An Enhanced Estimation of Distribution Algorithm for No-Wait Job Shop Scheduling Problem with Makespan Criterion

Shao-Feng Chen[1,2], Bin Qian[1,2,*], Rong Hu[1,2], and Zuo-Cheng Li[1,2]

[1] Department of Automation, Kunming University of Science and Technology, Kunming 650500, China
[2] Key Laboratory of Computer Technologies Application of Yunnan Province, Kunming 650500, China
bin.qian@vip.163.com

Abstract. In this paper, an enhanced estimation of distribution algorithm (EEDA) is proposed for the no-wait job shop scheduling problem (NWJSSP) with the makespan criterion, which has been proved to be strongly NP-hard. The NWJSSP can be decomposed into the sequencing and the timetabling problems. The proposed EEDA and a shift timetabling method are used to address the sequencing problem and the timetabling problem, respectively. In EEDA, the EDA-based search is applied to guiding the search to some promising sequences or regions, and an Interchange-based local search is presented to perform the search from these promising regions. Moreover, each individual or sequence of EEDA is decoded by applying a shift timetabling method to solving the corresponding timetabling problem. The experimental results show that the combination of the EEDA and the shift timetabling method can accelerate the convergence speed and is helpful in achieving more competitive results.

Keywords: No-wait job shop scheduling problem, sequencing, timetabling, enhanced estimation of distribution algorithm.

1 Introduction

This paper considers the no-wait job shop scheduling problem (NWJSSP) with the makespan criterion. The no-wait job shop is a job shop with the constraint that no waiting time is allowed between operations within any job. The no-wait constraints usually arise from requirements for processing environments or characteristics of jobs, such as steel production [1], computer systems [2], food processing [3], chemical industry [4], pharmaceutical industry, production of concrete wares [5] and semiconductor testing facilities [6].

The classical job shop scheduling problem is well-known to be strongly NP-hard [7], and it has been subject to intensive research during the past decades. In contrast, there are considerably fewer contributions dealing with the NWJSSP, which is also

* Corresponding author.

D.-S. Huang et al. (Eds.): ICIC 2014, LNAI 8589, pp. 675–685, 2014.

known to be NP-hard in the strong sense. The review of NWJSSP given by Hall and Sriskandarajah [8] showed that the considered problem is very difficult, especially for the large-size instances. The NWJSSP was proven to be NP-hard in the strong sense even for two-machine cases [9]. Thus, it is meaningful and practical to develop an effective algorithm for the considered problem.

Recently, a complete local search with limited memory (CLLM) algorithm [10] has been proposed for the NWJSSP to minimize the makespan criterion. CLLM is a recent metaheuristic that makes use of limited memory in a explicit manner. That is, all explored solutions are recorded to avoid the exploration of already visited solutions. The concept is similar to the idea of "tabu search with structured moves" in [11]. CLLM can achieve better results than variable neighborhood search (VNS), hybrid simulated annealing/genetic algorithm (GASA), and complete local search with memory (CLM) algorithm [12] when solving the NWJSSP [10]. CLLM is the best algorithm so far for the considered problem.

As a novel probabilistic-model based evolutionary algorithm, an estimation of distribution algorithm (EDA) was introduced by Baluja [13] for solving traveling the salesman problem (TSP) and job shop scheduling problem. EDA uses a variable-independence probability model to generate new population and guide the search direction. The evolution process of EDA is regarded as a process of competitive learning, whose 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, EDA has attracted much attention and has been used to solve some production scheduling problems. Salhi et al. [14] proposed an EDA to deal with the hybrid flow shop scheduling problem. Wang et al. [15] developed a hybrid EDA for the flexible job shop scheduling problem, whose local search scheme is designed based on the critical path. Zhang et al. [16] proposed an EDA for permutation flow shops to minimize total flowtime. Bai et al. [17] used an improved hybrid differential evolution-estimation of distribution algorithm for the nonlinear programming and the mixed integer nonlinear programming problems in engineering optimization fields. However, to the best of our knowledge, there has no published word addressing the NWJSSP by using the EDA-based algorithm.

