A Self-adaptive Hybrid Population-Based Incremental Learning Algorithm for *M*-Machine Reentrant Permutation Flow-Shop Scheduling

Zuo-Cheng Li, Bin Qian*, Rong Hu, Chang-Sheng Zhang, and Kun Li

Department of Automation, Kunming University of Science and Technology, Kunming 650500, China
bin.qian@vip.163.com

Abstract. This paper proposes a self-adaptive hybrid population-based incremental learning algorithm (SHPBIL) for the *m*-machine reentrant permutation flow-shop scheduling problem (MRPFSSP) with makespan criterion. At the initial phase of SHPBIL, the information entropy (IE) of the initial population and an *Interchange*-based search are utilized to guarantee a good distribution of the initial population in the solution space, and a training strategy is designed to help the probability matrix to accumulate information from the initial population. In SHPBIL's global exploration, the IE of the probability matrix at each generation is used to evaluate the evolutionary degree, and then the learning rate is adaptively adjusted according to the current value of IE, which is helpful in guiding the search to more promising regions. Moreover, a mutation mechanism for the probability model is developed to drive the search to quite different regions. In addition, to enhance the local exploitation ability of SHPBIL, a local search based on critical path is presented to execute the search in some narrow and promising search regions. Simulation experiments and comparisons demonstrate the effectiveness of the proposed SHPBIL.

Keywords: Population-based incremental learning algorithm, *m*-machine reentrant permutation flow-shop scheduling, information entropy, self-adaptive strategy, critical-path-based local search.

1 Introduction

This paper deals with the *m*-machine reentrant permutation flow-shop scheduling problem (MRPFSSP) with makespan criterion, which represents a typical subsystem in some semiconductor manufacturing system [1]. In MRPFSSP with n jobs and m machines, each job should be processed on machine $1, 2, ..., m, 1, 2, ..., m, ..., 1, 2, ...m$, and the job sequence is the same for each machine. It has been proved that the MRPFSSP is NP-hard, even for the two-machine case [2]. Choi et al. [1] presented several types of heuristic algorithms for the MRPFSSP to minimize makespan.

* Corresponding author.

Chen et al. [3] proposed a hybrid genetic algorithm (HGA) for the reentrant flow-shop scheduling problem (RFSSP) with makespan criterion. Chen et al. [4] developed a HGA to obtain approximate optimal solutions for the reentrant permutation flow-shop scheduling problem (RPFSSP) with makespan criterion. RPFSSP can be classified as a special case of MRPFSSP, in which the sequence of jobs is the same for each machine at each level. Therefore, some approaches for RPFSSP also can be expanded to address MRPFSSP.

As a novel probabilistic-model based evolutionary algorithm, population-based incremental learning algorithm (PBIL) was introduced by Baluja [5] for solving traveling salesman problem (TSP) and job-shop scheduling problem. PBIL uses a variable-independence probability model to generate new population and guide the search direction, which is the earliest model of estimation of distribution algorithm (EDA). The evolution process of PBIL is regarded as a process of competitive learning, and its probability model is updated by the current best solution at each generation to accumulate the information of excellent individuals. Due to its simple framework and outstanding global exploration ability, PBIL has attracted much attention and has been used to solve some production scheduling problems. Salhi et al. [6] proposed a PBIL to deal with hybrid flow-shop scheduling problem. Wang et al. [7] developed a hybrid PBIL for flexible job-shop scheduling problem, whose local search scheme is designed based on the critical path. Pang et al. [8] presented an adaptive PBIL algorithm for solving job-shop and flow-shop scheduling problem, which can obtain satisfied solutions.

In this paper, we propose a self-adaptive hybrid PBIL (SHPBIL) for MRPFSSP. Firstly, the information entropy (IE) of the initial population and an *Interchange*-based search are utilized to guarantee a good distribution of the initial population in the solution space, and a training strategy is designed to help the probability matrix to accumulate information from the initial population. Secondly, in SHPBIL's global exploration, the IE of the probability matrix at each generation is used to evaluate the evolutionary level, and then the learning rate is adaptively adjusted according to the current value of IE, which is helpful in guiding the search to more promising regions. Thirdly, a mutation mechanism for the probability model is developed to drive the search to quite different regions. Fourthly, to enhance the local exploitation ability of SHPBIL, a local search based on critical path is presented to execute the search in some narrow and promising search regions. Due to the hybridization of global search and local search, MRPFSSP can be solved effectively. Simulation experiments and comparisons show the effectiveness and robustness of the proposed SHPBIL.

