A Bayesian Statistical Inference-Based Estimation of Distribution Algorithm for the Re-entrant Job-Shop Scheduling Problem with Sequence-Dependent Setup Times

Shao-Feng Chen[1,2], Bin Qian[1,2,*], Bo Liu[3], Rong Hu[1,2], and Chang-Sheng Zhang[1]

[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
[3] Academy of Mathematics and Systems Science Chinese Academy of Sciences, Beijing 100190, China
bin.qian@vip.163.com

Abstract. In this paper, a bayesian statistical inference-based estimation of distribution algorithm (BEDA) is proposed for the re-entrant job-shop scheduling problem with sequence-dependent setup times (RJSSPST) to minimize the maximum completion time (i.e., makespan), which is a typical NP hard combinatorial problem with strong engineering background. Bayesian statistical inference (BSI) is utilized to extract sub-sequence information from high quality individuals of the current population and determine the parameters of BEDA's probabilistic model (BEDA_PM). In the proposed BEDA, BEDA_PM is used to generate new population and guide the search to find promising sequences or regions in the solution space. Simulation experiments and comparisons demonstrate the effectiveness of the proposed BEDA.

Keywords: bayesian statistical inference, estimation of distribution algorithm, re-entrant job-shop scheduling problem, setup times.

1 Introduction

This paper presents a meta-heuristic for the re-entrant job-shop scheduling problem with sequence-dependent setup times (RJSSPST). The criterion is to minimize the maximum completion time (i.e., makespan). The RJSSPST is a general job-shop scheduling problem characterized by re-entrant work flows and sequence-dependent setup times. This problem extends the classical job-shop scheduling problem (JSSP) by allowing for sequence-dependent setup times between adjacent operations on any machine and relaxing the restriction that each job visits a machine only once. The JSSP is well-known to be NP-complete [1]. From the point of view of computational complexity, the JSSP obviously reduces to the RJSSPST, which means the RJSSPST is also NP-complete. Moreover, the RJSSPST is commonly encountered in today's

* Corresponding author.

D.-S. Huang et al. (Eds.): ICIC 2014, LNAI 8589, pp. 686–696, 2014.

manufacturing environment. Thus, it is meaningful and practical to make the work of developing effective scheduling algorithms for the considered problem.

Although the RJSSPST appears frequently in commercial printing, plastic manufacturing, metal processing, cylinder-head manufacturing, side frame press shop, and semiconductor manufacturing, but mainly because of its complexity, it has received little attention from researchers. Recently, Sun [2] proposed a genetic algorithm (GA), a modified version of the nearest-setup heuristic [3], and a modified version of the shifting-bottleneck heuristic [4] to solve the RJSSPST. The test results show that GA performs better than the other algorithms.

In recent years, a novel probabilistic-model based evolutionary algorithm, i.e., the estimation of distribution algorithm (EDA), was first introduced by Baluja [5] for addressing the traveling salesman problem and the JSSP. EDA adopts a bran-new evolution scheme to drive the evolution process, which has no traditional crossover and mutation operations in the algorithm. The evolution process of EDA can be regarded as a process of competitive learning, and its probabilistic model is updated by the better solutions 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. [6] presented an EDA for the complex flow shop scheduling problem. Wang et al. [7] designed an EDA to deal with the flexible JSSP. Liu et al. [8] used a combination of particle swarm optimization with EDA for the permutation flowshop scheduling problem. Wang et al. [9] developed a bi-population based EDA for the flexible JSSP. Zhou et al. [10] applied a decomposition-based EDA for the multiobjective traveling salesman problems.

However, there are some shortcomings in the current EDA's model. As shown in Ruiz et al. [11], there are many similar blocks of jobs within the individuals' sequences in the latter stages of evolutionary methods. These similar blocks may occupy the same positions or different positions. If these blocks are disrupted, the algorithm has a high probabilistic to produce offspring with worse makespan values. In this paper, a bayesian statistical inference-based estimation of distribution algorithm (BEDA) is proposed. The purpose of bayesian statistical inference (BSI) is to describe the condition probabilistic of each decision variable [12]. And the new model of conditional probabilistic is built based on the frequencies of sub-sequences or neighbor operations in the high quality individuals. By combining the BSI with the EDA, the proposed BEDA can overcome the shortcomings in the current EDA's model to some extent.