In the current paper, an enhanced estimation of distribution algorithm (EEDA) is presented to deal with the NWJSSP. In our EEDA, the EDA-based search is adopted to perform global exploration in the sequence or solution space and guide the whole search to the promising regions/sequences, while an Interchange-based local search is developed to emphasize exploitation from those regions. Moreover, a so-called shift timetabling method is utilized to decode each individual or sequence of EEDA. Test results and comparisons with CLLM demonstrate the effectiveness of the presented EEDA.

The remainder of this paper is organized as follows. Section 2 briefly depicts the NWJSSP. Section 3 introduces the shift timetabling method. Section 4 presents EEDA in details. Section 5 provides simulation results and comparisons. Finally, Section 6 gives some conclusion and future work.

2 NWJSSP Description

2.1 NWJSSP

The NWJSSP can be depicted as follows. There are a set of m machines and a set of n jobs. The processing time of each job on each machine is deterministic. At any time, preemption and interruption are forbidden and each machine can process at most one job. To satisfy the no-wait restriction, each job must be processed without interruptions between consecutive machines. A sequence or schedule of jobs to be processed can be denoted as $\boldsymbol{\pi}=[\pi_1,\pi_2,\cdots,\pi_n]$. A processing of each job in $\boldsymbol{\pi}$ on each machine is called an operation. Each job must be processed in order of its predefined operations.

Let $C_{\max}(\boldsymbol{\pi})$ denote the makespan calculated function. The aim of NWJSSP is to find a sequence $\boldsymbol{\pi}^*$ in the set of all permutations Π such that

$$C_{\max}(\boldsymbol{\pi}^*)=\min_{\boldsymbol{\pi}\in\Pi} C_{\max}(\boldsymbol{\pi}). \tag{1}$$

2.2 Sequence and Timetabling Problems of NWJSSP

Obviously, the NWJSSP can be decomposed into the lower-level and the upper-level problems. The lower-level problem is called the sequencing problem, in which a processing sequence $\boldsymbol{\pi}$ of an optimal schedule is obtained for a given NWJSSP. The upper-level problem is called the timetabling problem, in which a set of the feasible start processing times of jobs in $\boldsymbol{\pi}$ on each machine (i.e., the timetable of the sequence obtained from the sequencing problem) are found to minimize $C_{\max}(\boldsymbol{\pi})$.

3 Shift Timetabling Method

It is a common belief and intuitively quite clear that the major difficulty in solving the NWJSSP lies in the sequencing problem. Existing algorithms spend most computation time on the sequencing problem while employing simple strategies for the timetabling problem [18]. In fact, even though a sequence for the problem is given, different timetabling methods' decoding results for this sequence is quite different. Effectiveness of an algorithm for the NWJSSP also depends greatly on the timetabling methods. In this paper, a shift timetabling method presented in [10] is utilized, which is briefly introduced in this section.

There some notations to be used are given as follows.

n	the number of jobs
m	the number of machines
$k\in\left[2,\lceil n/2\rceil\right]$	the number of sub-sequences

$J = \{J_1, J_2, ..., J_n\}$	the set of jobs
$\boldsymbol{\pi} = [\pi_1, \pi_2, \cdots, \pi_n]$	the sequence or schedule of jobs to be processed
$\pi_i \in \{J_1, J_2, ..., J_n\}$	the *i*th job in $\boldsymbol{\pi}$ and $i = 1, 2, \cdots, n$
$\boldsymbol{\pi}^{\mathbf{j}} = [\pi_{(j-1)*\lfloor n/k \rfloor+1}, \pi_{(j-1)*\lfloor n/k \rfloor+2}, \cdots \pi_{j*\lfloor n/k \rfloor}]$	the *j*th subsequence of $\boldsymbol{\pi}$ and $j = 1, 2, \cdots, k-1$
$\boldsymbol{\pi}^{\mathbf{k}} = [\pi_{(j-1)*\lfloor n/k \rfloor+1}, \pi_{(j-1)*\lfloor n/k \rfloor+2}, \cdots \pi_n]$	the last subsequence of $\boldsymbol{\pi}$ and $j = k$
$len(\boldsymbol{\pi}^{\mathbf{j}})$	the length of $\boldsymbol{\pi}^{\mathbf{j}}$ and $j = 1, 2, \cdots, k$
$\boldsymbol{\pi} = [\boldsymbol{\pi}^{\mathbf{1}}, \boldsymbol{\pi}^{\mathbf{2}}, \cdots, \boldsymbol{\pi}^{\mathbf{k}}]$	the sequence or schedule of jobs to be processed
$\pi_i^j \in \{J_1, J_2, ..., J_{\lfloor n/k \rfloor}\}$	the *i*th job in $\boldsymbol{\pi}^{\mathbf{j}}$
$t_{\pi_i^j}$	the start processing time of *i*th job in $\boldsymbol{\pi}^{\mathbf{j}}$
$L_{\pi_i^j}$	the process time of the *i*th job in $\boldsymbol{\pi}^{\mathbf{j}}$