The remainder of this paper is organized as follows. In the next section, the MRPFSSP is stated briefly. In section 3, SHPBIL is presented and described in detail. In section 4, simulation result and comparisons are given. Finally, we end the paper with some conclusion and future work in section 5.

2 Problem Description

The problem size is denoted by $n \times m \times L$ where n is the number of jobs, m is the number of machines, and L means the repeat or reentrant times. The MRPFSSP is usually defined as follows. Each job should be processed on machine $1, 2, ..., m, 1, 2, ..., m, ..., 1, 2, ...m$. In this case, every job can be decomposed into several layers and each layer starts on 1 and finishes on. That is, each job should visits every machine L (L>1) times. The job ordering is the same on all machines. No preemption is allowed. Once an operation is started, it must be completed without any interruption. At any time, each machine can process at most one job.

Denote $\pi = [\pi_1, \pi_2, ..., \pi_{n\times L}]$ the sequence or schedule of jobs to be processed, π_j ($j \in \{1, ..., n\times L\}$) the jth job in π, $l(\pi_j)$ the repeat or reentrant times of job π_j in $[\pi_1, ..., \pi_j]$, $p(\pi_j, k, l(\pi_j))$ the $l(\pi_j)$th processing time of job π_j on machine k, $C(\pi_j, k, l(\pi_j))$ the $l(\pi_j)$th completion time of job π_j on machine k. Let $C(\pi_0, k, l(\pi_0)) = 0$ and $C(\pi_j, k, 0) = 0$ for $k \in \{1, ..., m\}$ and $j \in \{1, ..., n\times L\}$. Then $C(\pi_j, k, l(\pi_j))$ can be calculated as follows:

$$\begin{aligned} C(\pi_j, 1, l(\pi_j)) = \max\{&C(\pi_{j-1}, 1, l(\pi_{j-1})), C(\pi_j, m, l(\pi_j) - 1)\} \\ &+ p(\pi_j, 1, l(\pi_j)),\ j = 1, ..., n\times L, \end{aligned} \tag{1}$$

$$\begin{aligned} C(\pi_j, k, l(\pi_j)) = \max\{&C(\pi_{j-1}, k, l(\pi_{j-1})), C(\pi_j, k-1, l(\pi_j))\} \\ &+ p(\pi_j, k, l(\pi_j)), j = 1, ..., n\times L, k = 2, ..., m. \end{aligned} \tag{2}$$

Thus, the makespan can be defined as

$$C_{\max}(\pi) = C(\pi_{n\times L}, m, l(\pi_{n\times L})) \, . \tag{3}$$

The MRPFSSP with the makespan criterion is to find a schedule $\boldsymbol{\pi}^*$ in the set of all schedules Π such that

$$\pi^* = \arg\{C_{\max}(\pi)\} \to \min\ , \ \forall \pi \in \Pi \, . \tag{4}$$

Obviously, the size of Π is $(n\times L)!/[L!]^n$.

3 SHPBIL

3.1 Solution Representation

Based on the properties of MRPFSSP, we adopt the operation-based solution representation, that is, every individual of the population is a feasible solution of the MRPFSSP, for example, $[\pi_1, \pi_2, ..., \pi_{4\times 2}] = [1, 2, 4, 2, 1, 3, 3, 4]$ is an individual when the problem's scale $n\times m\times L$ is set to $4\times 3\times 2$.

3.2 Probability Model and Updating Mechanism

Different from other evolutionary algorithms, PBIL generates new population by sampling a probability model (i.e., probability matrix). Hence, the probability model has a key effect on the performances of the PBIL. In this study, the probability matrix is defined as follows:

$$P_{matrix}(gen) = \begin{pmatrix} P_{11}(gen) & \dots & P_{1C}(gen) \\ \vdots & \ddots & \vdots \\ P_{n1}(gen) & \cdots & P_{nC}(gen) \end{pmatrix}_{n\times C}, \quad (5)$$

where $C = n \times L$, $\sum_{w=1}^{n} P_{wj}(gen) = 1$, and $P_{wj}(gen)$ is the probability of job w appearing in the jth position of π at generation gen.

Let $\boldsymbol{\pi}_i(gen)$ = [$\pi_{i1}(gen), \pi_{i2}(gen), ..., \pi_{iC}(gen)$] denote the ith individual in SHPBIL's population at generation gen, and $\alpha(gen)$ the learning rate at generation gen. The matrix $P_{matrix}(gen+1)$ is updated according to $\boldsymbol{\pi}_i(gen)$ by the following two steps:

Step 1: Set $x = \pi_{ij}(gen)$ and $p_{xj}(gen+1) = p_{xj}(gen) + \alpha(gen)$ for $j = 1,...,$ $n \times L$.