The rest of this paper is arranged as follows. In section 2, the RJSSPST is briefly introduced. In section 3, the BEDA is proposed and described in details. In section 4, computational results and comparisons are provided. Finally, we end the paper with some conclusions and future work in section 5.

2 RJSSPST

2.1 Mathematical Model of RJSSPST

The following describes the mathematical model for the RJSSPST. For the purpose of simplification, in the following model, each operation of each job is denoted by a unique notation.

i, j	Notations used for operations
B	a large enough positive number
D	The set of operations in the shop floor
M	The set of machines available in the shop floor
L	The repeat or reentrant times
M_j	The specific machine that operation *j* requires according to its process route
t_j	Standard processing time of operation *j*
C_j	Completion time of operation *j*
F_j	the start time of operation *j*
R_k	The dummy operation that describes the first activity in machine *k*
S_{ij}	The setup time between operation *i* and operation *j* if operation *i* performs just before operation *j*
x_{ijk}	The indicator variable

With the above notations, the RJSSPST can be formulated as a mixed integer programming problem as follows. Let

$$x_{ijk} = \begin{cases} 1, & \textit{if operation } j \textit{ is just after operation } i \textit{ in the machine } k \\ 0, & \textit{otherwise} \end{cases}$$

Then,

$$\textit{Min makespan} = \max(C_i = F_i + t_i) \quad i \in D \tag{1}$$

s.t.

$$\sum_{i \in D} x_{ijk} = L \tag{2}$$

$$\sum_{j \in D} x_{ijk} = L \tag{3}$$

$$F_i + t_i + S_{ij} \le F_j + B(1 - X_{ijk}) \tag{4}$$

$$F_i + t_i \le F_j \qquad i, j \in S_{ij} \tag{5}$$

$$t_{R_k} = 0 \qquad k \in M \tag{6}$$

$$S_{R_k i} = 0, S_{iR_k} = 0, \qquad i \in D, (i, k \mid k = M_i) \tag{7}$$

Equation 1 presents the objective function, which aims to reduce the cycle time of all jobs called makespan. Constraint 2 and constraint 3 force the job scheduling to have a available sequence in a schedule sequence of each machine. Constraint 4 requires that the start time of each operation in one machine should be larger than the completion time of the operation that performs just before this operation considering the setup times between the pervious operation and current operation. Constraint 5 ensures that an operation could not start until its preceding operation is done. In constraint 6 and constraint 7, processing time and the relationship of setup times between dummy operations and other operations, which require the same machine, are set to zero. The existence of variable F_i in the mathematical model eliminates the subtours in the solutions. It should be noted that the operation, which starts in the next location after dummy operation of a machine, is the starting point of sequence and consequently no setup is required to perform this operation.

2.2 Statement of RJSSPST

The above mathematical model is only used to depict the characteristics of RJSSPST. The following statement is utilized to code and decode each sequence or individual in the proposed BEDA. The RJSSPST has the following usual assumptions. There are *m* machines, *n* jobs and *L* times re-entrant in the production system. The problem size can be denoted by $n \times m \times L$. Each job consists of a set of operations that have to be processed in a specified sequence. Each operation has to be processed on a definite machine and has a processing time and a setup time which are deterministically known. Moreover, each job should be processed on each machine *L* times and the setup times of operations on each machine are sequence dependent. Once an operation is started, it must be completed without any interruption. At any time, each machine can process at most one job.

Denote $\boldsymbol{\pi} = [\pi_1, \pi_2, ..., \pi_{n \times m \times L}]$ the sequence or schedule of operations to be processed. The RJSSPST with the makespan criterion is to find a schedule $\boldsymbol{\pi}^*$ in the set of all schedules Π such that

$$\boldsymbol{\pi}^* = \arg(C_{\max}) \to \min , \ \forall \boldsymbol{\pi} \in \Pi . \tag{8}$$

3 BEDA for RJSSPST

In this section, we will present BEDA after explaining the solution representation, decoding scheme, population initialization, probabilistic model and updating mechanism, new population generation method and new probabilistic model construction strategy.