Based on the non-delay timetabling and enhanced timetabling strategies [18], the shift timetabling method is selected for solving the timetabling problem. Next, a brief description of the shift timetabling method is provided. A more extensive discussion can be found in [10]. Prior to the description of the shift timetabling, two concepts about shift are reviewed as follows.

1. *left shift*: Set the job as early as possible, if there is no conflicting.
2. *right shift*: 1) The start time of the setting job is greater than the time of prior job. 2) Under the premise of satisfying the condition 1), set the job as early as possible, if there is no conflicting.

Then, the shift timetabling method is described as bellows.

Step 1: $t_{\pi_1^1} \leftarrow 0, i \leftarrow 1, C_{\max} \leftarrow 0$. // * *Initialization* * //

Step 2: Divide sequence $\boldsymbol{\pi}$ into *k* sub-sequences $\boldsymbol{\pi}^{\mathbf{1}}, ..., \boldsymbol{\pi}^{\mathbf{k}}$.

Step 3: Repeat

Step 3.1: Perform *left shift* on all jobs in $\boldsymbol{\pi}^{\mathbf{i}}$, and then calculate start times $t'_{\pi_1^i}, ..., t'_{\pi^i_{len(\boldsymbol{\pi}^{\mathbf{i}})}}$. Set the generated schedule as $[\boldsymbol{\pi}^{\mathbf{1}}, \cdots, \boldsymbol{\pi}^{\mathbf{i}}]$ and calculate the makespan of $[\boldsymbol{\pi}^{\mathbf{1}}, \cdots, \boldsymbol{\pi}^{\mathbf{i}}]$ by $C' \leftarrow \max \{ \max_{j=1,...,len(\boldsymbol{\pi}^{\mathbf{i}})} \{ t'_{\pi_j^i} + L_{\pi_j^i} \}, C_{\max} \}$.

Step 3.2: Re-schedule $\boldsymbol{\pi}^{\mathbf{i}}$ by applying *right shift* to the head job in $\boldsymbol{\pi}^{\mathbf{i}}$ while *left shift* to the other jobs in $\boldsymbol{\pi}^{\mathbf{i}}$ sequentially. Calculate the start times $t''_{\pi_1 i},...,t''_{\pi_{len(\boldsymbol{\pi}^{\mathbf{i}})} i}$ and then calculate the corresponding makespan of $[\boldsymbol{\pi}^{\mathbf{1}},\cdots,\boldsymbol{\pi}^{\mathbf{i}}]$ by $C'' \leftarrow \max\{\max_{j=1,...,len(\boldsymbol{\pi}^{\mathbf{i}})}\{t''_{\pi_j i}+L_{\pi_j i}\}, C_{\max}\}$. If $C' > C''$, then $C_{\max} \leftarrow C''$ and $t_{\pi_1 i},...,t_{\pi_{len(\boldsymbol{\pi}^{\mathbf{i}})} i} \leftarrow t''_{\pi_1 i},...,t''_{\pi_{len(\boldsymbol{\pi}^{\mathbf{i}})} i}$, otherwise $C_{\max} \leftarrow C'$ and $t_{\pi_1 i},...,t_{\pi_{len(\boldsymbol{\pi}^{\mathbf{i}})} i} \leftarrow t'_{\pi_1 i},...,t'_{\pi_{len(\boldsymbol{\pi}^{\mathbf{i}})} i}$.

