267.4	558.50
LA29(20×10)	**1153**	1305	1330.7	569.55	1439	1488.3	500.75	1198	1214.6	512.1
LA36(15×15)	**1268**	1318	1326.3	687.78	1338	1356.8	758.5	**1268**	1283.3	599.0
LA37(15×15)	**1397**	1436	1441.1	790.57	1503	1519.2	714.01	1415	1425.8	817.2
LA38(15×15)	**1196**	1242	1251.0	731.47	1262	1294.3	708.33	1208	1217.5	723.3
LA39(15×15)	**1233**	1244	1247.3	720.53	1306	1320.3	709.20	1244	1246.4	614.0
LA40(15×15)	**1222**	1243	1286.4	855.50	1284	1299.8	738.17	1224	1233.1	766.26

Note:The bold letters are optimum.

Table 1. The Average Value of Ten Times Experiments And Optimal Values

The algorithm for JSSP mentioned above can be easily implemented on computer. We program the algorithm in C and run it on CPU AMD2800+with 1G.Table 1 shows that we can find the optimums of problem FT10, LA02, LA19 and LA36 when we apply HPSO algorithm to solve the 13 benchmark problems. We can obtain that there are 10 average value of ten times experiments of HPSO algorithm better than PGA algorithm, and the deviation between the average value of ten times experiments of HPSO algorithm and the optimum is lower than which between PSO algorithm and the optimum by 11.69% . Thus the overall search capability of the algorithm is improved, which make the algorithm get closer to the optimum solution.

We analyze the convergence of PSO algorithm with optimum keeping strategy and TS algorithm by Markov chain theory as for the Job Shop problem, and present a hybrid algorithm called HPSO algorithm with global asymptotic convergence based on the above

convergence theory. This algorithm has made full use of the large scale random search capability and the social cooperation of PSO algorithm, at the same time, the local parallel TS algorithm is embedded to improve the local search capability. We apply the above convergence theory to computational experiment and find the optimum of problem FT10, LA02 and LA19 in a short period. When compared with PGA algorithm and PSO algorithm, there are 10 average value in ten times experiments of HPSO algorithm better than PGA algorithm, and the deviation between the average value of ten times experiments of HPSO algorithm and the optimum is lower than that of between PSO algorithm and the optimum by 11.69%. Thus the overall searching capability of the algorithm is improved, which has demonstrated the effectiveness of solving Job Shop Scheduling problem by HPSO algorithm.

3.5 Summary

The theorem of No Free lunch (NFL) shows that there isn't a single solution that adapts to all problems effectively. Therefore, a hybrid algorithm of particle swarm optimization and tabu search algorithm (TS) for solving JSSP is proposed motivated by the strong global search capability of PSO algorithm and the good local search capability of TS algorithm. Meanwhile, the convergence of HPSO is proved, and experimental simulation results are given.

4. Strategies for Deadlock Elimination for JSSP using PSO

Deadlock is a state that the requests of scheduling transactions which contest resources one another can not be satisfied. Namely, the stagnancy is among transactions waiting for one another appears. When using a PSO algorithm to solve constrained optimization problems, deadlock is one of the key problems need to be solved. In PSO algorithm, we optimize the various operations of jobs based on PSO code. Because the different operations of the same job are studied as the separate object. Different objects can be in different machine queues. So this may be a single legitimate machine queue (two jobs does not occupy the same machine at the same time). Meanwhile, deadlock is latent among the queues on different machines. Efficiency and feasibility are decided by various conditions when using PSO to solve JSSP. Deadlock is a kind of the most important link. The strategies for eliminating deadlock are proposed in the book.

4.1 Deadlock Problem of JSSP

We find that the machines as resources are preempted by the jobs in the production process, and the rings that jobs are waiting for one another are created. The state is called as deadlock. Each job is waiting other resources occupied by another job. The utilization rate of the system will decline. If the particle deadlock can not be solved in time, the whole production system will collapse, and automate production will be unable to continue.

The computer scientist first put forward deadlock when dealing with the allocation of resources in the operating system. Coffman has given four necessary conditions for deadlocks as follows:

(1) Exclusion. Resources only can be allocated to a particular task or an idle task. Resources can not be occupied simultaneously by two tasks.

(2) Non-preemption. Resources is non-preemptive. When the corresponding process task is completed, the task will release occupied resources.
(3) Occupation and waiting. The task has been occupying some resources. Meanwhile, the task requests additional resources that have been occupied by other task.
(4) Circular and waiting. There exists a set of requested resources {P1,P2,…,Pn}. Where P1 is waiting for resources occupied by P2, P2 is waiting for resources occupied by P3, …, Pn is waiting for resources occupied by P1.
In the shop scheduling, there exists that the requests of scheduling transactions which contest resources one another can not be satisfied, so a state of stagnancy among transactions waiting for one another appears, resulting in deadlock and emergence of infeasible solutions. When using a hybrid PSO algorithm to solve JSSP, deadlock is one of the key problems need to be solved. To obtain valid hybrid PSO algorithm for JSSP, we study the reasons producing deadlocks in PSO algorithm, and present three countermeasures for deadlock elimination: encoding elimination, detection and reconstruction, and direct reconstruction.
While solving JSSP using a hybrid PSO, we firstly transfer JSSP into encoding denotation based on operations as the result of scheduling. Different operations of identical jobs may be processed on different machines, which can result in deadlock among job queues on different machines, being infeasible solution. In this solution space corresponding to encoding based on operation, it not only contains feasible solutions, but also contains infeasible solutions (namely, solutions with deadlock). For example with a 6×6 shop scheduling problem, Fig. 4 denotes one possible scheduling gantt chart, and the vertical coordinate denotes serial number of processing machine.