3.1 Solution Representation

Based on the properties of RJSSPST, we adopt the operation-based solution representation, that is, every individual of the population is a feasible solution of the

RJSSPST, for example, $[\pi_1, \pi_2, ..., \pi_{3\times2\times2}] = [1,2,1,3,2,1,3,3,1,2,3,2]$ is an individual when the problem's scale $n \times m \times L$ is set to $3 \times 2 \times 2$.

3.2 Decoding Scheme

Due to the constraints of RJSSPST, we can improve the solution to minimize the makespan criterion by converting the semi-active schedule to the active one. Thus, when decoding we check the possible idle interval before appending an operation at the last position, and fill the first idle interval before the last operation (called active decoding). The details of active decoding scheme used in this paper are very similar to that used in [13]. The difference between the two schemes only lies in the treatment of the setup times. In [13], the setup times of each operation are not considered.

3.3 Population Initialization

In order to search different regions in the huge solution space, we need to ensure that the initial individuals have widely distributed in the space. Therefore, BEDA adopts stochastic method to generate the initial population.

3.4 Probabilistic Model and Its Updating Mechanism

3.4.1 Probabilistic Model

A bayesian network in [14] is used as the probabilistic model of BEDA, whose structure (including nodes and directed arcs in the network) is partially predetermined. This probabilistic model can describe the probabilistic distribution of favorable values based on the high quality individuals in population. The operations in each high quality individual can be represented by a directed acyclic network as shown in Fig. 1.

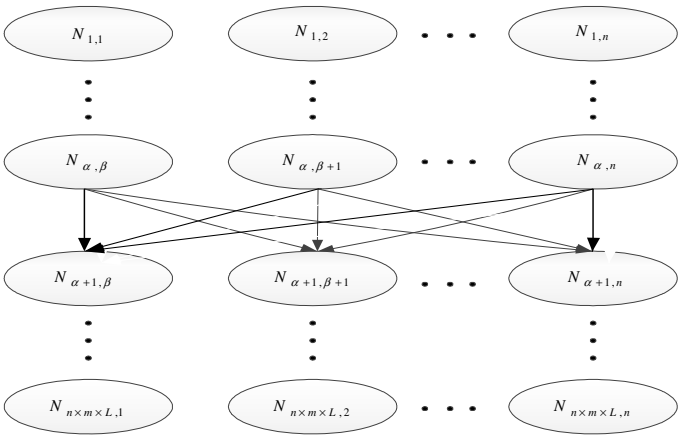

Fig. 1. The structure of bayesian network

The directed arc from node $N_{\alpha,\beta}$ to node $N_{\alpha+1,\beta'}$ ($\beta' \in 1,...,n$) represents the dependent relationship between the two nodes, which records the sub-sequence information of the high quality individuals.

3.4.2 Updating Mechanism

The probabilistic model is a most important part for the BEDA, because it is built for guiding generation new population. Building or updating the probabilistic model is equivalent to determining the bayesian network's parameters (i.e., the weights of directed arcs) according to the high quality individuals. After that, each individual of the next generation can be produced by iteratively sampling the probabilistic model from the first layer (i.e., $N_{1,\beta}$) to the last layer (i.e., $N_{n\times m\times L,\beta}$), which is useful for obtaining promising offspring.