Step 3.4: $i \leftarrow i+1$.

Step 4: Until $i > k$.

Step 5: Set $C_{\max}(\boldsymbol{\pi}) = C_{\max}$.

Step 6: Output $C_{\max}(\boldsymbol{\pi})$.

4 EEDA

In this section, EEDA presented after explaining the solution representation, probability model and updating strategy, new population generation method, *Insert*-based mutation, and *Interchange*-based neighborhood search with first move strategy.

4.1 Solution Representation

Based on the properties of the NWJSSP, we adopt the job-based solution representation, that is, every individual of the population is a feasible solution of the NWJSSP, for example, $[\pi_1,\pi_2,...,\pi_4] = [1,2,4,3]$ is an individual when the problem's scale n is set to 4.

4.2 Probability Model and Updating Mechanism

Different from other evolutionary algorithms, EDA 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 EDA. In this study, the probability matrix is defined as follows:

$$P_{matrix}(gen) = \begin{pmatrix} P_{11}(gen) & \cdots & P_{1n}(gen) \\ \vdots & \ddots & \vdots \\ P_{n1}(gen) & \cdots & P_{nn}(gen) \end{pmatrix}_{n\times n}, \tag{2}$$

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

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

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

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

4.3 New Population Generation Method

In each generation of the EDA, the new individuals are generated by sampling the probability matrix mentioned in 4.2. 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 $j = 1, \cdots, n \times m$.

Step 2.2: Set $j = 1$.

Step 2.3: $CJ = SelectJob(\pi_i(gen+1), j)$.

Step 2.4: If $l(CJ, \boldsymbol{\pi}_i(gen+1)) = 1$, 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 $j \le n \times m$, then go to Step 2.3.

Step 3: Set $i = i+1$.

Step 4: If $i \le PS$, then go to Step 2.

4.4 Insert-Based Mutation

Mutation is an important element of evolutionary algorithm, which is usually used to enhance the diversity of population and prevent the search from falling into local optima. Thus, in our EEDA, the *Insert*-based mutation is utilized to guide the search to more different regions. Denote P_m the mutation probability of each individual and $Insert(\boldsymbol{\pi}_i(gen),u,v)$ the insertion of $\pi_{iu}(gen)$ in the vth dimension of $\boldsymbol{\pi}_i(gen)$, $u \neq v$. The *Insert*-based mutation is described as the following steps:

Step 1: Set $i=1$.
Step 2: Randomly create a probability r where $r \sim [0,1)$.
Step 3: If $r \leq P_m$, then $\boldsymbol{\pi}_i(gen) = Insert(\boldsymbol{\pi}_i(gen),u,v)$.
Step 4: Set $i=i+1$.
Step 5: If $i \leq PS$, then go to Step 2.

4.5 Interchange-Based Neighborhood Search with First Move Strategy

Because *Interchange* is a simple neighborhood in the existing literatures, we use it to execute exploitation in local search. The $Interchange(\boldsymbol{\pi}_i(gen),u,v)$ operation denotes the interchange of the job at the uth dimension and the job at the vth dimension. The *Interchange*-based neighborhood of $\boldsymbol{\pi}_i(gen)$ can be expressed as

$$N_{interchange}(\boldsymbol{\pi}_i(gen)) = \{\, \boldsymbol{\pi}_i(gen)^{n,u,v} = Interchange(\boldsymbol{\pi}_i(gen),u,v) \mid n-1 \text{ and } v = u+1,...,n \,\} \quad (3)$$