Step 2: Set $p_{wj}(gen+1) = p_{wj}(gen+1) / \sum_{y=1}^{n} p_{yj}(gen+1)$ for $w = 1,...,n$ and $j =$ $1,...,n \times L$.

3.3 New Population Generation Method

In each generation of the SHPBIL, the new individuals are generated by sampling the probability matrix mentioned in 3.2. Denote PS the size of the population and $SelectJob(\boldsymbol{\pi}_i(gen+1), j)$ the function of selecting a job in the jth position of $\pi_i(gen+1)$ by using the matrix $P_{matrix}(gen)$. The procedure of $SelectJob(\pi_i(gen+1),$ $j)$ is described as follows:

Step 1: Randomly create a probability r where $r \sim [0,1)$.

Step 2: Get a candidate job CJ by the roulette-wheel selection scheme.

Step 2.1: If $r \sim [0, p_{1j}(gen))$, then set $CJ = 1$ and go to Step 3.

Step 2.2: If $r \sim [\sum_{y=1}^{w} p_{yj}(gen), \sum_{y=1}^{w+1} p_{yj}(gen))$ and $w \in \{1,\cdots,n-1\}$, then set $CJ = w+1$ and go to Step 3.

Step 3: Return CJ.

Let $l(CJ, \boldsymbol{\pi}_i(gen+1))$ denote the repeat times of CJ in $\boldsymbol{\pi}_i(gen+1)$. Then, the new population generation method is given as the following steps:

Step 1: Set $i = 1$.

Step 2: Generate a new individual $\boldsymbol{\pi}_i(gen+1)$.

Step 2.1: Set $\pi_{ij}(gen+1)$ =0 for $j = 1,\cdots,n$.

Step 2.2: Set $j = 1$.

```
    Step 2.3: $CJ = SelectJob(\pi_i(gen+1), j)$.
    Step 2.4: If $l(CJ, \boldsymbol{\pi}_i(gen+1)) = L$, then go to Step 2.3.
    Step 2.5: Set $\pi_{ij}(gen+1) = CJ$ .
    Step 2.6: Set $j = j+1$.
    Step 2.7: If $j \le n \times L$, then go to Step 2.3.
Step 3: Set $i = i+1$.
Step 4: If $i \le PS$ , then go to Step 2.
```

3.4 Population Initialization

The SHPBIL produces the initial population by using information entropy [9] and *Interchange*-based neighborhood search. Firstly, we generate the population based on information entropy theory to guarantee the initial population has a better distribution in the solution space. Secondly, since *Interchange* is a simple and effective neighborhood search in the existing literatures, an *Interchange*-based search can be utilized to enhance the quality of individuals in the population. According to our previous tests, we found that the initial population generated by our method could avoid both "super" and "similar" individuals effectively and the performance of our method was better than random-based or heuristic-based method.

Denote $pop(gen)$ the population at generation gen, $IE_j(gen)$ the information entropy of the jth position of all individuals in $pop(gen)$, $IE\_pop(gen)$ the information entropy of $pop(gen)$, and IE_0_pop the threshold of $IE\_pop(gen)$. $IE_j(gen)$ can be calculated by directly using the method in [9]. $IE\_pop(gen)$ can be expressed as:

$$IE\_pop(gen) = \sum_{j=1}^{n\times L} IE_j(gen) / (n \times L). \tag{6}$$

We generate the initial population $pop(0)$ by the following steps:

```
Step 1: Set $IE_0\_pop$ to a certain value.
Step 2: Set $pop(0) = null$.
Step 3: Generate the first individual $\boldsymbol{\pi}_1(0)$ randomly and add it to $pop(0)$.
Step 4: $i = 2$.
    Step 4.1: Generate an individual $\boldsymbol{\pi}_i(0)$ randomly and add it to $pop(0)$.
    Step 4.2: Calculate $IE\_pop(0)$.
    Step 4.3: If $IE\_pop(0) \le IE_0\_pop$, then delete $\boldsymbol{\pi}_i(0)$ from $pop(0)$.
    Step 4.4: $i = i+1$.
    Step 4.5: If $i \le PS$ , then go to Step 4.1.
Step 5: Output $pop(0)$.
```