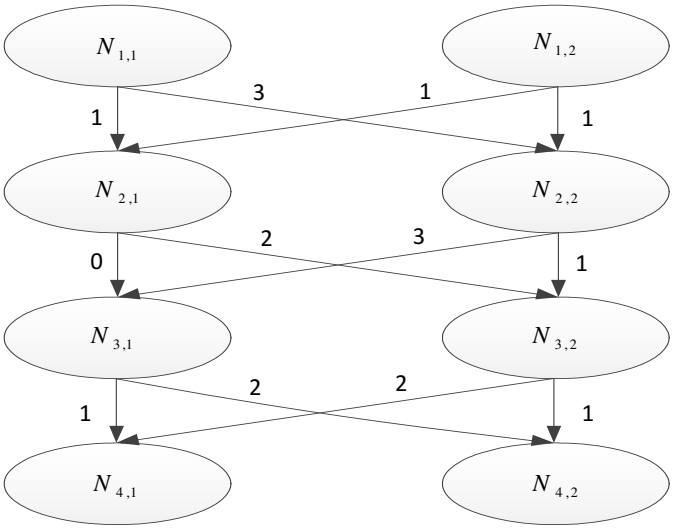

Fig. 2. A example network

The weights of directed arcs can be obtained simply by calculating the frequencies of sub-sequences or neighbor operations in the high quality individuals. Here we provide a concrete example to illustrate the calculation process in Fig. 2. In this example, $n\times m\times L$ is set to $2\times 2\times 1$, the high quality individuals are set as $\boldsymbol{\pi}_1^{\text{high\_quality}} = [1,2,1,2]$, $\boldsymbol{\pi}_2^{\text{high\_quality}} = [1,2,1,2]$, $\boldsymbol{\pi}_3^{\text{high\_quality}} = [1,2,2,1]$, $\boldsymbol{\pi}_4^{\text{high\_quality}} = [2,1,2,1]$, $\boldsymbol{\pi}_5^{\text{high\_quality}} = [1,1,2,2]$, $\boldsymbol{\pi}_6^{\text{high\_quality}} = [2,2,1,1]$. Then, the weight of each directed arc (i.e., the number placed on the arc) can be updated.

By using the weights of directed arcs as approximation probabilities, the conditional probabilities of all the near nodes can be calculated as follows:

$P(N_{1,1}) = (1+3)/6$, $P(N_{1,2}) = (1+1)/6$,

$P(N_{2,1} \mid N_{1,1}) = 1/(1+3)$, $P(N_{2,2} \mid N_{1,1}) = 3/(1+3)$,

$P(N_{2,1} \mid N_{1,2}) = 1/(1+1)$, $P(N_{2,2} \mid N_{1,2}) = 1/(1+1)$,

$P(N_{3,1} \mid N_{2,1}) = 0/(0+2)$, $P(N_{3,2} \mid N_{2,1}) = 2/(0+2)$,

$P(N_{3,1} \mid N_{2,2}) = 3/(3+1)$, $P(N_{3,2} \mid N_{2,2}) = 1/(3+1)$,

$P(N_{4,1} \mid N_{3,1}) = 1/(1+2)$, $P(N_{4,2} \mid N_{3,1}) = 2/(1+2)$,

$P(N_{4,1} \mid N_{3,2}) = 2/(2+1)$, $P(N_{4,2} \mid N_{3,2}) = 1/(2+1)$.

Obviously, if a weight of directed arc is equal to 0, it should be deleted from the bayesian network. This means a corresponding sub-sequence (i.e., two successive operations connected by this arc) can never be obtained in the iterative sampling process. For the purpose of keeping the diversity and dispersivity of the population to some extent, we replace each number 0 on the arc to 1. Therefore, the above conditional probabilities with number 0 on the arc should be changed as follows:

$P(N_{3,1} \mid N_{2,1}) = 1/(1+2)$, $P(N_{3,2} \mid N_{2,1}) = 2/(1+2)$.

3.5 New Population Generation Method

In each generation of the algorithm, the new individuals are generated by sampling the conditional probabilistic matrix mentioned in 3.4.2. Denote PS the size of the population, π_j ($j \in \{1,...,n \times m \times L\}$) the jth operation in $\boldsymbol{\pi}$, $pop(gen)$ the population at generation gen, $\boldsymbol{\pi}^{\mathbf{best}}(\mathbf{gen})$ the best individual of $pop(gen)$, $\boldsymbol{\pi}_{\mathbf{i}}(\mathbf{gen}) = \{\pi_{i1}(gen), \pi_{i2}(gen), ..., \pi_{i(n \times m \times L)}(gen)\}$ the ith individual in $pop(gen)$, and $Job(\boldsymbol{\pi}_{\mathbf{i}}(\mathbf{gen}), j)$ the function of selecting a job in the jth position of $\boldsymbol{\pi}_{\mathbf{i}}(\mathbf{gen})$ by using the conditional probabilistic matrix. Without loss of generality, we assume that $P(N_{1,\beta} \mid N_{0,\beta'}) = P(N_{1,\beta})$ ($\beta, \beta' \in 1,...,n$). The procedure of $Job(\boldsymbol{\pi}_{\mathbf{i}}(\mathbf{gen}), j)$ is described as follows:

Step 1: Set $\alpha = j$, $\beta' = \pi_{i(j-1)}(gen)$.

Step 2: Randomly create a probabilistic r where $r \sim [0,1)$.

Step 3: Get a candidate job CJ by the roulette-wheel selection scheme.

Step 3.1: If $r \sim [0, P(N_{\alpha,1} \mid N_{\alpha-1,\beta'})$, then set $CJ = 1$ and go to Step 4.

Step 3.2: If $r \sim [\sum_{\beta=1}^{w-1} P(N_{\alpha,\beta} \mid N_{\alpha-1,\beta'}), \sum_{\beta=1}^{w} P(N_{\alpha,\beta} \mid N_{\alpha-1,\beta'}))$ and $w \in \{2, \cdots, n\}$, then set $CJ = w$ and go to Step 4.

Step 4: Return CJ.

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

Step 1: Set $i = 1$.
Step 2: Generate a new individual $\boldsymbol{\pi}_i(\mathbf{gen+1})$.
  Step 2.1: Set $\pi_{ij}(gen+1) = 0$ for $j = 1, \cdots, n \times m \times L$.
  Step 2.2: Set $j = 1$.
  Step 2.3: $CJ = Job(\boldsymbol{\pi}_i(\mathbf{gen+1}), j)$.
  Step 2.4: If $l(CJ, \boldsymbol{\pi}_i(\mathbf{gen+1})) = m \times L$, 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 $j \le n \times m \times L$, then go to Step 2.3.
Step 3: Set $i = i + 1$.
Step 4: If $i \le PS$, then go to Step 2.

3.6 New Probabilistic Model Construction Strategy

Based on Schiavinotto and Stützle [15], the diameter of *Insert* is $n \times m \times L - 1$, which is one of the shortest diameters. That is, using *Insert* at most $n \times m \times L - 1$ times, one solution $\boldsymbol{\pi}$ can transit to any other solution $\boldsymbol{\pi}'$. This means *Insertion*-based neighborhood can perform a more efficient and thorough search than the other kinds of neighborhoods with the same running time. So, we utilize the *Insertion*-based neighborhood to generate $(n \times m \times L - 1)^2$ neighbors of $\boldsymbol{\pi}^{\mathbf{best}}(\mathbf{gen})$. Then, the best e% of the generated neighbors and $pop(gen)$ (i.e., the high quality individuals) are selected to construct the probabilistic model of BEDA.

3.7 Procedure of BEDA

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

Step 0: Denote $\boldsymbol{\pi}^{\mathbf{g\_best}}$ 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 $pop(1)$ by using the method in subsection 3.3.
Step 2: Set $gen = gen + 1$.
Step 3: Calculate the makespan of each individual in $pop(gen)$.
Step 4: Construct new probabilistic model by using the strategy in subsection 3.6, and update $\boldsymbol{\pi}^{\mathbf{g\_best}}$.

Step 5: Generate $pop(gen+1)$ by using the new population generation method in subsection 3.5.

Step 6: If $gen < genMax$, then go to Step2.

Step 7: Output $\boldsymbol{\pi}^{g\_best}$.

It can be seen from Steps 4 and 5 that the new generated individuals can aptly absorb the sub-sequences information of the high quality individuals during the evolution process and then guide the search to more promising regions. Thus, the algorithm is hopeful to obtain good results.