Let $FindBestN_{interchange}(\boldsymbol{\pi}_i(gen))$ denote the procedure of obtaining the best $\boldsymbol{\pi}_i(gen)^{n,u,v}$ (denoted by $\boldsymbol{\pi}_i(gen)^{n,u,v}_{best}$) in $N_{interchange}(\boldsymbol{\pi}_i(gen))$. For searching more regions, a first move strategy is applied in $FindBestN_{interchange}(\boldsymbol{\pi}_i(gen))$. That is, when the first $\boldsymbol{\pi}_i(gen)^{n,u,v}$ (denoted by $\boldsymbol{\pi}_i(gen)^{n,u,v}_{firstI}$) in $N_{interchange}(\boldsymbol{\pi}_i(gen))$ that improves $\boldsymbol{\pi}_i(gen)$ is obtained, $FindBestN_{interchange}(\boldsymbol{\pi}_i(gen))$ terminates and outputs $\boldsymbol{\pi}_i(gen)^{n,u,v}_{firstI}$.

Based on the previous tests, we find that applying $FindBestN_{interchange}(\boldsymbol{\pi}_i(gen))$ with the first move strategy to the top 10%-15% individuals of the population can perform a wider and deeper local search and obtain good results. Therefore, the top 10% individuals are chosen to apply the proposed neighborhood search. If $C_{\max}(\boldsymbol{\pi}_i(gen)^{n,u,v}_{best})$ or $C_{\max}(\boldsymbol{\pi}_i(gen)^{n,u,v}_{firstI})$ is less than $C_{\max}(\boldsymbol{\pi}_i(gen))$, $\boldsymbol{\pi}_i(gen)$ will be updated with $\boldsymbol{\pi}_i(gen)^{n,u,v}_{best}$ or $\boldsymbol{\pi}_i(gen)^{n,u,v}_{firstI}$.

4.6 Procedure of EEDA

According to the above subsections, the procedure of EEDA is proposed as follows:

Step 0: Let $genMax$ the maximum generation, $nonICount$ the consecutive non-improved times of π_{gbest}, and $CountMax$ the maximum number of $nonICount$.

Step 1: Initialization.

Step 1.1: Set $gen = 1$ and $nonICount = 0$.

Step 1.2: Randomly generate individual $\pi_i(1)$ for $i = 1,...,PS$, and calculate makespan of each individual by using the ***Shift Timetabling Method***, and update π_{gbest}.

Step 1.4: Set $p_{wj}(gen) = 1/n$ for $w = 1,...,n$ and $j = 1,...,n$.

Step 1.5: Update the matrix $Pmatrix(gen)$ by π_{gbest}.

Step 2: Set $gen = gen + 1$.

Step 3: Generate individual $\pi_i(gen)$ for $i = 1,...,PS$ by new population generation method, and calculate makespan of each individual by using the ***Shift Timetabling Method***, and update π_{gbest}.

Step 4: If π_{gbest} can be updated, then

$nonICount = 0$

Else

$nonICount = nonICount + 1$.

Step 5: If $nonICount < CountMax$, then update the matrix $Pmatrix(gen)$ by π_{gbest} and go to Step 2.

Step 6: Set $nonICount = 0$.

Step 7: Apply *Insert*-based mutation to the current population and calculate makespan of each individual by using the ***Shift Timetabling Method***.

Step 8: Apply *Interchange*-based neighborhood search with first move strategy to the top 10% individuals of the current population, and calculate makespan of each neighbor by using the ***Shift Timetabling Method***.

Step 9: Update π_{gbest}.

Step 10: Update the matrix $Pmatrix(gen)$ by π_{gbest}.

Step 11: If $gen < genMax$ then go to Step 2.

Step 12: Output π_{gbest}.

It can be seen from the above procedure that Step 10 uses π_{gbest} to update the probability matrix, which means new generated individuals can share the information of the global best individual and then guide the search to more promising regions, and Step 7 is the perturbation operator, which can restrain the search from dropping into local optima and drive the search to quite different regions. Moreover, Step 8 performs exploitation from the regions obtained by Steps 3 and 7. Due to well-balanced between exploration and exploitation, EEDA is hopeful to achieve good results.