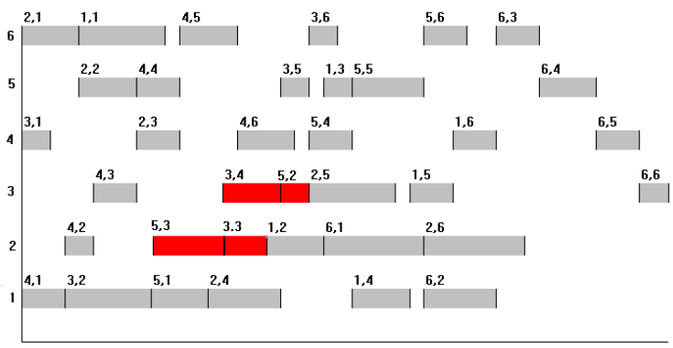

Fig. 4. A possible scheduling gantt of a typical (6×6 scheduling problem)

We can see from Fig. 4 that job queue on both machine 2 and machine 3 have deadlock, namely, operation (5,3)→(3,3) and (3,4)→(5,2) just a typical waiting deadlock. Because before operation (5,3) is processed, it must wait until operation (5,2) has be finished, while operation (3,4) before operation (5,2) must wait until operation (3,3) has be finished, but operation (3,3) is after (5,3), namely, this scheduling has deadlock. Occurrence of deadlock makes particles generated by G&T in hybrid PSO could not go on evolution, because the algorithm simulates course of processing according to scheduling scheme under the above dual restrictions, while valuing individuals, fitness values always are computed according to job's last makespan in processing system, the makespan of the last operation is just circulation ending time of the whole batch of jobs. Deadlock makes the process stagnated at

the position of deadlock and could not go ahead, so we could not gain this operation's maximal makespan in a common sense, which forms infeasible scheduling solution in solution space. So, while effectively solving JSSP using PSO or hybrid PSO, the deadlock matter is an obstacle which we must solve.

4.2 Deadlock Elimination and Reconstruction

4.2.1 Encoding Elimination

In deadlock elimination strategy, we design an encoding denotation based on operations as the result of scheduling. Then we utilize the encoding and decoding to eliminate deadlock (that is, the infeasible solution).

During the optimizing process, m segments of each particle are changed. If we produce solutions by encoding m segments of a particle directly, it is possible to produce some infeasible solutions, which is also called dead lock, and it will lead to bad optimization efficiency. Aiming at an $n \times m$ job shop scheduling problem in which the chromosome is made up of $n \times m$ genes, when the operation position of every machine is changed, the position of operations in the chromosome corresponding to the position of machines is changed, so a feasible solution is obtained, and infeasible solution is avoided, either the characteristic of PSO algorithm is reserved.

The encoding process based on operations is: for the problem that n jobs need to be processed on m machine. The chromosome is an $n \times m$ array that denotes all operations. For a matter of convenience, let m=3, n=3, and chromosome starts with such a segment of genes:[*2, 1, 3, 2, 1, 2, 1, 3, 3*]. We assign the same symbol for the same job's operation, where *1* denotes job J_1, *2* denotes job J_2, and *3* denotes job J_3. Because each job has three processes, each job appears three times in chromosome. The operation of job is equivalent to the order in chromosome.

The decoding process based on the encoding operation is: firstly, the chromosome is transformed to an orderly list that denotes an order of the production process. Secondly, according to the processing order of each operation, scheduling scheme is given. The scheme includes the start time and end time of every job.

For above 6×6 problem, in particle swarm chromosome encoding, there are six dimensions, each of which has a position value. Here, the position value is composed of six segments of chromosomes divided by machine devices, and chromosome's encoding method is an encoding composed of a whole segment chromosome (36 bits genes on it), where each gene which represents a job index is a decimal number. According to its size order, distribute operations over again for the whole segment of chromosome's genes. So, particle's position shift is not restricted to operation shift on a single machine, but within the whole segment of chromosome.

job	Machine sequence		
	operation1	Operation2	operation3
J1	1	2	3
J2	1	2	3
J3	2	1	3

Table 2. 3×3 Job Shop scheduling problem

See table 2, suppose that the chromosome of a 3×3 Job Shop scheduling problem is [***2 1*** 1 3 1 2 ***3*** 3 2], then the process sequence of each machine is as follows: M_1 [**2 1 3**], M_2 [**1 3 2**], M_3 [**1 2 3**]. If optimizing it with PSO, the sequence of M_1 is changed into [**3 2 1**], then the solution become infeasible, because the operations on each machine are all not the first operation. However, by recoding and decoding the chromosome again, new feasible solution can be converted. The new chromosome encoding is [***3 2*** 1 3 1 2 ***1*** 3 2], and the new sequence of each machine by decoding is: M_1[2 1 3], M_2[3 1 2], M_3[2 1 3].
Based on such encoding elimination method for deadlock, it makes this algorithm simplified. Each time dealing with deadlock, we only need to take out the first operation in the waiting schedule, and distribute it to corresponding machine according to the rule of earliest finish time; we take out all bits in chromosome and dispose them, and then we obtain a feasible scheduling scheme, it is not necessary to detect deadlock and tackle fitness value of deadlock scheduling. Although the algorithm is simple, it has good searching capability because the decoding process can create an active scheduling.

