ATLANTIS PRESS

Advances in Intelligent Systems Research, volume 146

2nd International Conference on Artificial Intelligence: Technologies and Applications (ICAITA 2018)

Study and Application of Glowworm Swarm Optimization in Flexible Job Shop Scheduling

Hongtao Wang[1], Dailin Xu[2], Chun Yan[2] and Hao Pan[2,*]

[1]Network Information Center, Wuhan University of Technology, Hubei Wuhan 430070, China

[2]School of Computer Science and Technology, Wuhan University of Technology, Hubei Wuhan 430070, China

[*]Corresponding author

***Abstract*—For satisfying performances of the job-shop scheduling (JSP) issues and as for those characteristics (such as easily falling into local optimum, insufficient stability and slower convergence rate) of the standard GSO, a PDE framework taking advantages of good global search performance, simple implementation and rapid convergence rate and a self-adaptive mechanism based on the S-shape variable-step for its improvement so that the balance between global search and local mining may be coordinated and its stability and convergency may be improved. Simulation experiments were carried out to our hybrid GSO based on the Brandimarte international standard examples and comparison and analysis were performed to the experimental results to verify its reliability and applicability.**

Keywords*—Job-Shop Scheduling (JSP) issue; Glowworm Swarm Optimization (GSO); Particle Swarm Optimization (PSO); Permutation-based Differential Evolution (PDE); self-adaptation

I. INTRODUCTION

Along with the rapid rise of the wave of information, users are having more and more demands and products are increasingly diversified and complicated; and the product quality requirements and time cost control are becoming higher than ever in market. The modern production pattern towards more varieties and small batches completely replaced the traditional one for only guaranteeing the production capacity; especially, the most significant change is the smart, information service and small batched process of production [1-2].

Entrepreneurs and scholars are focusing on high efficiency, flexibility and reliability (3-High for short) for manufacturing products. Improvement of the production scheduling plan as the key factor for achieving 3-High is also the core idea of modern enterprise production management. Production scheduling [3-4] is to improve the operation efficiency of various parts of a production line by means of production orders which meet the technological constraints as can as possible based on good operation of various machines. In case of any unexpected event, an efficient operation strategy may contribute the most rational approach for companies to ensure not only the emergency goals but also the various required indexes shall be achieved; for example, the manufacturing period is shortened and the inventories fall [5-7]. Along with the rapid development of the automation technology in the field of manufacturing, the case (the scheduling strategy depends on the human experiences) gradually gets out of the enterprise's production mode; on the other hand, selection of the effective and intelligent algorithm shall be crucial for customization of the process of scheduling plan.

GSO was utilized to solve the FJSP issue. In view of the standard GSO untimely falling into the local optimum and having disadvantages such as insufficient stability and slow convergence rate, the corresponding improvement method was put forward base on the S-shape variable-step self-adaptation mechanism and the PDE framework taking advantages of good global search performances, simple implementation and rapid convergence rate to coordinate the balance between global search and local mining and improve the stability and convergency of the algorithm.

II. IMPROVEMENT OF THE STANDARD GSO

A. Profile of GSO

As for GSO, glowworm individuals of the initial population may be randomly dispersed within the definition domain of the objective function. Each glowworm has its own search range where glowworms move to the brighter glowworm; similarly, those dimmer glowworms may also be attracted. While brightness of a glowworm changes, its own luciferin value which is proportional to the corresponding fitness of the objective function at its own position might also corresponding change; namely, larger fitness indicates a better position of the glowworm and it is more attractive for other glowworms within its search range. Glowworm move constantly and their luciferins, positions and decision ranges always update so that all glowworm individuals may be gathered within a certain range around a glowworm whose luciferin is higher after the specified number of iterations; thus, a few extremum points may be found for the issue to achieve optimization.

GSO may be generalized as the following four stages:

1) Initialization: The algorithm parameters (including radius of decision (r_s), decision range updating factor (η), radius of perception (r0), luciferin volatilization factor [ρ(0<ρ<1)] and fitness extraction ratio (γ)) and positions of glowworms shall be initialized.

2) Luciferin updating and changing: The luciferin value of any glowworm is directly related to its position, where the lower the fitness the smaller the luciferin value is. In addition, a part of luciferin would be volatilized while a slowworm moves.

Thus, its luciferin value shall be calculated prior to its next motion process; and the updating equation is as follows:

$$l_i(t+1) = (1-\rho) l_i(t) + \gamma J_i(t+1) \tag{1}$$

where: li(t)represents the luciferin value of Glowworm i at the tth iteration; and Ji(t+1) represents the fitness of Glowworm i at the (t+1)th iteration.