Denote $Interchange(\pi_i(gen),u,v)$ the interchange of the job at the *u*th position (i.e., $\pi_{iu}(gen)$) and the job at the *v*th position (i.e., $\pi_{iv}(gen)$). Then, the final $pop(0)$ can be obtained by using the *Interchange*-based neighborhood search as follows:

```
Step 0: Set $i$=1.
Step 1: Randomly choose two positions $u$ and $v$, where $\pi_{iu}(0) \neq \pi_{iv}(0)$. Then, set
        $\pi_i^1(0) = Interchange(\pi_i(0),u,v)$.
Step 2: Set $LOOP = 1$.
    Step 2.1: Randomly choose two positions $u$ and $v$, where $\pi_{iu}^{\ 1}(0) \neq \pi_{iv}^{\ 1}(0)$.
              Then, set $\pi_i^2(0) = Interchange(\pi_i^1(0),u,v)$.
    Step 2.2: If $f(\pi_i^2(0)) < f(\pi_i^1(0))$, then set $\pi_i^1(0) = \pi_i^2(0)$.
    Step 2.3: $LOOP = LOOP + 1$.
    Step 2.4: If $LOOP < n \times L \times (n \times L - 1)/2$, then go to Step 2.1.
Step 3: If $f(\pi_i^1(0)) < f(\pi_i(0))$, then set $\pi_i(0) = \pi_i^1(0)$.
Step 4: Set $i$=$i$+1. If $i \leq PS$, then goto Step 1.
Step 5: Output the final $pop(0)$.
```

3.5 Probability Matrix Training

The population initialization strategy in the above subsection can guarantee the initial population has a better distribution in the solution space. However, based on our previous tests, the PBIL algorithm with the proposed population initialization strategy was still likely to fall into low-quality local optima at the start phase. This phenomenon shows the fact that the probability matrix $P_{matrix}(gen)$ does not have enough historical information to track the relatively high-quality individuals in the initial population. Therefore, an information-entropy-based probability matrix training method is designed to enhance PBIL's search ability. Denote IE_{tra_ini} the initial training information entropy, $IE_{tra}\_P_{matrix}(0)$ the information entropy of $P_{matrix}(0)$, K_{tra} ($K_{tra} < 1$) the training constant, IE_{tra_0} the threshold of $IE_{tra}\_P_{matrix}(0)$, and $\pi^{lbest}(0) = [\pi_1^{lbest}(0),...,\pi_{n\times L}^{lbest}(0)]$ the best individual of the initial population $pop(0)$. Based on the method in [8], $IE_{tra}\_P_{matrix}(0)$ can be calculated by the following formulation:

$$IE_{tra}\_P_{matrix}(0) = -\sum_{i=1}^{n} \sum_{j=1}^{n\times L} P_{ij}(0) \times \ln(P_{ij}(0)) . \tag{7}$$

The procedure of probability matrix training can be expressed as follows:

```
Step 0: Set $IE_{tra\_ini} = n \times L \times \ln(n)$.
Step 1: Set $IE_{tra\_0} = K_{tra} \times IE_{tra\_ini}$.
Step 2: Train the probability matrix.
```

Step 2.1: Randomly select *u* and *v*, where $\pi_u^{lbest}(0) \neq \pi_v^{lbest}(0)$.

Step 2.2: $\pi_{neighbor}^{lbest}(0) = Interchange(\pi^{lbest}(0), u, v)$.

Step 2.3: Update the probability matrix $P_{matrix}(0)$ according to $\pi_{neighbor}^{lbest}(0)$ by using the updating method in subsection 3.2.

Step 2.4: Calculate $IE_{tra}\_P_{matrix}(0)$ of the current $P_{matrix}(0)$ by using (7).

Step 2.5: If $IE_{tra}\_P_{matrix}(0) > IE_{tra\_0}$, then go to Step 2.1.

Step3: Output a new probability matrix $P_{matrix}(0)$ and the current $IE_{tra}\_P_{matrix}(0)$.

It can be seen from the above procedure that Step 2.1 and Step 2.2 compose a perturbation operator, and Step 2.5 is used to help $P_{matrix}(0)$ to accumulate information from the *Interchange*-based neighbors of the best individual in $pop(0)$. Moreover, the output $IE_{tra}\_P_{matrix}(0)$ is used in the calculation of a threshold IE_0 in the next subsection.

3.6 Self-adaptive Adjusting Strategy

PBIL is a stochastic search algorithm, which guides the search direction by sampling the probability model. The updating mechanism of probability model is a key factor to perform global exploration. However, in the process of probability model updating, learning rate is a sensitive parameter. That is, a small value of learning rate will slow down the convergence speed. On the contrary, a large value of learning rate will cause the population to converge too early. Thus, it is important to choose suitable value for learning rate.