5 Simulation Result and Comparisons

5.1 Experimental Design

In order to test the performance of the proposed EEDA, the well known job shop benchmarks are tested. The $n \times m$ combinations include 10×5 and 10×10. All algorithms are coded in Delphi 2010 and are executed on Mobile Intel Core 2 Duo 2.2 GHz processor with 8GB memory.

In EEDA, we use the following parameters: the population size $PS = 50$, the learning rate $\alpha = 0.02$, the maximum consecutive non-improved times $CountMax = 20$, the maximum generation $genMax = 200$, the mutation probability $P_m = 0.3$, and the length of $\boldsymbol{\pi}^j$ (i.e., len) is set to 3 and $n - 3 \times (k-1)$ for $j = 1, 2, \cdots, k-1$ and $j = k$, respectively.

5.2 Comparisons of EEDA and CLLM

For the purpose of manifesting the effectiveness of EEDA, we compare EEDA with CLLM [10]. The CLLM method is the best algorithm so far for the considered problem. EEDA and CLLM are compared on some classic benchmarks (i.e., La01- La05, Orb01-Orb10, La16-La20). We set the maximum generation of EEDA=300 and let CLLM run at the same amount of computer time as EEDA. Each benchmark is independently run 20 times for comparison. The simulation results are listed in Table 1, where *BEST* denotes the best makespan, *AVG* denotes the average makespan. *WORST* denotes the worst makespan, and *SD* denotes the standard derivation.

Table 1. Comparisons of *BEST*, *WORS*, *AVG* and *SD* of CLLM and EEDA

Instances			CLLM[10]				EEDA			
	n	*m*	*BEST*	*WORST*	*AVG*	*SD*	*BEST*	*WORST*	*AVG*	*SD*
La01	10	5	975	1128	1057.65	37.58	**971**	**1031**	**991.15**	**24.50**
La02	10	5	988	1041	1009.35	14.80	**961**	**1005**	**978.70**	**11.53**
La03	10	5	862	919	889.55	19.62	**820**	**867**	**860.25**	**10.41**
La04	10	5	917	983	940.15	**12.37**	**887**	**923**	**896.80**	12.81
La05	10	5	784	886	841.67	27.94	**781**	**829**	**791.15**	**13.38**
Orb01	10	10	**1615**	1771	1680.32	40.74	**1615**	**1663**	**1631.60**	**21.30**
Orb02	10	10	1572	1694	1633.73	51.28	**1485**	**1518**	**1516.35**	**7.19**
Orb03	10	10	1603	1745	1678.49	35.18	**1599**	**1685**	**1633.10**	**22.53**
Orb04	10	10	1750	1874	1812.39	30.34	**1653**	**1732**	**1679.25**	**18.17**
Orb05	10	10	1424	1574	1508.80	34.42	**1370**	**1451**	**1402.85**	**20.75**
Orb06	10	10	1616	1716	1665.55	43.75	**1555**	**1557**	**1556.60**	**0.80**
Orb07	10	10	711	797	753.15	33.25	**706**	**725**	**711.45**	**7.74**
Orb08	10	10	**1319**	1452	1353.95	35.90	**1319**	**1350**	**1320.55**	**6.70**
Orb09	10	10	1581	1710	1646.25	41.83	**1445**	**1560**	**1520.80**	**34.24**
Orb10	10	10	1691	1749	1724.95	20.33	**1557**	**1601**	**1582.60**	**9.55**
La16	10	10	1650	1804	1729.77	25.34	**1575**	**1628**	**1591.85**	**10.14**
La17	10	10	1484	1592	1536.84	25.36	**1371**	**1417**	**1387.80**	**11.43**
La18	10	10	1597	1687	1649.34	27.82	**1555**	**1630**	**1589.35**	**16.37**
La19	10	10	1610	1764	1710.91	39.17	**1482**	**1533**	**1508.90**	**7.83**
La20	10	10	1655	1846	1768.10	42.67	**1526**	**1683**	**1616.25**	**23.58**
Average			1370.2	1486.6	1429.5	31.98	**1311.65**	**1369.4**	**1338.36**	**14.54**

From Table 1, it is shown that the EEDA is better than CLLM with respect to solution quality. The values of *BEST*, *AVG* and *SD* obtained by EEDA are much better than those obtained by CLLM. Moreover, the *WORST* values of EEDA are smaller than the *AVG* values of CLLM for all benchmarks except Orb03. Thus, EEDA is an effective algorithm for the NWJSSP.