4.2.2 Detection and Reconstruction

In process of PSO's colony evolution, it can easily bethink of decline strategy, namely, abandon infeasible solution brought by deadlock while only feasible solution is preserved, in which way we could not have to consider the infeasible solutions. This method is available to problems which have weak restrictions, since for weak restrictions feasible solutions have larger proportion in searching solution space; however, this method could still find some good solution from searching space. But, as to JSSP model which is a sort of problem with strong restriction, according to the encoding method in chart 1, feasible solutions have a little ratio in searching space, the complexity of searching for feasible solution is not inferior to the original problem (as to 10×10 problem's first-generation population, the author generates scheduling by particle evolution, in which, the proportion of feasible scheduling is less than 3%, see table 3). It is obvious that, with above encoding method, as to JSSP, only considering feasible solution is not enough.

	1	2	3	4	5	6	7	8	9	10
0	0.98	0.98	0.81	0.86	0.75	0.79	0.78	0.79	0.75	0.68
10	0.65	0.66	0.65	0.63	0.58	0.59	0.55	0.55	0.51	0.47
20	0.46	0.44	0.47	0.48	0.48	0.55	0.46	0.44	0.43	0.48
30	0.46	0.49	0.42	0.49	0.46	0.41	0.38	0.41	0.43	0.37
40	0.70	0.69	0.71	0.73	0.71	0.71	0.66	0.68	0.60	0.63
50	0.69	0.74	0.69	0.65	0.69	0.67	0.74	0.76	0.70	0.75
60	0.71	0.76	0.65	0.74	0.71	0.66	0.66	0.65	0.63	0.65

Table 3. The proportions of deadlock particles to the total particles in seventy generations of FT10

Note that, one characteristic of this sort of deadlock scheduling, namely, is high quantity and can hardly be found one by one, but its existence can be easily detected. Here, we provide design for deadlock elimination and reconstruction, that is, in routine G&T algorithm "randomly selects an operation in clash set", is modified as "according to the

operations' sequence in deadlock's scheduling generated by PSO algorithm, select operations in clash set *G*". Consequently, it makes the scheduling being deadlock restored, making the ratio of feasible solution ascend gradually in process of evolution and selection.
As to JSSP, the detecting times of deadlock is denoted as "T", the worst case is that we examine circularly each operation queue once, while we only need to examine m×n times, if there is no deadlock in scheduling, T is m×n times. In process of detection, none but deadlock can make the algorithm process stagnate, that is, the algorithm is blocked by deadlock, in which case we can eliminate deadlock by rebuilding clash set using G&T. In the clash set, according to the order of job index of operation on machine in scheduling having deadlock, select operations, thereby, the scheduling having deadlock gets restoration.

4.2.3 Direct Reconstruction

From above analysis for 10×10 problem's deadlock, we can see, in the encoding based on operation, deadlock is in a great deal. In practice, among JSSP's strong restriction problems, deadlock scheduling is in a great proportion in the whole solution space, the effect of the way detecting deadlock is not necessary very good. Because detection need a lot of cost, the worst case of deadlock detection is the same as JSSP, it also a combinatorial blast problem, and deadlock's concrete information is skimble-skamble for JSSP. So, we design a direct reconstruction which directly rebuilds clash set using G&T. In the clash set, according to the order of job index of operation on machine in scheduling having deadlock, we select operations, namely, no necessary detect deadlock for evolving particle and directly reconstruct new solution, which makes the ratio of better solution ascend gradually, finally making the swarm go ahead towards optimization. This method avoids deadlock detection, comparatively, time performance may be simplified.
Seen from the above, the direct reconstruction is different from deadlock elimination reconstruction in that we do not judge whether there is a deadlock, but directly rebuild clash set using G&T. In the clash set, according to the order of job index of operation on machine in scheduling having deadlock, we select operations, thereby, the scheduling having deadlock gets restoration.

4.3 Experiments and Analysis

The experiment in this section aims at JSSP, solving 13 typical benchmarks hard problems, such as FT10, LA02, LA19, LA21, LA24, LA25, LA27, LA29, LA36, LA37, LA38, LA39, LA40.
In hybrid PSO (HPSO) using deadlock elimination strategy, the population size of PSO is set as |O|*70%, where |O| is the total number of operation; every evolution generation number is |O|*70%; we select |O|*P_l particles from particle swarm and perform TS search, if though TS's search process has passed half of total operation number we still can not obtain better solution, we end TS process.
The experiment in this section is implemented with C code, the experiment environment is: CPU with Pentium-4 2.4G, and memory with 512M. HPSO1, HPSO2, and HPSO3 are hybrid PSO respectively using encoding elimination, detection and reconstruction, and direct reconstruction. The average fitness values of ten times searching solution of HPSO1, HPSO2, and HPSO3 respectively are 2.66%, 3.13% and 2.62%.
Seen from the experiment results, the effect of hybrid PSO using encoding elimination is worse than that using direct reconstruction, since that, with encoding elimination, the

scheduling schemes generated by particle whose position has changed have many repeated scheduling schemes, namely, there are many particles which have the same position in the particle swarm, and most of them are non-active scheduling, which reduces the diversity of particles, and conflicts with PSO algorithm whose original intention aims at enhancing the diversity of particles, although the running time of the algorithm is short, the searching performance is bad comparatively.