The information entropy is utilized to measure the evolutionary degree. The evolution of PBIL is a process of accumulating the excellent solutions' historical information. That is, during the evolutionary process of PBIL, each $P_{wj}(gen)$ in probability matrix $P_{matrix}(gen)$ changes gradually from initial values to 0 or 1. Let $IE\_P_{matrix}(gen)$ denote the information entropy of $P_{matrix}(gen)$. By using the method in [8], $IE\_P_{matrix}(gen)$ can be expressed as:

$$IE\_P_{matrix}(gen) = -\sum_{i=1}^{n} \sum_{j=1}^{n \times L} P_{ij}(gen) \times \ln(P_{ij}(gen)), \quad gen \geq 0. \tag{8}$$

Obviously, $IE\_P_{matrix}(gen)$ decreases with an increase of gen, and $IE\_P_{matrix}(gen)$ tends to 0 when gen increases to a large value. Denote $IE_0 = k_0 \times IE_{tra}\_P_{matrix}(0)$ the threshold of $IE\_P_{matrix}(gen)$ and α_0 the maximum learning rate. The learning rate $\alpha(gen)$ is adaptively adjusted according to the formula as follows:

$$\alpha(gen) = \begin{cases} \exp[k_1 \times (IE\_P_{matrix}(gen)/IE_0 - 1)] \times \alpha_0 & IE\_P_{matrix}(gen) < IE_0 \\ \beta^{k_2 \times (IE\_Pmatrix(gen)/IE_0 - 1)} \times \alpha_0 & IE\_P_{matrix}(gen) \geq IE_0 \end{cases}, \tag{9}$$

where k_0, k_1, and k_2 are the adjusting parameters and β ($0<\beta<1$) is the base number of exponential function. The self-adaptive adjusting strategy is reflected in formula (9). That is, with the increase of gen, $\alpha(gen)$ is set to a relatively small value to increase the population diversity and track the population at the beginning phase, and it is set to a comparatively large value to speed up the process of accumulating information at the middle phase, and it is set to a small value to enhance the search precision and avoid falling into local optima at the last phase.

3.7 Mutation Mechanism

Each $P_{wj}(gen)$ in probability matrix $P_{matrix}(gen)$ changes gradually from initial values to 0 or 1 when gen increases to a large value. This means the population diversity and exploration ability are decreased with the increase of gen. Inspired by Bajula's work [10], we propose a mutation operator to disturb the probability matrix. Denote $\alpha_m(gen)$ the mutation rate at generation gen and set $\alpha_m(gen)=\alpha(gen)/2$. The process of the proposed mutation mechanism can be described as follows:

Step 1: Randomly select *u*, *v* and *q*, where $u \neq v \neq q$.

Step 2: Randomly select ω_1, ω_2 and ω_3.

Step 3: Set $P_{\omega_1 u}(gen)=P_{\omega_1 u}(gen)+\alpha_m(gen)$, $P_{\omega_2 v}(gen)=P_{\omega_2 v}(gen)+\alpha_m(gen)$, and $P_{\omega_3 q}(gen)=P_{\omega_3 q}(gen)+\alpha_m(gen)$.

Step 4: For $w=1,..,n$, set $P_{wu}(gen)=P_{wu}(gen)/\sum_{y=1}^{n}P_{yu}(gen)$, $P_{wv}(gen)=P_{wv}(gen)/\sum_{y=1}^{n}P_{yv}(gen)$, and $P_{wq}(gen)=P_{wq}(gen)/\sum_{y=1}^{n}P_{yq}(gen)$.

3.8 Critical-Path-Based Local Search

3.8.1 Critical Path and Block

Referring to the work of Grabowski and Wodecki [11], the MRPFSSP can also be described by a graph model, which illustrates the technological constraints for each job and the processing sequence on each machine. The longest path from node $(1,1)$ to $(m, n\times L)$ in the graph is defined as the critical path, and the length of the critical path is equal to $C_{\max}$.

3.8.2 Search Strategy

Grabowski and Wodecki [11] had given a detailed definition of the moves and neighborhood structure for the permutation flow-shop problem, which can avoid invalid moves inside blocks. We extend Grabowski's structure in this subsection, which can also provide some narrow and promising search regions. Denote $Insert(\pi_i,u,v)$ the insertion of the job π_{iu} before π_{iv} when $u>v$ and after π_{iv} when $u<v$,

$B_k = [\pi_k^1, \pi_k^2, ..., \pi_k^{kb}]$ the kth block and kb the length of B_k. Then, we define *Left moves* $M_L(B_k)$ and *Right moves* $M_R(B_k)$ as follows:

$M_L(B_k)$: $Insert(B_k, u, 1)$ for $u = 2, ..., kb$.

$M_R(B_k)$: $Insert(B_k, u, kb)$ for $u = 1, ..., kb-1$.

Denote $\pi^{lbest}(gen) = [\pi_1^{lbest}(gen), \pi_2^{lbest}(gen), ..., \pi_{n\times L}^{lbest}(gen)]$ the local best individual of the current population $pop(gen)$, BL the total number of blocks in $\pi^{lbest}(gen)$, $N^{lbest}(B_k)$ the set of neighbors of $\pi^{lbest}(gen)$ when applying $M_L(B_k)$ and $M_R(B_k)$ to $\pi^{lbest}(gen)$, and $FindBestN^{lbest}(\pi^{lbest}(gen))$ the scanning procedure of finding the best neighbor in $N^{lbest}(B_1) \cup N^{lbest}(B_2) \cup \cdots \cup N^{lbest}(B_{BL})$. The procedure of the critical-path-based local search for $\pi^{lbest}(gen)$ is given as follows:

```
Step 1: Set $\pi^{i\text{-}0} = \pi^{lbest}(gen)$.
Setp 2: $Insert$ -based perturbation.
    Step 2.1: Randomly select $u$ and $v$, where $\pi_u^{lbest}(gen) \neq \pi_v^{lbest}(gen)$ and $u > v$.
    Step 2.2: $\pi^{lbest}(gen) = Insert(\pi^{lbest}(gen), u, v)$.
Step 3: $\pi^{i\text{-}1} = FindBestN^{lbest}(\pi^{lbest}(gen))$.
Step 4: If $f(\pi^{i\text{-}1}) < f(\pi^{i\text{-}0})$, then $\pi^{i\text{-}0} = \pi^{i\text{-}1}$.
Step 5: Output $\pi^{i\text{-}0}$.
```

3.9 Procedure of SHPBIL

Based on the contents in the above subsections, we propose the procedure of SHPBIL as follows:

```
Step 0: Denote $\pi^{gbest}$ the global best individual and $genMax$ the maximum generation.
Step 1: Initialization.
    Step 1.1: Set $gen = 0$.
    Step 1.2: Generate the initial population $pop(0)$ by using the method in subsection 3.4 and set $\pi^{gbest} = \pi^{lbest}(0)$.
    Step 1.3: Set $P_{wj}(gen) = 1/n$ for $w = 1, ..., n$ and $j = 1, ..., n \times L$.
    Step 1.4: Train the probability matrix $P_{matrix}(0)$ by using the method in sub-section 3.5.
Step 2: Set $gen = gen + 1$.
Step 3: Calculate the learning rate $\alpha(gen-1)$ by using the self-adaptive adjusting strategy in subsection 3.6.
```

Step 4: Generate the population $pop(gen)$ by using the new population generationmethod in subsection 3.3, and calculate the makespan of each individual and update $\pi^{lbest}(gen)$.

Step 5: Disturb the probability matrix $P_{matrix}(gen-1)$ by using the mutation mech-anism in subsection 3.7.

Step 6: Apply critical-path-based local search in subsection 3.8 to $\pi^{lbest}(gen)$ and update π^{gbest}.

Step 7: Update the probability matrix $P_{matrix}(gen)$ according to π^{gbest} by using the updating mechanism in subsection 3.2.

Step 8: If $gen < genMax$, then go to Step2.

Step 9: Output π^{gbest}.

It can be seen from the above procedure that Step 7 uses π^{gbest} obtained in Step 6 and the self-adaptive $\alpha(gen-1)$ obtained in Step 3 to update the probability matrix, which means new generated individuals can aptly absorb the information of the global best individual during the evolution process and then guide the search to more promising regions, and Step 5 is the perturbation operator, which can restrain the search from dropping into local optima and drive the search to quite different regions. Moreover, Step 6 performs exploitation from the regions obtained by Step 4. Since both exploration and exploitation are stressed and balanced, SHPBIL is hopeful to obtain good results.