3) Updating the glowworm positions: After updating the luciferin value of the glowworm every time, it may selected a glowworm whose luciferin value is larger within its decision range and then moves closely. There are generally a few glowworms suitable for its moving; thus, the probability of its moving towards each neighboring glowworm shall be calculated by:

$$p_{ij} = \frac{l_j(t) - l_i(t)}{\sum_{k \in N_i(t)} l_k(t) - l_i(t)} \tag{2}$$

where: $N_i(t) = \{ j \mid d_{i,j}(t) < r_d^i(t), l_i(t) < l_j(t) \}$ ($j \in N_i(t)$) represents the eligible neighboring glowworms set for Glowworm i at the tth iteration; di,j(t) represents the Euclidean space distance between Glowworms i and j at the tth iteration; and $r_d^i(t)$ represents the radius of the decision range of Glowworm i at the tth iteration.

In accordance with the Roulette rules, the Neighboring Glowworm j is selected to update the position of Glowworm i in accordance with the following equation:

$$x_i(t+1) = x_i(t) + s\left(\frac{x_j(t) - x_i(t)}{\left\| x_j(t) - x_i(t) \right\|} \right) \tag{3}$$

4) Updating the scope of the decision range: each glowworm may dynamically update its the radius of decision based on its decision range updating factor after a motion process; and the decision radius is updated in accordance with the following equation:

$$r_d^i(t+1) = \min\left\{ r_s, \max\left[0, r_d^i(t) + \eta\left(n_t - \left| N_i(t) \right| \right) \right] \right\} \tag{4}$$

where: rs represents the radius of decision; η represents the dynamic decision range updating factor; and nt represents the threshold of the size of Ni(t).

B. Advantages and Disadvantages of GSO

After a specified number of iterations, all individuals in the population of glowworms may move to those glowworms located in better positions, respectively, which are regarded as the extremum values of the objective function; and the optimized extremum value shall be the global optimal solution. Thus, GSO may be utilized to gain the global optimal solution for a single-mode optimization function; on the other hand, it may also be to solve the local optimal solution for a multi-mode function. During the iteration period, glowworms may not be affected each other. Thus, this algorithm takes advantages of concurrent and simple implementation, easy operation and only a few parameters. In addition, this algorithm has a strong local search capability because each glowworm has its own perception range, where the optimal solution may be quickly found within a certain range.

In spite of many advantages of GSO, it still has defects such as slow convergence, poor stability and low solution precision.

The optimal solution of GSO primarily depends on each glowworm moving towards those excellent individuals within its decision range till the optimal value has been found. The decision range of each individual depends on its own radius of perception. If there was no eligible neighboring glowworm within the decision range for a certain glowworm, it glowworm would stop search. In extreme cases, if all glowworms were spaced very sparsely or their luciferin values were equal, GSO would not be convergent to the peak. Thus, GSO is extremely dependent on the excellent individuals to lower the detection and convergence rates for peaking.

Besides, glowworms may gather around the peak and tend to convergence at the end of iterations, when glowworm individuals have been very close to the peak. If any glowworm individual moved by an oversized step, it would cross the peak not to achieve the convergence; namely, oscillation would constantly occur near the peak; thus, the solution convergence rate and precision would be reduced.

III. Improvement of GSO

A. Improvement of GSO Based on the Step Self-adaptive Mechanism

Attraction between glowworm individuals may be under control by means of adjustment of their own glowing intensities so that individuals may move; and such process is optimized by means of GSO where the step (s) shall be crucial for its convergence and stability; on the other hand, the convergence and stability would not be compromised better while a constant step was applied. A larger step would intensify the probability of "prematurity" convergence and lead to remarkable oscillation at the end of the iteration period though the convergence would be effectively improved; thus, the resulting precision would fall; otherwise, a too small step would greatly lower the convergence rate. A kind of S-shape dynamically changing step was applied here to replace the constant step for the standard GSO with reference to the feature of sine function based on the standard GSO.

The maximum number of Iterations and the initial step for all individuals are set as Tmax and S0, respectively. The step may change adaptively in the S shape along with the growth of the number of Iterations (t), which is updated by:

$$s = s_0 - s_{\min} \cdot \beta \cdot \sin\left(t / T_{\max}\right) \tag{5}$$

where: Smin represents the minimum step threshold; and β represents the control parameter.

While assuming the initial parameters (S0=10 , β=2, Smin=5, and Tmax=200), the corresponding step changing curve is shown in Figure 1. At the beginning of iterations, the step keeps relatively large to effectively accelerate the convergence but prevent "prematurity" of the algorithm; on the other hand, the step keeps relatively small at the end of iterations and there would be a trade-off between its convergency and stability. While the step reaches 0, the self-adaptation would grow.

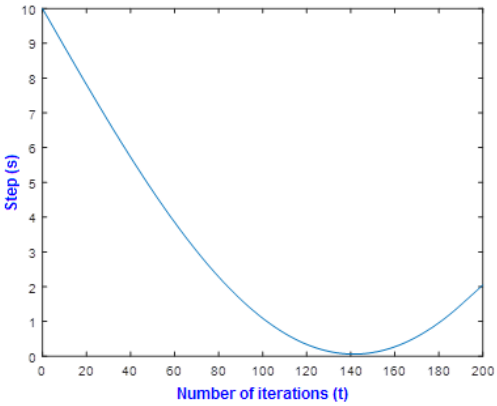

FIGURE I. STEP VS. NUMBER OF ITERATIONS

For improvement of the search performances of the algorithm, how to maintain a trade-off between global development and local mining is very important in the process of optimization. Fusion of PDE [8-9] as a relatively open calculation framework of the local search method is convenient. Pan et al. verified the global optimization by means of PDE and then local search trials around individuals; and this approach is highly desirable. Thus, PDE was applied here to improve CSGSO and increase the richness of population.

In comparison of the traditional differential evolution algorithm, PDE defines the position-based addtion & subtraction for mutation operation in accordance with permutation features. For lowering damage to good structures as can as possible and guaranteeing the convergence rate, the crossing operation is introduce based on permutation during the crossing stage for PDE.

1) Representation of solution: For easy expression of the following solution processes, the solution of the scheduling issue was defined here based on the permutation method. If there are n procedures, the solution is expressed by P=(π1,π2,...πn) where (π1,π2,...πn) represents a permutation including n procedures.

2) Mutation:

$$V = P_3 \oplus \left(P_1 \otimes P_2\right) = P_3 \oplus L \tag{6}$$

where: $\otimes$ represents the difference between positions of two kinds of permutation methods for calculation of the same task; and L represents the position deviation vector defined as the operational result of $P_1 \otimes P_2$, whose dimensions is the same as that of P_1 or P_2.

While assuming P1=(5,7,3,4,1,6,2) and P2=(2,6,4, 3, 7, 5, 1) and by taking Task 5 as an example, it is the first task of P1 and the sixth task of P2; then their position difference is 5 and the corresponding calculation result (L) is shown in Figure 2. $\oplus$ represents a new permutation converted from the current permutation in accordance with Position L . While assuming P3=(7, 3, 2, 4, 1, 6, 5), the first task of P3 is 7 and the corresponding position difference is 5 in L ; thus, Task 7 is positioned at Position 1+5=6 in the new permutation (V). The complete V is shown in Figure 3.

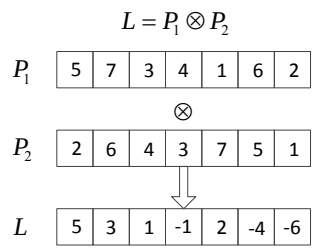

FIGURE II. A $\otimes$ OPERATION EXAMPLE

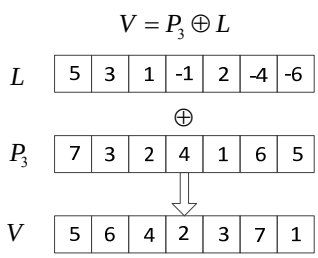

FIGURE III. A $\oplus$ OPERATION EXAMPLE

3) Crossing: For maintaining good structures of the parent population to the progeny population as can as possible, crossing is defined here by the following method:

Step 1: Random Integers R1 and R2 are generated within [1,n], which serve as the position information of the crossing segment.

Step 2: the segment between R1 and R2 in Parent Individual P1 or P2 or the segment before R1 or behind R2 in P1 or P2 is copied to Progeny Individual C.

Step 3: the task included in Progeny Individual shall be deleted in another parent individual; and the remaining tasks (excluded in Progeny Individual C) shall be filled in those vacant positions of Progeny Individual C. Related descriptions of crossing are represented in Figure 4.

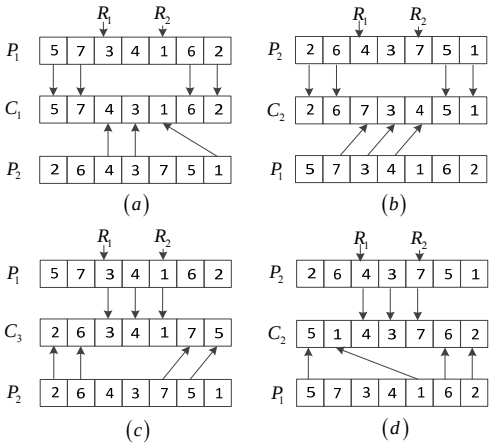

FIGURE IV. CROSSING OPERATION EXAMPLES

4) Reparation: Based on mutation and crossing, PDE rather than DE is more sensitive to reparation; on the other hand, its characteristics shall be comprehensively taken into account for solution of an actual issue and reparation shall be depend on the actual demands. For example, while a batch scheduling issue is solved by means of PDE, mutation and crossing can guarantee all solutions during the evolution process shall be feasible so that reparation shall be unnecessary. On contrast, if a FJSP issue is solved by means of PDE, the same workpiece may be processed in various sequences for procedures; thus, the corresponding reparation shall be prepared in accordance with such constraint condition. It is assumed that Sequences(1,2,3) shall be satisfied among Tasks 1, 2 and 3 as for Individual. The reparation (Figure 5) is put into operation to first find out the relative positions of Tasks 1, 2 and 3 in P1; and then 3 positions may be adjusted each other.

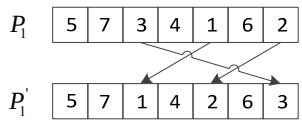

FIGURE V. A REPARATION EXAMPLE

5) Selection operation: By taking the objective function value as the selection standards, comparison is performed to the locally searched optimal individual and individuals of the current population to ensure those better individuals shall move into the progeny population.

IV. APPLICATION OF HYBRID GSO INTO FJSP

A. Objective Function and Fitness Design

The minimum manufacturing period was regarded as the index for evaluation performances of FJSP here; and the objective function is:

$$\min f = \min C_{\max} = \min\left(\max_{1\le j\le n}\left(C_j\right)\right) \tag{7}$$

where: n represents the number of workpieces.

The above minimization issue may be directly converted into the following fitness function:

$$Fit\left(f\left(x\right)\right) = -f\left(x\right) \tag{8}$$

B. Solution Steps of Hybrid GSO

Due to the flexibility factor, FJSP rather than traditional JSP is more complicated. As for solution of FJSP, the technology processes shall be met, moreover, not only the constraint relationship between sequences of a few procedures for the same workpiece but also the processing period and state of any machine shall be comprehensively taken into account; thus, each procedure shall be performed by means of a rational machine. After completion of crossing, individuals may generally occur; and they will be compared with the original ones to select the optimal individual for the next generation. Such method guarantees those excellent individuals shall remain as can as possible in the evolution process. Integration of the discrete differential evolution based on permutation and step self-adaptation mechanism may lead to the hybrid GSO for operation, whose implementation steps are as follows:

Step 1: initialization of algorithm parameters (such as number of iterations (Gmax), local search probability (pl) and number of population);

Step 2: Random generation of Parent Population X by means of the randomly generated algorithm and calculation of the target value of the corresponding scheduling strategy for each individual; and updating the optimal individual (current_optimal_X). Assuming k=0;

Step 3: Judgment of $gen \ge iter$? Yes, implementation of the population improvement strategy; and assuming gen=0. Then, go to Step 4; otherwise, directly go to Step 4;

Step 4: implementation of mutation to Population Xk (currently the kth generation) to gain Population Vk; and assuming i=1;

Step 5: implementation of crossing to the ith individual of Population Vk to gain four temporary vectors, respectively;

Step 6: implementation of reparation to the four temporary vectors to gain 4 new vectors and comparison of fitness values of 4 new vectors to get the optimal temporary vector, which is regarded as the ith individual of Population Uk;

Step 7: Judgment of $i \ge NP$? Yes, go to Step 8 and assuming j=1; otherwise, assuming i=i+1, go to Step 5;

Step 8: Random generation of a real number to the jth individual (U_k^j) of Population Uk and Judgment of this real number being less than pl? Yes, go to Step 9; otherwise, go to Step 11;

Step 9: To call CSGSO for local search and evaluation of U_k^j; if an individual ($U_k^{j'}$) better than U_k^j is searched, $U_k^{j'}$ replaces U_k^j;

Step 10: Judgment of $j \geq NP$? Yes, go to Step 11; otherwise, assuming j=j+1, go to Step 8.

Step 11: Comparison of the fitness values of the mth individual (X_k^m) of Population X_k and the mth individual (U_k^m) of Population U; if U_k^m is better than X_k^m in effects, U_k^m is used to update X_k^m.

Step 12: Judgment of $m \geq NP$? Yes, go to Step 13; otherwise, assuming m=m+1, go to Step 11.

Step 13: Judgment of the optimal individual has changed? Yes, updating $current\_optimal\_X$, and assuming $gen = 0$; otherwise, assuming gen=gen+1, updating current_optimal_X.

Step 14: Judgment of $G \geq G_{\max}$? Yes, outputting statistical results for several iterations; otherwise, assuming $G = G+1$, go to Step 3.

C. Solving FJSP by Means of Hybrid GSO

1) Population initialization: As for starting the evolution algorithm, selection of which method for generation of the initial population is generally greatly related to its search efficiency and the quality of its last solution so that it shall be undoubtedly important. As for solving any FJSP issue, a feasible solution shall include two aspects, namely encoding procedures and machines; the former is to specify scheduling sequences for various procedures; and the latter is to define the processing machine for each procedure.

While assuming:

NP represents the number of individuals of a Population;

G represents the current number of iterations ($G = 0, 1, \cdots, G_{\max}$)

$X_{i,G}$ represents the ith individual of the Gth generation of the Population, and

$$X_{i,G} = \left[S_{i,G}, R_{i,G}\right]^T$$

where: $\vec{S}_{i,G} = \left[s_{1,i,G}, s_{2,i,G}, \cdots, s_{d,i,G}\right]$ represents the encoding part of procedures for $X_{i,G}$; $\vec{R}_{i,G} = \left[r_{1,i,G}, r_{2,i,G}, \cdots, r_{d,i,G}\right]$ represents the encoding part of machines for $X_{i,G}$; and d represents the total number of all procedures to be processed.

The initial population (G=0) was generated here by means of a random function.

(1) Generation of the temporary population (Y, size: represents NP and dimensions: D=2d), whose ith individual is represented by $Y_i = [y_{1,i}, y_{2,i}, \ldots, y_{D,i}]^T$; and

(2) Generation of the jth variable by means of Eq. (9), whose minimum and maximum are represented by $y_{j,\min}$ and $y_{j,\max}$, respectively.

$$y_{j,i,0} = y_{j,\min} + \left(y_{j,\max} - y_{j,\min}\right) \times rand\left(0,1\right) \tag{9}$$

For convenience, the corresponding range ($\left[y_{j,\min}, y_{j,\max}\right]$, $j = 1, 2, \ldots, D$) for each variable of Y is assumed as $[-\delta, \delta]$ (for example: $\delta = 1$). As for the individual vector ($Y = [y_1, y_2, \ldots, y_D]^T$), the random variable ($y_j$) shall be necessarily converted into a variable conforming to the scheduling constraints for its rational application in the PDE-based algorithm; namely:

$\vec{Y}_1 = [y_1, y_2, \ldots, y_d]$ → Procedures encoding: $\vec{S}_i = \left[s_1, s_2, \cdots, s_d\right]$; and

$\vec{Y}_2 = \left[y_{d+1}, y_{d+2}, \ldots, y_D\right]$ → Machines encoding: $\vec{R}_i = \left[r_1, r_2, \ldots, r_d\right]$.

2) Encoding and decoding: PDE-based variable-step GSO was utilized here to solve the FJSP issues, which is different from any traditional evolution algorithm. Mutation and crossing of PDE are performed based on permutation to guarantee the integrity of a structure block to a certain extent. Thus, it is necessarily guaranteed that each individual of the initial population shall be of practical significance prior to starting any iteration; namely, the above $\vec{S}_i = \left[s_1, s_2, \cdots, s_d\right]$ and $\vec{R}_i = \left[r_1, r_2, \ldots, r_d\right]$ shall possess real significance. Finally, the initial population was gained, whose individuals vector is $X = [S, R]^T$

For easy solution, various procedures may be sequenced and they correspond to a unique integer (ID, $ID = 1, 2, \ldots, d$) which may exclusively represents a procedure.

After gaining ID for each procedure, the actual meanings of the encoding vectors ($\vec{S}$ and $\vec{R}$) may be explained as follows:

$\vec{S}$ represents the sequential processing procedures; and

$\vec{R}$: Variable r_j represents Procedure s_j may be performed in the rjth machine of the machine set.

For example, $\vec{S}=[4,1,7,5,2,8,3,9,6]$, various procedures shall be scheduled in accordance with sequential constraints for FJSP procedures by means of the following sequences:

O_{21}, O_{11}, O_{31}, O_{22}, O_{12}, O_{32}, O_{13}, O_{33} and O_{23}

Conversion of $\vec{Y}_1$ into $\vec{S}$ was performed in accordance with Largest Position Value (LPV) rules put forward by Wang et al. The following result may be obtained based on Eq. (9):

$$\vec{Y}_1=[0.7,-0.2,0.3,0.9,-0.6,0.5,-0.4,-0.3,0.1] \quad (9)$$

Each variable y_j of $\vec{Y}_1$ corresponds to Integer $ID=j$; and descending sorting of various variables of $\vec{Y}_1$ leads to a full permutation on ID. For satisfying the sequential constraints of various procedures for the same workpiece, the ID permutation shall be adjusted and $\vec{Y}_1$ shall be adjusted correspondingly to gain finally $\vec{S}$.

As for transition from $\vec{Y}_2$ to $\vec{R}$, $\vec{L}=[l_1,l_2,\cdots,l_d]$ shall be first specified, where l_j ($j=1,2,\ldots,d$) represents that the size of the machines set may be selected for Procedure ID ($ID=j$). Then, conversion from $y_j\in[-\delta,\delta]$ to $r_j\in[1,l_j]$ may be performed by means of Eq. (10).

$$r_j=round\left(\frac{1}{2\delta}(l_j-1)(y_j+\delta)+1\right) \quad (10)$$

So far, conversions from $\vec{S}$ to $\vec{Y}_1$ and $\vec{Y}_2$ to $\vec{R}$ may be performed and scheduling constraints are satisfied; moreover, the initial population is of practical significance, where each individual is $\vec{X}=[S,R]^T$.

3) Evolution: PDE is to solve FJSP based on permutation; namely, the scheduling sequences are to be optimized while the machines encoding has been determined. Thus, evolution was primarily carried out here in view of procedures encoding ($\vec{S}$) for solving FJSP; and the machine called for each procedure shall not change along with evolution.

Mutation: it is a key link for guaranteeing the richness of a population and it means: an individual vector ($\vec{X}_{i,G}$) is selected from the parent population as the target vector and a series of interference policies is performed to this target vector to generate Mutation Vector $\vec{V}_{i,G}=[v_{1,i,G},v_{2,i,G},\ldots,v_{D,i,G}]^T$. Mutation described in Section 2.2 was performed here to $\vec{S}_{i,G}$ corresponding to $\vec{X}_{i,G}$ to generate Procedures Encoding Vector $\vec{V^S}_{i,G}$ corresponding to $\vec{V}_{i,G}$, whose operation conforms to Eq. (11). The machine selected for each procedure remain unchanged for the moment; whereas, $\vec{R}_{i,G}$ is correspondingly adjusted along with mutation to $\vec{S}_{i,G}$ to gain a new full permutation (Machines Encoding Vector $\vec{V^R}_{i,G}$); namely $\vec{V}_{i,G}=\left[\vec{V^S}_{i,G},\vec{V^R}_{i,G}\right]^T$.

$$\vec{V^S}_{i,G}=\vec{S}_{i,G}\oplus\left(\vec{S}_{r_1^i,G}\otimes\vec{S}_{r_2^i,G}\right) \quad (11)$$

where: r_1^i and r_2^i represent two integers which are randomly generated in $[1,NP]$ and are not i.

Crossing: it is the basis for individuals of a population to exchange information. With reference to descriptions in Section 2.2, crossing of $\vec{X}_{i,G}$ and $\vec{V}_{i,G}$ may generate 4 new vectors (namely $C^1_{i,G}$, $C^2_{i,G}$, $C^3_{i,G}$ and $C^4_{i,G}$).

Reparation: $C^1_{i,G}$, $C^2_{i,G}$, $C^3_{i,G}$ and $C^4_{i,G}$ are possibly not conform to the sequential constraints which shall be satisfied for various procedures for the same workpiece, which may be correspondingly adjusted by means of reparation described in Section 3.3.1 in this case so that they may be converted into a feasible solution. Then their fitness function values are calculated; and the vector whose fitness function value is maximum is selected as Test Vector $\vec{U}_{i,G}=\left[u_{1,i,G},u_{2,i,G},\ldots,u_{D,i,G}\right]^T$.

4) Local search: PDE has strong global search capacity but it may possibly fall into the local optimum. For maintaining a trade-off between development and mining in the search space, a neighborhood structure (N6) was established based on the critical path to $\vec{U}_{i,G}$, where was searched by means of the variable-step GSO described in Section 2.1 to find out better individuals as can as possible for updating $\vec{U}_{i,G}$.