The difference between deadlock elimination reconstruction and directly reconstruction is that, the former detects and judges whether the solution has deadlock, if the solution has deadlock, reconstruct it; the latter doesn't detect solution, but directly reconstructs solution. In deadlock elimination and reconstruction, we total up all infeasible particles in every generation, namely, the proportion of particle being deadlock to the total particle number in the swarm. And we have totaled for the typical problems (FT10, LA02), from which we can see the difference of the two methods.

	1	2	3	4	5	6	7	8	9	10
0	0.91	0.83	0.89	0.88	0.77	0.75	0.63	0.60	0.56	0.52
10	0.48	0.40	0.40	0.27	0.26	0.32	0.18	0.28	0.30	0.28
20	0.28	0.19	0.17	0.17	0.15	0.15	0.20	0.13	0.16	0.16
30	0.19	0.22	0.05	0.07	0.03	--	--	--	--	--

Table 4. The proportions of deadlock particles to the total particles in thirty five generations of LA02

We can see from the experiment results, as to JSSP in which the number of job and machine is more or less equal, deadlock elimination reconstruction and direct reconstruction are all square in their search time and search results. As to JSSP in which the number of job is more than the number of machine, it is better employing hybrid PSO using direct reconstruction. Because in JSSP in which the number of job is more than the number of machine, the operation number of each job is oppositely small, thereby, the number of infeasible solution after particle evolution is oppositely small, which can be seen from the statistic result in table 4, since the number of the feasible solution is oppositely large, the detection algorithm runs and does not know the solution is feasible solution until the algorithm runs out, so the algorithm's running time is oppositely long, and after particle's evolution, the number of particles whose fitness value are smaller than the fitness value of the primary particle is oppositely large, which is a disadvantage for obtaining global optimal solution, but as to JSSP in which the number of job and machine is more or less equal, the number of the infeasible solution after particle evolution is oppositely large, for which we can conclude from the statistic result in table 3 that detection algorithm can drop midway, which can reduce the computation time, at the same time, among particles obtained by improved G&T repair algorithm, the number of particles whose fitness value after evolution is smaller than the fitness value before evolution is oppositely small, which makes for the search for optimal solution. The experiment results also indicate that, the hybrid PSO using direct reconstruction has better effect and advantage for solving JSSP.

In seventy generations of FT10 and thirty five generations of LA02, the proportions of deadlock particles to the total particles are shown in table 3 and table 4, in which the proportion is the ratio of deadlock scheduling to the total scheduling in the particle swarm in every generation. The population size of PSO is set as $|O|*70\%$, where $|O|$ is the total

number of operation; every evolution generation number are |O|*70%, the total number of operation is m×n, namely, (10×10=100) the generation number is 70.

Benchmarks problem	Optimal Makespan	HPSO1			HPSO2			HPSO3		
		Optimal Value	Average Value	Time (sec)	Optimal Value	Average Value	Time (sec)	Optimal Value	Average Value	Time (sec)
FT10(10×10)	**930**	934	939.2	44.83	**930**	936.7	36.83	**930**	937.5	37.69
LA02(10×5)	**655**	**655**	667.6	1.41	**655**	688.4	7.275	**655**	656.0	4.22
LA19(10×10)	**842**	850	883.4	19.97	**842**	849.9	14.604	**842**	845.5	13.87
LA21(15×10)	**1046**	1055	1067.2	211.66	1078	1088.8	255.09	1050	1080.9	244.25
LA24(15×10)	**935**	954	959.3	228.62	950	958.2	258.69	944	950.4	250.91
LA25(15×10)	**977**	986	991.6	214.48	989	992.6	249.28	**977**	983.4	250.24
LA27(20×10)	**1235**	1265	1277	204.66	1269	1302.9	288.13	1260	1288.9	258.35
LA29(20×10)	**1153**	1203	1210.8	257.08	1253	1273.1	280.78	1200	1227.8	289.42
LA36(15×15)	**1268**	1292	1300	288.23	1274	1283.1	301.90	1280	1296.4	290.85
LA37(15×15)	**1397**	1433	1448.3	310.52	1415	1439.1	359.83	1415	1435.8	360.72
LA38(15×15)	**1196**	1209	1220	340.71	1212	1221.0	343.31	1204	1255.8	350.81
LA39(15×15)	**1233**	1248	1264.2	350.81	**1233**	1260.0	340.30	**1233**	1260.5	364.73
LA40(15×15)	**1222**	1230	1234.6	364.21	1229	1236.2	360.77	1229	1238.6	385.62

Note: boldface is the optimization.