6 Conclusion and Future Work

To the best of the current authors' knowledge, this is the first report on the application of estimation of distribution algorithm (EDA) for the no-wait job shop scheduling problem (NWJSSP) with the makespan criterion. In our presented enhanced EDA (EEDA), EDA-based global search was used to execute exploration for promising regions within the whole sequence or solution space, while an *Interchange*-based local search was designed to stress exploitation from these promising regions. In addition, a shift timetabling method is selected to decode each individual or sequence of EEDA. Simulations and comparisons based on benchmarks demonstrate the effectiveness of the presented EEDA. Our future work is to develop some EDA-based algorithms to deal with the re-entrant NWJSSP.

Acknowledgments. This research was partially supported by National Science Foundation of China (No. 60904081), 2012 Academic and Technical Leader Candidate Project for Young and Middle-Aged Persons of Yunnan Province (No. 2012HB011), and Discipline Construction Team Project of Kunming University of Science and Technology (No. 14078212).

References

1. Wismer, D.A.: Solution of the Flowshop Scheduling-problem with No Intermediate Queues. Operations Research 20(6), 89–97 (1972)
2. Reddi, S., Ramamoorthy, C.: A Scheduling-problem. Operational Research Quarterly 24(44), 1–6 (1973)
3. Raaymakers, W., Hoogeveen, J.: Scheduling Multipurpose Batch Process Industries with No-wait Restrictions by Simulated Annealing. European Journal of Operational Research 126, 131–151 (2000)
4. Rajendran, C.: A No-wait Flowshop Scheduling Heuristic to Minimize Makespan. Journal of The Operational Research Society 45(4), 472–478 (1994)
5. Grabowski, J., Pempera, J.: Sequencing of Jobs in Some Production System. European Journal of Operational Research 125(5), 35–50 (2000)
6. Ovacik, I., Uzsoy, R.: Decomposition Methods for Complex Factory Scheduling-Problems. Kluwer Academic Publishing, Dordecht (1997)
7. Lenstra, J.K., Rinnooy Kan, A.H.G.: Computational Complexity of Discrete Optimization Problems. Annals of Discrete Mathematics 4(1), 21–40 (1979)
8. Hall, N.G., Sriskandarajah, C.: A survey of machine scheduling problems with blocking and no-wait in process. Operations Research 44, 510–525 (1996)

9. Sahni, S., Cho, Y.: Complexity of Scheduling Shops with No-wait in Process. Mathematics of Operations Research 4, 448–457 (1979)
10. Zhu, J., Li, X.P., Wang, Q.: Complete Local Search with Limited Memory Algorithm for No-wait Job Shops to Minimize Makespan. Eur. J. Oper. Res. 198(2), 378–386 (2009)
11. Glover, F.: Tabu search – Part II. ORSA Journal on Computing 2, 4–32 (1990)
12. Framinan, J.M., Schuster, C.: An Enhanced Timetabling Procedure for The No-wait Job Shop Problem: A complete local search approach. Computers & Operations Research 331, 1200–1213 (2006)
13. 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)
14. 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)
15. Wang, S.Y., 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)
16. Zhang, Y., Li, X.P.: Estimation of Distribution Algorithm for Permutation Flow Shops with Total Flowtime Minimization. Computers & Industrial Engineering 60, 706–718 (2011)
17. Bai, L., Wang, J., Jiang, Y.H., Huang, D.X.: Improved Hybrid Differential Evolution-Estimation of Distribution Algorithm with Feasibility Rules for NLP/MINLP Engineering Optimization Problems. Chinese Journal of Chemical Engineering 20(6), 1074–1080 (2012)
18. Pan, J.C.H., Huang, H.C.: A Hybrid Genetic Algorithm for No-wait Job Shop Scheduling Problems. Expert Systems with Applications 36, 5800–5806 (2009)