5) Selection: A test vector ($\vec{U}_{i,G}$) may generally be gained by means of the above evolution. Generally, the operation for selecting the vector remaining in the progeny is called as selection. The strategy of PDE is generally to select an individual whose fitness value is relatively good from $\vec{U}_{i,G}$ and $\vec{X}_{i,G}$ and maintain such individual in the progeny; and the specific operation equation is as follows.

$$\vec{X}_{i,G+1}=\begin{cases}\vec{U}_{i,G} & (f(\vec{U}_{i,G})\le f(\vec{X}_{i,G}))\\ \vec{X}_{i,G} & \text{(other cases)}\end{cases} \quad (11)$$

6) Improvement of the population by means of the pre-scheduling strategy: While evolution algorithm is put into operation to search the optimal solution of a scheduling issue, no better quality solution may always be found out by means of several continuous iterations. Such phenomenon is inevitable currently for the evolution algorithm. A pre-scheduling strategy was put forward to improve the population in view of such case.

As for Population $\vec{X}$ to be improved, $\vec{S}$ presents the scheduling sequences for all processing procedures. Now, the decoding process may be performed to improve $\vec{R}$ so that various procedures may be completed as soon as possible and the entire manufacturing period may be shortened. The operation processes are as follows:

1) The first procedure of $\vec{S}$ is taken;

2) The processing equipment to complete the procedure as soon as possible is selected as the processing machine in the machine set for this procedure;

3) The state of the above machine is set as BUSY;

4) Similarly, the above steps are repeated to determine machines for the 2nd, 3rd, ……procedures till processing machines for all procedures are determined.

V. EXPERIMENTAL STUDY ON SOLVING FJSP BY MEANS OF HYBRID GSO

A. Related Experimental Settings

The above HGSO was performed here by means of our MATLAB program whose operation environment is as follows: i3 CPU; frequency: 2.1G; memory: 4GB

For verifying the feasibility of the improved HGSO solving FJSP, 10 international standard examples (Mk01-Mk10) designed by Brandimarte were utilized as our simulation objects. Because the randomness of the algorithm would give rise to a certain calculation error, each test case was continuously simulated for 20 times. With reference to the existing research results, the setting parameters of the algorithm are presented in Table 1. The minimum manufacturing period is the optimization objective for all measurement experiments.

TABLE I. SETTING PARAMETERS OF THE ALGORITHM

Size of evolution population	1000
Number of Iterations	500
Number of Iterations for local search	50
Fitness extraction ratio	0.6
Luciferin volatilization factor	0.4
Control parameter	0.08

B. Experimental Results and Analysis

Some operation results for solving BRdata by means of LEGA and our HGSO are presented in Table 2. LEGA was put forward by Ho et al. by fusion of the machine study mechanism and GA. $n \times m$ represents the numbers of workpieces and machines in the issue; LB and UB represent the lower and upper bounds of the optimal solution, respectively; C^*_{best} represents the optimal value in the literature; Cbest represents the optimal solution (namely the optimized manufacturing period) for our HGSO; Aver represents the average operation result; and t represents the average operation period for CPU to continuously be put into operation for 20 times in the same environment, whose unit is second.

Direct comparison was carried out to the relative error ($dev = 100 \times \left(C^*_{best} - C_{best}\right) / C^*_{best}$) and results in other literatures. Figure 6 shows the Gantt chart for an optimal solution scheduling by means of HGSO.

TABLE II. OPERATION RESULTS OF BRDATA

Issue	$n \times m$	(LB,UB)	LEGA		HGSO			
			C^*_{best}	$Aver$	C_{best}	$Aver$	t	dev
Mk01	10×6	$(36,42)$	40	41.5	40	40.0	1.7	0
Mk02	10×6	$(24,32)$	29	29.1	26	26.1	2.5	10.3
Mk03	15×8	$(204,211)$	204	204.0	204	204.0	1.3	0
Mk04	15×8	$(48,81)$	67	67.3	60	60.4	4.8	10.4
Mk05	15×4	$(168,186)$	176	178.1	173	175.3	8.2	1.7
Mk06	10×15	$(33,86)$	67	68.8	60	60.2	16.2	11.7
Mk07	20×5	$(133,157)$	147	152.9	139	140.0	20.5	5.4
Mk08	20×10	$(523,523)$	523	523.3	523	523.0	2.6	0
Mk09	20×10	$(299,369)$	320	327.7	307	307.0	34.6	4.1
Mk10	20×15	$(165,296)$	229	235.7	200	204.6	41.4	12.2

Table 2 presents optimal solutions, average optimal solutions, average operation periods and relative errors while HGSO were put into operation for several times to those test examples in various sizes. In comparison of those optimal solutions for LEGA, optimal solutions for 7 examples are better for HGSO; in addition, optimal solutions are gained for another 3 examples for the listed algorithms. On the whole, our HGSO rather than LEGA achieved better performances.

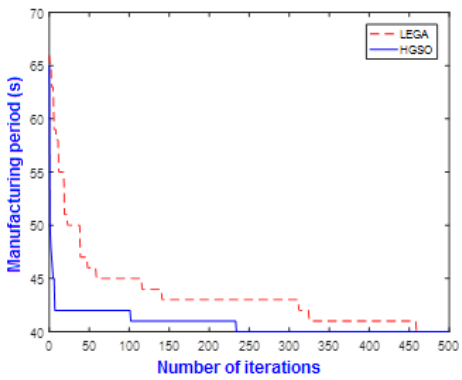

FIGURE VI. CONVERGENCE OF MK02 EXAMPLE

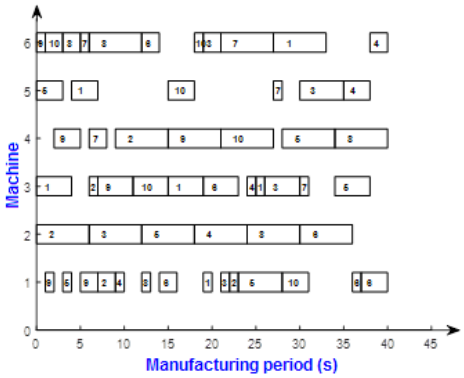

FIGURE VII. THE CORRESPONDING PROCESSING GANTT CHART FOR THE OPTIMAL SCHEDULING FOR MK01

For visual presentation of calculation results, Figure 6 presents comparison of convergence curves for Mk01 by means of HGSO and LEGA, which indicate that HGSO's objective function value falls faster than that of LEGA though their optimal solutions are the same; and the numbers of iterations for the same optimal solution (40) are 460 for LEGA and 235 for HGSO, respectively; thus, HGSO has better convergence.

Figure 7 presents the corresponding Gantt chart for an optimal solution for Mk01 by means of HGSO, where data in bars represent workpieces for procedures and the sequential bars marked the same data represent various procedures in turn for the same workpiece. Figure 8 indicates that this scheme may satisfy all constraints for FJSP.

VI. CONCLUSIONS

As for those defects (such as weak global search performances, easily falling into local optimum, insufficient stability and slower convergence rate) for solving FJSP issues by means of GSO, some improvement measures were taken to the standard GSO. Our conclusions are as follows:

1) In view of those defects (such as slow convergence rate and easily falling into local search) of GSO, the PDE framework taking advantages of good global search performances, easy implementation and rapid convergence rate was utilized to gain a trade-off between the global development and local mining and improve the convergence rate.

2) In view of those defects (such as poor stability and slow convergence) of GSO, the step self-adaptive mechanism was utilized to improve the constant step for the standard GSO and its feasibility and effectiveness were verified by means of simulation experiments.

3) Our HGSO was applied into FJSP for performance of local search based the neighborhood structure (N6); and the pre-scheduling strategy was utilized to deal with the optimal solution unchanging for long time; and the detailed solution processes were described. Based on Brandimarte standard examples, simulation was performed for many times. Comparison and analysis of operation results for HGSO and LEGA proved that HGSO shall be feasible and effective for solving FJSP.

REFERENCES

[1] Chen Tongjiang. The Research on Deepening the Application Strategies in Manufacturing Informatization [D]. Huazhong University of Science and Technology, 2010.

[2] Michael Pinedo, et al. Scheduling: principle, algorithms and system [M]. Tsinghua University Press, 2007.

[3] CHENG Qi, et al. Research on Information Resource Management Methods for Large Manufacturing Enterprise [J]. Machine Design and Manufacturing Engineering, 2012, 41(9): 20-22.

[4] Gu Xinlin, Zhang Dong, et al. Integration for manufacturing servitization and informationization [J]. Computer Integrated Manufacturing Systems, 2010, 16 (11): 2530-2536.

[5] Kammer M, Akker M V D, Han H. Identifying and exploiting commonalities for the job-shop scheduling problem [J]. Computers & Operations Research, 2011, 38(11): 1556-1561.

[6] Wang Chao, Liu Jieping, Chang Weitao, et al. Progress on Technology of JSP Dynamic Scheduling [J]. Equipment Manufacturing Technology, 2011 (4): 144-148.

[7] Liu Xiangde. Research on Some Critical Issues about Job Shop Real-time Scheduling [D]. Chongqing University, 2013.

[8] Wang Shenwen, Ding Xinli, et al. Survey of Differential Evolution [J]. Journal of Wuhan University (Natural Science Edition), 2014, 60 (4): 283-292.

[9] Wang Wanliang, Wang Lei, Wang Haiyan, et al. Dynamic Job Shop scheduling based on hybrid differential evolution algorithm [J]. Computer Integrated Manufacturing Systems, 2012, 18 (3): 531-539.