An Efficient Estimation of Distribution Algorithm for Job Shop Scheduling Problem

Xiao-juan He[1,2], Jian-chao Zeng[2], Song-dong Xue[2], and Li-fang Wang[2]

[1] College of Electrical and Information Engineering, Lanzhou University of Technology, Lanzhou 730050, China
[2] Complex System and Computational Intelligence Laboratory, Taiyuan University of Science and Technology, Taiyuan 030024, China
hexj1993@tom.com

Abstract. An estimation of distribution algorithm with probability model based on permutation information of neighboring operations for job shop scheduling problem was proposed. The probability model was given using frequency information of pair-wise operations neighboring. Then the structure of optimal individual was marked and the operations of optimal individual were partitioned to some independent sub-blocks. To avoid repeating search in same area and improve search speed, each sub-block was taken as a whole to be adjusted. Also, stochastic adjustment to the operations within each sub-block was introduced to enhance the local search ability. The experimental results show that the proposed algorithm is more robust and efficient.

Keywords: Job Shop scheduling problem, estimation of distribution algorithm, neighboring operations, probability model.

1 Introduction

The Job Shop scheduling problem (JSP) is one of the well-known NP-hard combinatorial optimization problems. Many methods for JSP were proposed[1]. The traditional methods such as branch-and-bound, mixed integer programming, and linear programming were not practical because they require a large amount of computation time especially for large scale problem solving[2]. Instead, a lot of researchers put their effort into meta-heuristic and local search optimization strategies for solving JSP. Among those are priority dispatching rules, shifting bottleneck procedure, tabu search algorithm, simulated annealing method, etc[3]. Evolutionary algorithms have also been applied to JSP solving and especially genetic algorithm displays obvious superiority[4,5,6]. Recently, an evolutionary algorithm based on probability analysis, i.e., estimation of distribution algorithms (EDAs)[7,8] has became very popular. The algorithm relies on the construction and maintenance of a probability model that characterizes satisfactory solutions for a problem. Some better solutions are selected from population and are used to update the probability model iteratively. This probabilistic model is used to guide further exploration of search

B.K. Panigrahi et al. (Eds.): SEMCCO 2010, LNCS 6466, pp. 656–663, 2010.

space and realize evolution process. EDAs effectively overcomes the problem of blocks being disrupted, has shown to perform very well on a wide variety of problems. Clearly, the key of EDAs for solving JSP is a probability model constructing[9]. Because of the complexity of job shop scheduling problem, it is very difficult to build a probability model being characteristics of JSP problem. Therefore the estimation of distribution algorithms is seldom applied in this field. In this paper, a new EDAs using frequency information of multi-operations neighboring was proposed for avoiding need of complex probability model in JSP solving.

This paper is organized as follows: In Section 2, we shortly describe the basic framework of EDAs and JSP. Section 3 gives a detail description of new proposed algorithm for JSP. Section 4 discusses the experimental results and shows the efficiency of proposed algorithm. Finally, Section 5 summarizes the contribution of this paper.

2 Problems Description

2.1 Job Shop Scheduling Problem

The job shop scheduling problem can be briefly described as follows[2]. There are n jobs to be processed through m machines. Each job consists of a sequence of operations. The operations of each job must be processed in the given sequence. The processing time is fixed and known. There are several constraints on jobs and machines as follows: 1) There are precedence constraints among operations of one job and there are no precedence constraints among operations of different job; 2) Each machine can process at most one operation at a time interval, and it cannot be interrupted; and 3) At most one operation of each job can be processed at a time interval. The objective of JSP is determining the appropriate operation sequences for all jobs that minimizes makespan($C_{\max}$).

2.2 Basic Framework of EDAs

EDAs has introduced a new paradigm of evolutionary computation, representing relations between the variables involved in the problem domain by building probability model. According to the complexity of probability models and different sampling methods, EDAs has a lot of different concrete realization methods. Main steps are summarized as follows:

Step 1. Generate initial population.
Step 2. Evaluate population. Constructed probability model according to the information of some better individuals selected from population.
Step 3. Generate new population sampling from the constructed model, evaluate new population.
Step 4. Update the probability model.

Repeat do the step 3and step 4 until the termination criteria is met.

3 EDAs for JSP

3.1 Encoding and Decoding Methods

The encoding method most used to solve the scheduling problem is operation-based encoding. Thus, we also apply this method in this paper. The same number denotes different operations of the same job, meaning their order of appearance in chromosome representation. The times which each job will appear in chromosome equal to the numbers of all operations of each job. For example, in a 3*3 job shop problem, a scheduling is denoted as 9 numbers, each job will appear 3 times exactly. An individual (1 3 2 2 1 3 3 2 1) is generated randomly, which denotes in turn process operations. '1'shows job1, '2'shows job2, '3'shows job3, and the 1th 1 shows the 1 operation of job 1, and the 2th 1 shows the 2 operation of job 1, and the 3th 1 shows the 3 operation of job 1, in the same way, etc. The important feature of the encoding method is that all offspring generation are feasible scheduling.

The decoding is done by priority-based way. That is, according to the order given operations in schedule and the current machine's idling state as well as the current operation's processing state, the operation that has the earliest starting time can be selected to be processed under the conditions both of the order remaining and of the other operations not being delayed.

3.2 Population Initialization and Fitness Evaluate

Let the set of operations $(o_{11}...o_{1m}, o_{21}...o_{2m}, ..., o_{n1}...o_{nm}) \to (o_1, o_2, ..., o_{n\times m})$, $\pi = (\pi(1), \pi(2), ..., \pi(n\times m))$ is a integer permutation form 1 to $n\times m$. Let the population be N. N integer sequences are randomly generated and consist of the initial population. One sequence of operations denotes one individual. Evaluate the fitness (makespan) of individuals. The less makespan be, the larger the corresponding value of fitness. Some better individuals are selected as superior population according to some proportional. Let the superior population size be D.

3.3 Construct and Update Probability Model

In this paper, we have constructed probability model according to the order of operations of individuals in superior population based on edge histogram[10]. Let the initial population be represented as $P(0)$, and the initial superior population be represented as D^0 . The kth individual is as $x_k^0 = (\pi_k^0(1), \pi_k^0(2), ..., \pi_k^0(n\times m))$, set $k \in D$. The initial population is generated by uniform distribution. Through calculating the frequencies of pair-wise neighboring operations appearing in superior population D^0 , the probability matrix is constructed as follows:

$$P^t = \begin{bmatrix} p_{11}^t & p_{12}^t & \cdots & p_{1,n\times m}^t \\ p_{21}^t & p_{22}^t & \cdots & p_{2,n\times m}^t \\ \cdots & \cdots & \cdots & \cdots \\ p_{n\times m,1}^t & p_{n\times m,2}^t & \cdots & p_{n\times m,n\times m}^t \end{bmatrix}$$

Where t represents the iterative number, and t=1.

$$p_{ij}^{\ t} = (1-\alpha)\cdot 1/n\times m + \frac{1}{D}\sum_{k=1}^{D}\delta_{ij}(x_k^0)\ , \quad i,j=1,2,...n\times m \tag{1}$$

$$\delta_{ij}(x_k^0) = \begin{cases} 1 & \pi_k^0(h)\to o_i \wedge \pi_k^0(h+1)\to o_j \\ 0 & otherelse \end{cases} \tag{2}$$

where $h\in\{1,2,...,(n\times m-1)\}$, $\delta_{ij}(x_k^0)$ equals to 1 denotes that the neighboring operation $o_i o_j$ appear in the structure of the kth individual. Otherwise $\delta_{ij}(x_k^0)$ equals to 0. The parameter α is a learning rate, and $\alpha\in(0,1)$. The higher the Parameter α is, the greater the contribution of the previous generation population to the next generation population is, otherwise the weaker the contribution. At the same time, This statistics way can increase the probability that the pair-wise neighboring operations appearing by high frequency evolve to the next generation population.

At iteration t(t>1), The new probability matrix is built through computing the frequencies of the pair-wise neighboring operations appearing in superior population D^t . Updated the probability is as follows[8]:

$$p_{ij}^{\ t+1} = (1-\alpha).p_{ij}^{\ t} + \alpha.\frac{1}{D}\sum_{k=1}^{D}\delta_{ij}(x_k^t)\ , \quad i,j=1,2,...n\times m \tag{3}$$

3.4 The Process of Block Optimization

According to the value of probability matrix, the structure of optimal individual is divided into some independent sub-blocks.

3.4.1 Construct the Model of Block Structure

If there are multi-neighboring operations, their frequency appearing in superior population is higher. Then in order to avoid the multi-neighboring operations being destroyed and repeating search in same space in iterative process, these operations are connected into a whole block and take part in iterative process.

At iteration l , let the corresponding probability matrix be P^l . According to P^l , the model of block structure is built by dividing to the structure of optimal individual.

Here, a parameter ∂ is set advanced. ∂ means a connection condition, set $\partial\in(0,1)$. If the frequency of multi-neighboring operations appearing in superior population is bigger than the parameter ∂ , then these neighboring operations can be connected into a block. The ∂ is bigger, the frequency is higher. Conversely it is lower. The main process is as follows: denotes the optimal individual as $x^* = (\pi_*^{\ l}(1), \pi_*^{\ l}(2), ..., \pi_*^{\ l}(n\times m))$, and set $\pi_*^{\ l}(h)\to o_i \ \wedge\ \pi_*^{\ l}(h+1)\to o_j$. Corresponding to the matrix P^l , we consider that the operation o_i and o_j have

relativity and belong to a same sub-block if p_{ij} is higher than ∂. Otherwise they belong to different sub-block. Then, they separately being as a whole take part in next iterative process.

If value of ∂ is too small, the connection condition is too easy to be met, then the multi-neighboring operations can rapidly be connected into a sub block, which will lead to the local optimum of the algorithm. Conversely, if ∂ is too bigger, the connection condition is difficult to be met, and the algorithm is always searching in whole space, and the search efficiency will be lowered, which can not easy to search the optimal solution. Generally empirical set $\partial \in (0.7, 0.9)$.

3.4.2 Generate Initial Sub-block Population

Let the number of sub-blocks be integer R. M permutations are randomly generated form 1 to R as sub-block initial population. Let initial sub-block superior population be represented by G^0.

3.4.3 Built and Update Block Probability Model

The fitness of individuals is calculated according to permutation of sub-blocks. Select the optimal individual and the sub-block superior population. Let the size of the sub-block superior population be m. At iteration t, let the sub-block superior population be G^t. The kth individual is denoted as $y_k^t = (\tau_k^t(1), \tau_k^t(2), ..., \tau_k^t(R))$, $k \in m$. Because the sub-blocks are independent each other, and the order among all sub-blocks can be arbitrarily exchanged, so in the G^t, the frequencies by which each sub-block is arranged on all positions are computed, the block model of probability is built as follows:

$$Q^t = \begin{bmatrix} q_{11}^t & q_{12}^t & \cdots & q_{1R}^t \\ q_{21}^t & q_{22}^t & \cdots & q_{2R}^t \\ \cdots & \cdots & \cdots & \cdots \\ q_{R1}^t & q_{R2}^t & \cdots & q_{RR}^t \end{bmatrix}$$

Where q_{ij}^t is the probability in which the ith sub-block is arranged on the jth position. Set i, j=1,2,…R. In the iterative process, The probability is updated as follows[3]:

$$q_{ij}^{t+1} = (1-\beta).q_{ij}^t + \beta.\frac{1}{m}\sum_{k=1}^{m} \lambda_{kij}^t(y_k^t) \tag{4}$$

$$\lambda_{kij}^t = \begin{cases} 1 & \tau_k^t(i) \text{ is arranged on the jth position} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

Set $\beta \in (0,1)$, α and β have the same meaning. The values of α and β may be equal or not. Let α equal to β in this paper.

3.4.4 Generate New Sub-block Population

The new sub-block population are generated by roulette wheel sampling from each column of the probability model Q^t. The new sub-block superior population are chosen from new sub-block population, then return to step 3.4.3, the probability model Q^t will be updated.

The above steps from 3.4.3 to 3.4.4 are done repeatedly. When the fitness value of optimal individual remains unchanged in continual several iterative processes, two positions are randomly generated and their operations are exchanged within each sub-block, thus some new individuals will be generated. Let these new individuals and the individuals of sub block superior population be combined and then be reorder according to their fitness value, and then the new sub-block superior population is selected from them again.

3.5 Generate New Population

In this paper, the new population contains two portions. First portion contains N individuals which are generated by roulette wheel sampling from probability matrix P^l. The sampling method is as follows: Randomly select a initial point(operation), o_i for example, the next operation is selected from the ith row of probability matrix P^l by roulette wheel. If the selected next operation is o_j, then the next operation is selected again from the jth row of probability matrix P^l using roulette wheel, etc. Until all operations are selected. The other portion is the new sub-block superior population which is obtained through the process of block optimization. Let these two portion populations compose of the new population and go to 3.3.

Do repeatedly the above step from 3.3 to 3.5 until the termination criteria are met. The elitist strategy is applied to the whole iterative process for ensuring the convergence of algorithm.

In order to illustrate the performances of the proposed algorithm, no other heuristic algorithm is used in optimal process. The proposed algorithm is a blind search algorithm.

4 Computational Results

In this paper, we use some instances that are taken from the OR_Library as test benchmarks to test our new proposed algorithm. In these instances, FT06, FT10 and FT20 are designed by Fisher and Thompso and the others are the first instance of other type instance set. Because the proposed algorithm in this paper is a pure EDAs that doesn't hybrid other heuristic algorithms in optimal process, so the computation results are compared with pure PSO[6]. In pure PSO, the population size is 30 and the most iteration number is 10^5. In this paper, the parameters are defined in the following: the population size is 30 and the most iteration number is 5000. Let

the selected proportion of the superior population be 0.3, and set α =0.15and ∂ =0.9. When the fitness value of the best solution is not improved after the continual 5 times in iteration process, the process of block optimal is executed. After the process of block optimal iterates 5 times, return the next iterative process. Each instance is executed for 10 runs. In Table 1, Instance means the problem name. Size means the problem size. Optimum means the best known solution for the test instance. Let Min/Max/Average separately be the minimum value, maximal value and average value which are obtained in 10 iteration process of each instance.

Table 1. The experimental results comparison of EDAs and the PSO for JSP

Instance	Size	Optimum	algorithm					
			PSO			EDAs		
			Min	Max	average	Min	Max	average
FT06	6×6	55	55	59	56.1	55	55	55.0
FT10	10×10	930	985	1084	1035.6	937	937	937.0
FT20	20×5	1165	1208	1352	1266.9	1184	1184	1184.0
LA01	10×5	666	666	688	668.6	666	666	666.0
LA06	15×5	926	926	926	926.0	926	926	926.0
LA11	20×5	1222	1222	1222	1222	1222	1222	1222.0
LA16	10×10	945	945	956	1035	945	946	945.5
LA21	15×10	1046	1102	1147	1128.4	1071	1073	1071.6
LA26	20×10	1218	1263	1351	1312.6	1257	1261	1257.8
LA31	30×10	1784	1789	1897	1830.4	1789	1789	1789.0
LA36	15×15	1268	1373	1436	1409.2	1292	1315	1312.7

From the result of Table 1, we can see that the values of Min, Max and Average in EDAs are all less than those in the pure PSO. This shows that EDAs has better performance than pure PSO. At the same time, the difference among Min, Max and Average in EDAs is very small. It shows that EDAs has well stability.

5 Conclusions

In this paper, we adopt EDAs to solve the job shop scheduling problem. The probability model is built through considering the information of neighboring operations appearing in superior population. This makes the probability model well reflect the character of job shop scheduling problem, and improves the ability of EDAs to solve job shop scheduling problem. The simulation results show that the proposed algorithm is more robust and efficient. Because the algorithm is a blind search without hybrid other heuristics search algorithm, then we test the performance of hybrid algorithm remain for future.

Acknowledgments. This paper is supported by The Shanxi Science Foundation for Young Scientists under Grant 2010021017-2.

References

1. Garey, M.R., Johnson, D.S., Sethi, R.: The complexity of flowshop and jobshop scheduling. Mathematics of Operations Research 1, 117–129 (1976)
2. Blazewicz, J., Domschke, W., Pesch, E.: The job shop scheduling problem: Conventional and new solution techniques. European Journal of Operational Research 93, 1–33 (1996)
3. Binato, S., Hery, W.J., Loewenstern, D.M.: Resende M.G.C. 2002, A GRASP for job shop scheduling. In: Ribeiro, C.C., Hansen, P. (eds.) Essays and Surveys in Metaheuristics. Kluwer Academic Publishers, Dordrecht (2002)
4. Watanabe, M., Ida, K., Gen, M.: A genetic algorithm with modified crossover operator and search area adaptation for the job-shop scheduling problem. Computers &Industrial Engineering 48, 743–752 (2005)
5. Lian, Z., Jiao, B., Gu, X.: A similar particle swarm optimization algorithm for job-shop scheduling to minimize makespan. Applied Mathematics and Computation 183, 1008–1017 (2006)
6. Lin, T.-J., Horng, S.-J., Kao, T.-W., Chen, Y.-H., Run, R.-S.: An efficient job-shop scheduling algorithm based on particle swarm optimization. Expert Systems with Applications 37, 2629–2636 (2010)
7. Baluja, S.: Population-Based Incremental Learning: A Method for Integrating Genetic Search Based Function Optimization and Competitive Learning. Technical Report CMU-CS-95-163. Carnegie Mellon University, Pittsburgh (1994)
8. Pelikan, M., Goldberg, D.E., Lobo, F.: A Survey of Optimization by Building and Using Probabilistic Models. ILLiGAL Report No.99018, University of I11inois at Urbana Champaign, I11inois Genetic Algorithms Laboratory, Urbana, I11inois (1999)
9. Pelikan, M., Goldberg, D.E., Cantu-Paz, E.: BOA: the Bayesian Optimization Algorithm. In: Proceedings of the Genetic and Evolutionary Computation Conference GECCO 1999, vol. I, pp. 525–532 (1999)
10. Tsutsui, S.: Probabilistic Model-Building Genetic Algorithms in Permutation Representation Domain Using Edge Histogram. In: Guervós, J.J.M., Adamidis, P.A., Beyer, H.-G., Fernández-Villacañas, J.-L., Schwefel, H.-P. (eds.) PPSN 2002. LNCS, vol. 2439, Springer, Heidelberg (2002)