Table 5. The optimal value and average values in ten times' experiments

4.4 Summary

To obtain valid hybrid PSO algorithm for JSSP, the reasons producing deadlocks in hybrid PSO algorithm is studied, and three strategies for eliminating deadlocks are proposed. When solving highly constrained combinatorial optimization problems, deadlock is one of the key problems need to be solved. The experiment results show that HPSO algorithm is a kind of feasible and effective method for sloving JSSP. With contrast experiments on 13 hard benchmark problems, both the results of deadlock detection and optimization objective results show that direct reconstruction is more effective, and better than the other two methods in searching quality. The hybrid PSO using direct reconstruction for deadlock problem has more advantages comparatively. The deadlock elimination algorithm, namely, the hybrid PSO for JSSP given in this book, has improved the solution quality of the hybrid PSO for JSSP, and which has provided a feasible and effective method for solving deadlock problem in PSO.

5. Other Hybrid PSO Algorithms for JSSP

As the diversity of optimization, the research experience and preferences of researchers often determine the selection of algorithms. The application of algorithms is various, and the new optimal alogorihm is difficlut proposed based on the natural mechanism. Therefore, the hybrid algorithm is an important and effective way to improve algorithm. In the section, we introduce other hybrid PSO algorithm for solving JSSP, including hybrid algorithm of PSO and simulated annealing (SA) algorithm, hybrid algorithm of PSO and genetic algorithm (GA).

5.1 Hybrid Algorithm of PSO and SA

5.1.1 Hybrid Algorithm Design

Simulated annealing algorithm (SA) is a kind of stochastic search algorithm based on Monte Carlo iteration search strategy. SA algorithm has the jumping ability and strong universality, and it is easy to be realized. However, the running time of SA is long and its efficiency is low. PSO algorithm has strong universality. However, PSO has the disadvantages of prematurity and the tendency of falling into the local optimization. Therefore, a hybrid algorithm based on PSO and SA algorithm (HPSOSA) is proposed for improving the overall quality of the optimization algorithm.

According to the characteristics of random and large-scale search of PSO, we adopt PSO to construct a group of the initial solutions with good quality and dispersion. At the same time, each particle pursues the process of parallel search of SA in the population P_l. SA algorithm not only is a supplement of PSO and beneficial for the local improvement, but also has the probabilistic jumping ability of escaping the local optimization.

5.1.2 Enhanced Simulated Annealing

In theory, SA algorithm can search the global optimal solution with probability 1 only if the parameters of algorithm satisfy the convergence conditions. However, it is impossible that some convergence conditions be met strictly according to SA algorithm convergence theory. The selection of SA algorithm parameters is still a problem. In the book, SA algorithm is used to local search of the hybrid algorithm, and we improve the process and sampling of SA algorithm.

(1) The operational pattern at a single comparison of traditional SA algorithm requests the fully high initial temperature and slow temperature drop. SA algorithm optimizes the part solution of the population each generation of PSO algorithm, and each random searching selects a value from a range as the initial temperature.

(2) The sampling process of traditional SA algorithm requests that sampling time at each temperature is long enough, and the temperature tends to *0* eventually. When the temperature remains unchanged in continuous n steps at the current state, we think that the Metropolis sampling is stable. Then, SA algorithm will terminate calculation in the temperature. If the optimal solution remains unchanged in continuous n steps at cooling process, we think that the algorithm is convergent.

The parameters of SA algorithm:
(1) Initial temperature t_0
The higher the initial temperature is, the larger rate of quality solution will be obtained. However, the cost of calculation will increase too. Therefore, we should certainly tradeoffs quality and efficiency when choosing the initial temperature. Before the local search in ESA, we make sure of the biggest difference between two targets ($|\Delta max|$) of PSO. Then according to the difference, the initial temperature is set by function $t_0=-\Delta max/lnp_r$ (p_r is initial speedups at convergence). If p_r is close to *1*, and the initial random status can express the whole status space, the algorithm will accept all status almost in same probability, will not accept the restriction of the smallest solution completely.
(2) cooling rateω ($0<\omega<1$)
The more ω is close to *1*, it shows that the slower the cooling rate decreases, and vice versa. Moreover, the algorithm has different search depths in different cooling rates. Therefore, SA algorithm uses the strategy of the variable cooling rate for improving the randomness of search by the random changing ω values in the search process.
(3) Iterative times L
The iterative times of each temperature is fixed value. When the temperature is high, the algorithm will accept all status almost in same probability, and the iterative times can reduce. When the temperature is gradually decreasing, the algorithm will reject most of status almost in same probability. If the iterative times reduce, the objective function will converge to a local optimization prematurely. In the enhanced algorithm, SA has different iterative times in different temperature. When the temperature is decreased, the iterative times of the same temperature will increase.

5.2 HPSOSA Algorithm

Considering normality of the objective function, the optimal solution is situated in an active schedule. In HPSOSA algorithm, Giffler-Thompson (G&T) algorithm is adopted to construct the initial solution of PSO, which has the ability of random and large-scale search. According to the convergence of PSO, each particle pursues the process of SA parallel searching in the degressive population P_l. Then, the solutions which aren't selected and new solutions by SA conduct the search solutions of PSO in the current generation. Adopt G&T algorithm to regenerate the particle, when $X_{id}(t)=P_{id}=P_{gd}$ in generation t.
In HPSOSA algorithm, we not only utilize parallel SA algorithm improving the search area, but also use PSO algorithm ensuring the convergence. The hybrid algorithm give attention to the accuracy and efficiency of optimization.