4 Simulation Result and Comparisons

4.1 Experimental Design

In order to test the performance of the proposed SHPBIL, a set of instances under different scales is randomly generated. The $n \times m \times L$ combinations include $10\times5\times3$, $10\times8\times6$, $20\times10\times3$, $20\times12\times5$, $30\times10\times3$, $30\times15\times6$, $40\times10\times2$, $40\times15\times3$, $50\times10\times3$, and $50\times20\times4$. The processing time $p(\pi_j, k, l(\pi_j))$ is generated from a uniform distribution [1, 100]. All algorithms are coded in Delphi7.0 and are executed on Mobile Intel Core 2 Duo 2.0 GHz processor with 2GB memory.

For each instance, each algorithm is run 20 times independently. Based on our previous experiments, the parameters of SHPBIL are set as follows: the population size $PS = 50$, the threshold $IE_0\_pop = 0.2$, the maximum learning rate $\alpha_0 = 0.04$, the adjusting parameters $k_0 = 0.9$, $k_1 = 0.3$ and $k_2 = 0.7$, the base number of exponential function $\beta = 0.25$ and the training constant $K_{tra} = 0.25$.

4.2 Comparisons of HGA, Random+MN3+SO, HGA_V, and SHPBIL

For the purpose of showing the effectiveness of SHPBIL, we compare SHPBIL with a hybrid genetic algorithm (HGA) [4] and a so-called Random+MN3+SO algorithm

[1]. HGA is an effective algorithm for RPFSSP. Random+MN3+SO algorithm is one of the most effective algorithms for MRPFSSP [1]. Moreover, we also compare SHPBIL with HGA_V, which is a variant of an effective HGA for RFSSP [3]. In HGA_V, we generate the first part of initial population randomly. The maximum generations of HGA, HGA_V and SHPBIL are set to 1000, 50000 and 500, respectively. The running time of Random+MN3+SO is decided only by the scale of problem. The simulation results are listed in Table 1 and Table 2, where $BEST$ denotes the best makespan, AVG denotes the average makespan, $WORST$ denotes the worst makespan, T_{avg} denotes the average running time, and SD denotes the standard derivation.

From Table 1 and Table 2, it is shown that the SHPBIL is better than HGA, HGA_V and Random+MN3+SO with respect to solution quality. The values of AVG,

Table 1. Comparisons of *BEST* and *AVG* of HGA, HGA_V, Random+MN3+SO and SHPBIL

Instances	HGA		HGA_V		Random+MN3+SO		SHPBIL		
	BEST	*AVG*	*BEST*	*AVG*	*BEST*	*AVG*	*BEST*	*AVG*	*WORST*
10×5×3	1995	2014.10	1990	2029.45	2186	2323.40	**1968**	**1995.70**	2012
10×8×6	4265	4325.25	4595	4672.00	4986	5367.35	**4247**	**4287.75**	4399
20×10×3	3914	3960.60	**3877**	3933.35	3976	4322.60	3892	**3920.40**	3947
20×12×5	**6808**	**6923.80**	6956	7032.35	7093	7591.15	6842	6963.95	7038
30×10×3	5808	5855.45	**5790**	5843.20	5989	6219.40	5796	**5834.50**	5880
30×15×6	11739	11846.60	**11649**	11846.80	12100	12653.85	11674	**11823.65**	11902
40×10×2	4848	4906.90	4851	4893.55	4850	5013.95	**4793**	**4806.90**	4822
40×15×3	7751	7842.85	7633	7717.50	7764	8019.25	**7592**	**7676.65**	7718
50×10×3	8832	8908.75	8933	8998.10	8788	8930.65	**8689**	**8702.10**	8717
50×20×4	12968	13092.75	12937	13017.45	12903	13446.35	**12855**	**12993.00**	13072
Average	6892.80	6967.71	6921.10	6998.38	7063.50	7388.80	**6834.80**	**6900.46**	6950.70

Table 2. Comparisons of T_{avg} and SD of HGA, HGA_V, Random+MN3+SO and SHPBIL

Prob-	HGA		HGA_V		Random+MN3+SO		SHPBIL	
lems	T_{avg}	*SD*	T_{avg}	*SD*	T_{avg}	*SD*	T_{avg}	*SD*
10×5×3	7.20	**10.31**	18.89	17.94	0.01	95.03	**3.40**	11.27
10×8×6	75.85	44.31	25.51	**34.77**	0.13	264.25	**24.89**	42.36
20×10×3	79.45	18.77	34.58	29.20	0.23	200.31	**32.08**	**17.97**
20×12×5	469.17	53.04	179.67	83.29	1.66	223.28	**156.61**	**47.26**
30×10×3	299.85	29.43	128.10	50.73	1.05	143.00	**107.62**	**25.70**
30×15×6	954.68	89.09	766.40	80.79	16.01	340.50	**683.41**	**59.84**
40×10×2	74.81	28.16	74.16	14.65	0.51	94.01	**69.85**	**8.99**
40×15×3	362.55	60.45	261.33	41.42	4.81	199.69	**215.31**	**34.50**
50×10×3	697.84	40.87	537.64	44.55	4.67	80.72	**455.03**	**8.43**
50×20×4	2058.76	76.48	1301.29	**61.74**	40.65	355.98	**1266.78**	64.41
Average	508.02	45.09	332.76	45.91	6.97	199.68	301.50	**32.07**

BEST and *SD* obtained by SHPBIL are much better than those obtained by HGA, HGA_V and Random+MN3+SO. Moreover, the *WORST* values of SHPBIL are smaller than the *AVG* values of the other compared algorithms for almost all the instance. Thus, SHPBIL is an effective algorithm for the *m*-machine reentrant permutation flow-shop scheduling problem.

5 Conclusion and Future Work

This paper proposed a self-adaptive hybrid population-based incremental learning algorithm (SHPBIL) to solve the *m*-machine reentrant permutation flow-shop scheduling problem (MRPFSSP). In SHPBIL, the initial population was generated by using several presented methods, the global search was performed through the improved PBIL with adaptive learning rate and mutation scheme, and a local search was guided by the critical-path-based neighborhood. Since the search behavior was enriched as well as global exploration and local exploitation were well balanced, MRPFSSP can be solved effectively. Simulation results and comparisons based on a set of randomly-generated instances showed the effectiveness of SHPBIL. Our future work is to develop some PBIL-based algorithms to deal with re-entrant job-shop scheduling problems.

Acknowledgments. This research was partially supported by National Science Foundation of China (No. 60904081) and 2012 Academic and Technical Leader Candidate Project for Young and Middle-Aged Persons of Yunnan Province (No. 2012HB011).

References

1. Choi, S.W., Kim, Y.D.: Minimizing Makespan on an M-machine Re-entrant Flowshop. Computers & Operations Research 35(5), 1684–1696 (2008)
2. Choi, S.W., Kim, Y.D.: Minimizing Makespan on a Two-machine Re-entrant Flowshop. In: Proceedings of the Fifth Asia Pacific Industrial Engineering and Management Systems Conference, vol. 31(19), pp. 1–10 (2004)
3. Chen, J.S., Pan, J.C.H., Lin, C.M.: A Hybrid Genetic Algorithm for the Re-entrant Flowshop Scheduling Problem. Expert Systems with Applications 34(1), 570–577 (2008)
4. Chen, J.S., Pan, J.C.H., Lin, C.M.: Solving the Reentrant Permutation Flow-shop Scheduling Problem with a Hybrid Genetic Algorithm. International Journal of Industrial Engineering 16(1), 23–31 (2009)
5. Baluja, S.: Population-based Incremental Learning: a Method for Integrating Genetic Search based Function Optimization and Competitive Learning. Technical Report CMU-CS-94-193. Carnegie Mellon University, Pittsburgh (1994)
6. Salhi, A., Rodriguez, J.A.V., Zhang, Q.F.: An Estimation of Distribution Algorithm with Guided Mutation for a Complex Flow Shop Scheduling Problem. In: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, London, UK, pp. 570–576 (2007)

7. Wang, S., Wang, L., Zhou, G., Xu, Y.: An Estimation of Distribution Algorithm for the Flexible Job-shop Scheduling Problem. In: Huang, D.-S., Gan, Y., Gupta, P., Gromiha, M.M. (eds.) ICIC 2011. LNCS (LNAI), vol. 6839, pp. 9–16. Springer, Heidelberg (2012)
8. Pang, H., Hu, K., Hong, Z.: Adaptive PBIL Algorithm and Its Application to Solve Scheduling Problems. In: Proceedings of the 2006 IEEE Conference on Computer Aided Control Systems Design, Munich, Germany, pp. 784–789 (2006)
9. Hong, S.K.: Shape Optimization of Electromagnetic Devices Using Immune Algorithm. IEEE Transactions on Magnetics 33(2), 1876–1879 (1997)
10. Baluja, S., Caruana, R.: Removing the Genetics From the Standard Genetic Algorithm. In: Proceeding of the International Conference on Machine Learning, Lake Tahoe, CA, pp. 38–46 (1995)
11. Grabowski, J., Wodecki, M.: A very Fast Tabu Search Algorithm for the Permutation Flow Shop Problem with Makespan Criterion. Computers & Operations Research 31(11), 1891–1909 (2004)