4 Simulation Result and Comparisons

4.1 Experimental Design

In order to test the performance of the proposed BEDA, a set of instances under different scales is randomly generated. The $n \times m \times L$ combinations include $10 \times 10 \times 2$, $10 \times 10 \times 3$, $20 \times 10 \times 3$, and $30 \times 10 \times 3$. For each scale, we generate 5 groups of data. The processing time t_j is generated from a uniform distribution [1,100] and the set-up time S_{ij} is generated from [1,20]. All algorithms are coded in Delphi 2010 and are executed on Mobile Intel Core 4 Duo 2.2 GHz processor with 8GB memory.

For each instance, each algorithm is run 20 times independently. Based on our previous experiments, the parameters of BEDA are set as follows: the population size $PS = 100$, and the proportion of the high quality individuals e%=20%. The parameters of GA are set the same as those in [2].

4.2 Comparisons of GA, BEDA_V1 and BEDA

For the purpose of showing the effectiveness of BEDA, we compare BEDA with GA and BEDA_V1 (i.e., a variant of BEDA). The difference between BEDA and BEDA_V1 lies in the probabilistic model and updating mechanism. The BEDA_V1 uses the probabilistic model and updating mechanism proposed by Baluja [5]. The learning rate α of BEDA_V1 is set to 0.1. The maximum generations of algorithms are set to 500. The running time of the algorithms are decided only by the scale of problems. The simulation results are listed in Table 1, where *BEST* denotes the best makespan, *AVG* denotes the average makespan and *WORST* denotes the worst makespan.

From Table 1, it can be seen that BEDA performs much better than GA and BEDA_V1 with respecting to solution quality. The values of *BEST*, *AVG* and *WORST* obtained by BEDA are much better than those obtained by GA and BEDA_V1. Specifically speaking, for instances 1 to 20, BEDA is better than GA and BEDA_V1 on all statistical indicators (i.e., *BEST*, *AVG* and *WORST*) except the *WORST* indicator of instances 09 and 12, and the *BEST* indicator of instance 18. So, it can be concluded that BEDA is an effective algorithm for the RJSSPST. Moreover, the test results also manifest a bayesian statistical inference-based probabilistic model is more suitable for guiding the search to the promising regions in the solution space of RJSSPST.

Table 1. Comparisons of *BEST*, *WORS* and *AVG* of GA, BEDA_V1 and BEDA

instances	*n*	*m*	*L*	GA[2]			BEDA_V1			BEDA		
				BEST	*WORST*	*AVG*	*BEST*	*WORST*	*AVG*	*BEST*	*WORST*	*AVG*
01	10	10	2	2771	3543	2971.95	2328	3307	2580.95	**1886**	**2493**	**2131.95**
02	10	10	2	2625	3864	3450.05	2135	2929	2500.35	**1612**	**2499**	**2129.70**
03	10	10	2	2815	3258	2949.40	1956	2852	2332.90	**1605**	**2089**	**1896.15**
04	10	10	2	2583	3387	2871.75	2176	2955	2477.70	**1867**	**2398**	**2119.05**
05	10	10	2	2911	4048	3771.25	2469	3376	2916.65	**1827**	**2905**	**2420.50**
06	10	10	3	5137	8789	6481.75	4720	9926	7473.25	**4613**	**8763**	**5986.85**
07	10	10	3	5098	11086	8974.95	4827	8662	6547.00	**3727**	**7088**	**5079.25**
08	10	10	3	5524	9112	6812.40	4644	7739	6045.90	**3424**	**6209**	**4810.05**
09	10	10	3	4992	**6713**	5658.35	4837	9019	6866.45	**4080**	7473	**5537.65**
10	10	10	3	5713	8771	7423.65	5179	12207	7592.95	**4012**	**8438**	**5860.35**
11	20	10	3	6757	7606	7155.10	6280	6592	6430.45	**5962**	**6541**	**6412.60**
12	20	10	3	6943	7816	7336.45	6442	**6685**	6663.40	**6256**	6691	**6490.60**
13	20	10	3	6901	7396	7208.65	6175	6724	6430.45	**6076**	**6667**	**6424.00**
14	20	10	3	6523	7342	7056.55	6130	6589	6335.50	**6022**	**6562**	**6323.80**
15	20	10	3	7030	7696	7357.90	6493	6832	6697.30	**6442**	**6685**	**6663.40**
16	30	10	3	8767	10348	9748.75	7993	8665	8489.05	**7882**	**8566**	**8388.90**
17	30	10	3	9550	10747	10294.00	8668	9211	8924.30	**8473**	**9208**	**8914.80**
18	30	10	3	8860	10015	9524.95	**7885**	8572	8263.00	7966	**8320**	**8224.00**
19	30	10	3	9193	10270	9812.05	8095	8839	8486.35	**7981**	**8614**	**7485.00**
20	30	10	3	9325	10561	10162.75	8182	9193	8850.25	**8089**	**9103**	**8827.30**

5 Conclusion and Future Work

This paper proposed a bayesian statistical inference-based estimation of distribution algorithm (BEDA) to solve the re-entrant job-shop scheduling problem with sequence-dependent setup times (RJSSPST). In BEDA, the initial population was generated by using stochastic method, and the search in the solution space was executed through sampling and updating the probabilistic model of BEDA. Simulation results and comparisons based on a set of instances showed the effectiveness of the proposed BEDA. To the best of our knowledge, this is the first paper on the application of estimation of distribution algorithm (EDA) for the RJSSPST. Our future work is to develop some BEDA-based algorithms to deal with re-entrant no-wait job-shop scheduling problem.

Acknowledgments. This research was partially supported by National Science Foundation of China (No. 60904081, 71101139), 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. Garey, M.R., Johnson, D.S., Sethi, R.: The Complexity of The Flowshop And Jobshop Scheduling. Computer Science Department 1(2), 117–129 (1976)
2. Sun, J.U.: A Genetic Algorithm for A Reentrant Job-shop Scheduling Problem with Sequence-dependent Setup Times. Engineering Optimization 41(6), 505–520 (2009)
3. Karg, L.L., Thompson, G.L.: A Heuristic Approach to The Traveling Salesman. Management Science 10(2), 225–248 (1964)
4. Adams, J., Balas, E., Zawack, D.: The Shifting Bottleneck Procedure for Job Shop Scheduling. Management Science 34(3), 391–401 (1988)
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.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)
8. Liu, H., Gao, L., Pan, Q.K.: A Hybrid Particle Swarm Optimization with Estimation of Distribution Algorithm for Solving Permutation Flowshop Scheduling Problem. Expert Systems with Applications 38(4), 4348–4360 (2011)
9. Wang, L., Wang, S.Y., Xu, Y., Zhou, G., Liu, M.: A Bi-population Based Estimation of Distribution Algorithm for The Flexible Job-shop Scheduling Problem. Computers & Industrial Engineering 62(4), 917–926 (2012)
10. Zhou, A., Gao, F., Zhang, G.: A Decomposition Based Estimation of Distribution Algorithm for Multiobjective Traveling Salesman Problems. Computers and Mathematics with Applications 66(10), 1857–1868 (2013)
11. Ruiz, R., Maroto, C., Alcaraz, J.: Two New Robust Genetic Algorithms for The Flowshop Scheduling Problem. Omega 34(5), 461–476 (2006)
12. Huelsenbeck, J.P., Ronquist, F.: MrBayes: Bayesian Inference of Phylogenetic Trees. Bioinformatics 17(8), 754–755 (2001)
13. Qian, B., Li, Z.H., Hu, R., Zhang, C.S.: A Hybrid Differential Evolution Algorithm for The Multi-objective Reentrant Job-shop Scheduling Problem. In: The 10th IEEE International Conference on Control & Automation, pp. 485–489 (2013)
14. Zhang, R., Wu, C.: A Hybrid Local Search Algorithm for Scheduling Real-world Job Shops with Batch-wise Pending Due Dates. Engineering Applications of Artificial Intelligence 25(2), 209–221 (2012)
15. Schiavinotto, T., Stützle, T.: A Review of Metrics on Permutations for Search Landscape Analysis. Computers & Operations Research 34(10), 3143–3153 (2007)