HPSOSA algorithm is described as:

```
Begin
PSn ← DetermineSizeOfParticleSwarm()
P_gd ← NULL; P_id ← NULL; P_l ← 0.2;
ConstructionInitializeParticleSwarm ( PSn )
while termination conditions not satisfied do
    CalculationParticleValidity( V_id )
    CalculationParticleNewPosition( X_id )
    if ( rand( 0, 1 ) < P_l ) then
            ESASearch( )
    else
        ApplyLocalSearch( )
    end if
    CalculateParticleFitnessValue( )
    Update( P_id )
    Update( P_gd )
    if ( P_id== P_gd ) then
      GenerateNewParticle( i )          //G&T algorithm
    end if
end while
Output P_gd
End
```

5.3 Hybrid Algorithm of PSO and GA

5.3.1 Hybrid Algorithm Design

Genetic algorithm (GA) is a kind of algorithm for solving JSSP, and the algorithm has the ability of large-scale and global-convergence search. However, after the evolution to a certain evolutional generations, the objective function value (satisfaction) almost does not change, falling into the local optimization. If there is no external interference, it is difficult to leave the local optimum. Meanwhile, if external intervention can break the balance, the probability of searching the optimal solution will increase.

PSO algorithm is a kind of search algorithm based on iteration search strategy, and PSO algorithm has the characteristics of global optimization. In the last few years, along with application study is further, PSO has expanded its application to solve JSSP. It is found that PSO algorithm is good in the early evolution for solving JSSP. However, particle populations will quickly lose diversity, and PSO algorithm will cause premature convergence or slow the global convergence.

We should choose the algorithms with different characteristics, make them integrate mutually, give full play to their advantages, and generate the better efficiency of optimization. It is undoubted that it is an effective way to solve JSSP. Therefore, the hybrid parallel GA and PSO (HPSOGA) algorithm is established for solving JSSP. Considering the difference of the search mechanism and characteristics of PSO and GA algorithm, the parallel asynchronous hybrid method is adopted as the hybrid method. There are the global search in two populations individually, using the principle of different search algorithms.

Simultaneity, migration Operator is adopted to achieve the intercommunication between PSO and GA algorithm. When the optimization algorithm of one population falls into the local optimization, another algorithm disturbs the first population. Through exchanging individuals in the evolutionary process, the search area and accuracy are improved.
The HPSOGA algorithm is established for solving JSSP, its parallel hybrid model is illustrated in Fig. 5.

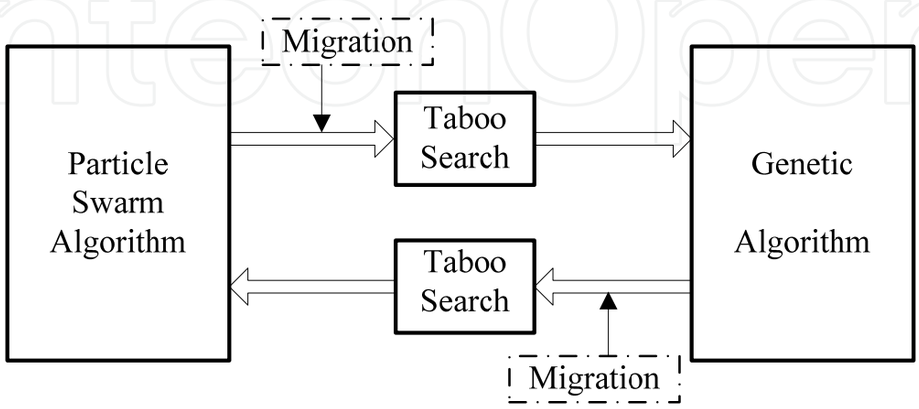

Fig. 5. HPSOGA parallel hybrid model with TS

5.3.2 Migration Operator Design

The key cycle of the hybrid algorithm is the designing of communication medium, which can make different algorithms exchange their information with each other so as to cause populations to share high quality seed. Thereby, introduce migration operation, which not only improves the diversity of GA population and enlarges the search scope of solution space, but also enhances the convergence of PSO population and deals with particles evolution halt problem. As two different algorithms are employed parallel hybrid to search optimization solution, it is too unilateral that only a single migration strategy is adopted. According to different specificity of both algorithms, two migration strategies related with two different migration occasions are designed, corresponding a migration strategy in each migration occasion is employed.
(1) Considering the fact that a particle stops evolution in the process of PSO evolution, if the particle which stops evolution is detected out, a solution randomly from GA population is selected, and it is diverted to PSO and replaced with the particle which stops evolution; if there are several particles which stop evolving in PSO population, the other GA solutions which have different fitness value are selected and diverted to PSO, and the corresponding particles which stops evolution are replaced; the particle which stops evolution is migrated to GA population and replaced with the individual whose fitness value is lowest in GA population.
(2) According to the fact that GA is prone to convergence, in the process of GA population evolution, the convergence factor *cf* is detected every certain generation in this algorithm. When a convergence factor is smaller than the preset value, select several good solutions whose fitness value are different from PSO population, and divert them to GA population so as to disturb GA evolution; meanwhile, select several good solutions whose fitness value

are different from GA population, and then diverts them to PSO population. So it can be achieved that two populations exchange their good individuals. The computation function of convergence facto *cf* is formulated as formula (16), in which f_i denotes the fitness of chromosome *i*.

$$cf = \frac{\sum | f_i - \overline{f} |}{n} \tag{23}$$

During the initial phase of migration operation, smaller the migration rate and larger migration intervals are set. The main reason is to maintain the diversity of population, not destroy the inherent evolution mode of the algorithm. Then, with the increase of evolving generation, migration rates is increased and the migration interval is decreased gradually, which benefits improving of good individual spread in the whole population, convergence speed, accelerating solving speed, and achieving new balance. Thus it assures that it can find best solution of JSSP. This is a dynamic migration strategy.

For each migration strategy, if local search method is employed to conduct deeply search for migrated good individuals, on the one hand, the approximately best or the global best solution of the problem can be found as soon as possible, on the other hand, the better guide for migrating objective population can also be provided. The tabu search (TS) is employed as the local search operator.

A list structure with *1* in length is employed in TS algorithm. In the process of TS search, if neighbor which improves solution is found, the neighbor is saved in the list with probability γ *(0<*γ*<1)*. When the list is full, replace the neighbor in the list with the new neighbor needed to be saved; when better solution is still not found within the maximal generation, pop the neighbor saved in the list, and go on searching. The function of recording neighbor which is potential to improve solution is performed by the list. Meanwhile the neighbor is put into the list once, γ will be decreased once by a certain rate λ *(*λ*<1)*. Because the more deeply the search is conducted, the less the chance of improving solution is. So it make the probability of input list descend, which can lower the chance of the same neighbor is input to the list repeatedly, avoid useless search, and dynamic memory function of the list is achieved.

5.3.3 HPSOGA Algorithm

HPSOGA algorithm is described as:

GAn ← DetermineSizeOfGAPopulation()

PSn ← DetermineSizeOfParticleSwarm()

GAS_{bs} ← NULL；P_c ← 0.85; P_m ← 0.1;

cf = 0；*iter* = 0; *mig* = 0; //mig is migration parameter

P_{gd} ← NULL; P_{id} ← NULL; S_{bs} ←NULL;

ps = 0； //ps is the number of stopping evolvement

```
Begin
ConstructionInitializePopulation ( GAn )
ConstructionInitializeParticleSwarm ( PSn )
while termination conditions not satisfied do
    RouletteWheelSelectionOperation( )              //selection operation
    CrossoverOperation( P_c )
    MutationOperation( P_m )
     CalculatePopulationFitnessValue( )
    if (iter % 5 == 0) then
        cf ← ComputeConvergenceFactor( )
        if ( cf < 0.05 ) then
                mig++
        end if
    end if
     MigrationFromParticleSwarm( mig , TS )         // migration operation
    Update( GAS_bs )
     CalculationParticleValidity( V_id )
     CalculationParticleNewPosition( X_id )
     CalculateParticleFitnessValue( )
     Update( P_id )
     Update( P_gd )
     ps ← NonEvolvementParticleNumber  ()
     if ( ps > 0 ) then
          MigrationFromGAPopulation( ps , TS )      // migration operation
     end if
     iter++
end while
Output  S_bs ← max{ AI(P_gd), AI(GAS_bs ) }
End
```

5.4 Summary

In the section, we introduce another two different hybrid PSO algorithms for solving JSSP. When constructing the hybrid PSO algorithm, the key is how to use the different optimized mechanism of algorithms, making up for deficiencies each other. With the deepened study of the intelligent optimization problem, effective hybrid algorithms will continue to emerge for solving JSSP.

6. Summary

In the chapter, we describe PSO algorithm, propose HPSO algorithm for solving JSSP, analyze the convergence of HPSO, and provide three strategies for deadlock elimination. Then, we present other hybrid PSO algorithm. The chapter can give readers comprehensive research results on hybrid particle swarm optimization algorithms for JSSP, which will promote research and application of JSSP.

7. References

Byung Joo Park, Hyung Rim Choi, Hyun Soo Kim.(2003). A hybrid genetic algorithm for the job shop scheduling problem. *Computers & industrial engineering*, Vol.45 (4),597-613

Cagnina L, Esquivel S, Gallard R.(2004). Particle Swarm Optimization for Sequencing Problems: A Case Study, *Proceedings of 2004 Congress on Evolutionary Computation, Oregon, Portland*, pp. 536-540

Chang Fang，Liaw.(2000). A hybrid genetic algorithm for the open shop scheduling problem. *European Journal of Operational Research*, Vol.124(1), 393-407

Cheng R, Gen M.(1996). Atutorial survey of job-shop scheduling problems using genetic algorithms-I. Representation. *Computers and Industrial Engineering*, Vol. 30(4), 983-997

Christian B, Dirk C M.(1999). Production scheduling and rescheduling with genetic algorithms. *Evolutionary Computation*, Vol.7(1), 1-17

Eberhart R, Kenned J.(1995). A New Optimizer Using Particles Swarm Theory, *Proc Sixth International Symposium on Micro Machine and Human Science*, pp. 39-43. Nagoya, Japan: IEEE service Center.Piscataway

Eugeniusz Nowicki, Czeslaw Smutnicki.(1996). A Fast Taboo Search Algorithm for the Job Shop Problem. *Management Science*, Vol.42 (6), 797- 813

Giffler, B. and Thompson, G. L.(1960). Algorithms for Solving Production Scheduling Problems. *Operations Research*, Vol.8(4), 487-503

Isaacson D L , Madsen R W.(1976). *Markov chain theory and applications*, John Wiley&Sons Inc

Jain A S, Meeran S.(1999). Deterministic job-shop scheduling: past, present and future. *European Journal of Operational Research*, Vol.113(2), 390~434

Jerald J, Asolcan P, Prabaharan G, et al.(2004). Scheduling Optimization of Flexible Manufacturing Systems Using Particle Swarm Optimization Algorithm. *Advanced Manufacturing Technology*

K ennedy J, Eberhart R.C.(2001). *Swarm Intelligence*, San Francisco: San Francisco Morgan Kaufman Publishers

Kirkpatrick, S., Gelatt, C. D. (Jr) and Vecchi, M. P.(1983). Optimization by Simulated Annealing, *Science*, Vol.220(5), 671-680

Kolonko, M.(1998). Some New Results on Simulated Annealing Applied to the Job Shop Scheduling Problem, *the European Journal of Operational Research*

Leticia Cagnina, Susana Esquivel, Raul Gallard.(2004). Particle Swarm Optimization for Sequencing Problems: A Case Study, *Proceedings of 2004 Congress on Evolutionary Computation, Oregon, Portland,* pp. 536-540

Lian Z G, Jiao B, Gu X S.(2006). A similar particle swarm optimization algorithm for job shop scheduling to minimize makespan. *Appl. Math. Comput.* , Vol.183, 1008-1017

Liu H B, Abraham A, Choi O, et al.(2006). Variable neighborhood particle swarm optimization for multi-objective flexible job shop scheduling problems. *LNCS*, Vol.4247, 197-204

Ph. Preux, E.-G. Talbi.(1999). Towards hybrid evolutionary algorithms. I*nternational transactions in operational research*, Vol.6 (6), 557-570

Radcliffe N .J, Surrv P D.(1995). *Fundamental limitations on search algorithms*, U K: Cniversity of Edinburnh

Sha D Y, Hsu C Y.(2006). A hybrid particle swarm optimization for job shop scheduling problem. *Comput. Ind. Eng*. Vol.51, 791-808

SongXiao-Yu, Cao Yang, Meng Qiu-Hong.(2008). Study on particle swarm algorithm for Job Shop scheduling problems. *Xi Tong Gong Cheng Yu Dian Zi Ji Shu/Systems Engineering and Electronics*, Vol.30(12), 2398-2401

Van Laarhoven, P. J. M.,Aarts, E. H. L., Lenstra, J. K.(1992). Job Shop Scheduling by Simulated Annealing, *Operations Research*, Vol.40,113-135

Wolpert D H, Macready W G.(1995). *No Free Lunch theotrems on search algorithms,* U K: Univercity of Edinburnh

Xia weijun, Wu zhiming, Zhangwei, et al.(2004). A New Hybrid Optimization Algorithm for the Job Shop Scheduling Problem. *Proceedings of the 2004 American Control Conference Boston, Massachusette*, pp. 5552-5557

Xia W J, Wu Z M.(2005). An effective hybrid optimization approach for multi-objective flexible job shop scheduling problems. *Comput. Ind. Eng.* Vol.48, 409-425

Xia W J, Wu Z M.(2006). A hybrid particle swarm optimization approach for the job shop scheduling problem. *Int. J. Adv. Manuf. Technol.* Vol.29, 360-366

Xu Gang, Wu Zhi ming.(2002). Deadlock-free scheduling method using Petrinet model analysis and GA. *Proc of the IEEE Int Conf on Control Application*, pp. 1153-1159

Yamada, T. and Nakano, R.(1996). Job-Shop Scheduling by Simulated Annealing Combined with Deterministic Local Search, *Meta-heuristics: Theory and Applications, Kluwer Academic Publishers*, Boston, MA, USA, Chapter 15, pp. 237-248

Yamada, T. and Nakano, R.(1996). Scheduling by Genetic Local Search with Multi-Step Crossover, *PPSN'IV Fourth International Conference on Parallel Problem Solving from Nature*, pp. 960-969, Berlin, Germany, Sept 22-26

Yamada, T. and Nakano, R.(1996). A Fusion of Crossover and Local Search, *ICIT'96 IEEE International Conference on Industrial Technology*, pp. 426-430, Shanghai, China, Dec 2-6

Yannakakis, M.(1990). The Analysis of Local Search Problems and their Heuristics, *Proceedings of the 7th Annual Symposium on Theoretical Aspects of Computer Science*, pp. 298-311.

Yannakakis, M.(1997). Computational Complexity of Local Search, in Aarts, E. H. L. and Lenstra, *J. K (eds) Local Search in Combinatorial Optimization, Wiley, Chichester, Chapter 2,* pp. 19-55

Future Manufacturing Systems

Edited by Tauseef Aized

ISBN 978-953-307-128-2

Hard cover, 268 pages

Publisher Sciyo

Published online 17, August, 2010

Published in print edition August, 2010

This book is a collection of articles aimed at finding new ways of manufacturing systems developments. The articles included in this volume comprise of current and new directions of manufacturing systems which I believe can lead to the development of more comprehensive and efficient future manufacturing systems. People from diverse background like academia, industry, research and others can take advantage of this volume and can shape future directions of manufacturing systems.

INTECH

open science | open minds

InTech Europe

University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China

Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen