International Journal of Industrial Engineering, 18(4), 162-168, 2011.

IMPROVED CULTURAL ALGORITHMS FOR JOB SHOP SCHEDULING PROBLEM

Weiling Wang, Tieke Li

School of Economics and Management
University of Science and Technology,
Beijing 100083, China
Corresponding author: Weiling Wang, wangweiling@126.com

This paper presents a new cultural algorithm for job shop scheduling problem. Unlike the canonical genetic algorithm, in which random elitist selection and mutational genetics is assumed. The proposed cultural algorithm extract the useful knowledge from the population space of genetic algorithm to form belief space, and utilize it to guide the genetic operator of selection and mutation. The different sizes of the benchmark data taken from literature are used to analyze the efficacy of this algorithm. Experimental results indicate that it outperforms current approaches using canonical genetic algorithms in computational time and quality of the solutions.

Keywords: Job shop scheduling problem, Cultural algorithm, Genetic algorithm, *K*-nearest neighbor method, Neighbor search mutation

(Received 2 July 2009; Accepted in revised form 10 September 2010)

1. INTRODUCTION

The importance of scheduling has increased in recent years due to the growing consumer demand for variety, reduced product life cycles, changing markets with global competition and rapid development of new processes and technologies. These economic and commercial market pressures emphasize the need for a strategy to minimize inventory while maintaining customer satisfaction levels of production and delivery specification. Often, this requires an efficient, effective and accurate scheduling plan.

The classical Job-shop Scheduling Problem (JSP), a common model of scheduling in practice[1] has been proved to be NP-hard[2].Scholars have proposed many meta-heuristics methods to solve the scheduling problems in the recent decades, including tabu search, evolutionary programming, genetic algorithm and artificial neural networks, but these algorithms have one or more drawbacks, such as premature convergence and being trapped into local optimum or taking too much computation time. In recent years, the Cultural Algorithm (CA) proposed by R.G. Reynolds in 1994[3] has become a candidate for many optimization applications, owing to its flexibility and efficiency. It has been successfully applied to solve scheduling optimization problems and promises to overcome some shortcomings of the above optimization methods[4][5][6]. On the base of them, this paper represents a new cultural algorithm to solve the classical job shop scheduling problem, which uses the knowledge of the excellent chromosome of the schema to guide selection and mutation operator. We introduce the K-nearest neighbor method[7] for the dynamic learning in the process of selection operator, and introduce neighbor search mutation in the process of mutation operator. Experimental results prove that the proposed algorithm is effective.

2. PROBLEM FORMULATIONS

According to the $(\alpha|\beta|\gamma)$ notation[8], the classical job shop scheduling problem with objective of minimizing makespan of the schedule can be denoted as $J_m|C_{\max}$. Let the variable y_{ij} denotes the start time of job $j = 1,2,...,n$ being processed on machine $i = 1,2,...,m$. In addition, the processing time of job j on machine i , p_{ij} is known for each operation in a job's process routing. The mathematical model of the classical job shop scheduling problem can be described as follows[9]:

Min $C_{\max}$

Subject to $y_{kj} - y_{ij} \geq p_{ij}$ ... (1)

$$y_{ij} - y_{il} \geq p_{ij} \cup y_{il} - y_{ij} \geq p_{ij} \quad \ldots \quad (2)$$

$$y_{ij} \geq 0 \quad \ldots \quad (3)$$

In this formulation, the first set of constraints ensure that operation (k, j) can not start before operation (i, j) is completed (precedence constraints). The second sets of constraints ensure that some ordering exists among operations of different jobs that have to be processed on the same machine (exclusive constraints). The third set of constraints ensures that the start time of every operation of every job is feasible. What's more, we shall assume that:

All machines are available at time 0.

All jobs are released at time o.

3. THE PROPOSED CULTURAL ALGORITHM

3.1 The architecture of the algorithm

Cultural algorithm can model cultural evolution [3], which is one of knowledge-based evolution strategies. We use it preserve beliefs of effective schedules that contain good schemata from each generation. In a cultural-based GA (or CGA), domain knowledge is used to represent, store and transmit knowledge from one generation to the next. It helps to reduce the search space by pruning useless parts and promote desirable parts. CGA models such 'cultural influences' through two spaces: *population space* and *belief space*. The population space is implemented by a canonical GA, in that it uses belief-revised operators of selection, crossover and mutation, while the belief space models the *behavioral traits* of the current population. The application of the belief-space model is similar to the memories of the Tabu Search in which the good features of previous solutions are used to influence current solution [5].

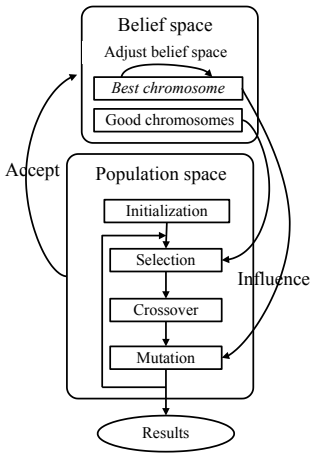

Figure 1. The overall control-flow of CGA

Figure 1 gives the overall control-flow of CGA. The belief space contains the possible schema that can be produced over generalizations on population space. In our algorithm, we divide it into two parts: *best chromosome belief space* which is the best individual and *good chromosome belief space* which consist of predefined number of good individuals. The Population space is implemented by a canonical GA. Good schema is selected after a predefined number of generations and is used to modify the belief space so as to improve the search. The current state of the belief space is then used to modify the performance of current individuals in the population by using belief to revise selection and mutation operators.

3.2 The details of the algorithm

The proposed algorithm in detail is depicted as follows.

Representation

We used the *job Permutation with repetition Representation*[10]. For the classical JSSP of size $n \times m$, a chromosome is composed of the genes of $n \times m$, and each gene denotes each job. For example, in case of the chromosome, [3, 2, 2, 1, 1, 2, 3, 1, 3] of Figure 2, first gene "3" denotes operation 1 of job 3. Second gene "2" denotes operation 1 of job2. Third gene "2" denotes operation 2 of job 2. The advantage of this representation is that every individual produced by the genetic operators is feasible[11].

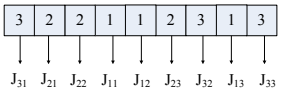

Figure 2. 3×3 JSSP

We use two ways to generate the representation as follows:

(1) Utilize the heuristic (SPT,MWKR,LWKR等)to generate the initial chromosome；

(2) Generate randomly。

Decoding

We use an active schedule-based decoding algorithm. A semi-active schedule is a feasible schedule where no operation can be started earlier without changing the order or violating the constraints on any one of the machines. An active schedule is a feasible schedule where no operation can be started earlier without delaying at least one other operation or violating the constraints on any one of the machines [9]. The semi-active schedule does not guarantee to obtain optimal results and its makespan is still larger or equal to the active one. The active schedule decoding algorithm has a complexity of $O(n^2)$ while the semi-active decoding algorithm has a complexity of $O(n)$, n being the total number of operations.

Selection operator

We employ tournament-based selection to generate a subpopulation. Then a predefined number of good individuals are selected from the subpopulation. The similarity of the individuals in *good chromosome belief space* with them is copulated respectively to find k similar chromosomes by *K*-Nearest Neighborhood Method [7]. Then these k chromosomes are inserted to the current subpopulation by replacing the k worse ones. The formulation of the similarity is depicted as follows:

$$S(chm1, chm2) = \sum_{i=1}^{l} chm1(i) \otimes chm2(i)$$

Where chm1 and chm2 denotes the two individuals respectively, i denote the position of the gene in the chromosomes; l denote the length of the chromosomes which is generally $n \times m$ in classical JSSP with size $m \times n$. $chm1(i) \otimes chm2(i)$ turns 1 if the genes are same in the responding position, and otherwise it turns 0. Therefore, if the value of S is large, two chromosomes are close. This measure takes $O(m \times n)$computational time.

Crossover operator

Here two-point crossover is applied [11]. To explain its operation, consider two parents: (2 1 2 1 1) and (1 1 1 2 2). A substring is selected randomly from parent 1: (2 1 2 1 1). Reading the string from left to right, we know that operations in the selected substring are the first operation of job J_1, the second operation of job J_2, and the second operation of job J_1. The corresponding positions of the characters in this string are then found and deleted in the second parent: (1 1 1 2 2). The substring is inserted to the second parent at the same position in the first parent to create a new child: (1 1 2 1 2).

Mutation operator

Mutation operator is inspired by the paper [4] [12]. The main idea of it is to make the individual better similar with the best individual by selecting the appropriate the mutated position of the chromosomes and doing *NSM* • (Neighbor

Search Mutation).At first we choose an individual from the subpopulation according to the probability of the mutation. Compare this individual with the best individual in *Operation belief space* to record the different position between them *diffpos* and the number of different position *i*. According to the number *i* it decide the mutation way. The detail is depicted as follows:

(1) When $i \le 1$, that is to say that the selected chromosome is the same with the best chromosome. Then select 3 genes randomly and do *NSM*;

(2) When $i = 2$, that is to say that there is only two position value differently with the best chromosome. Then swap them with each other.

(3) When $i \ge 3$, we generate a random permutation P from 1to *i*, and get 3 values in turn from left to right in P every times, which is used to decide the gene position to do *NSM* , then do mutation operator, until repeating $\frac{i}{3}$ times.

The notion of *NSM* • (Neighbor Search Mutation) shows in figure 4 as follows. For 4×4 JSSP，we assume get 3 genes, then calculate the different combinations of these 3 genes, and select the best combination to enter into subpopulation.

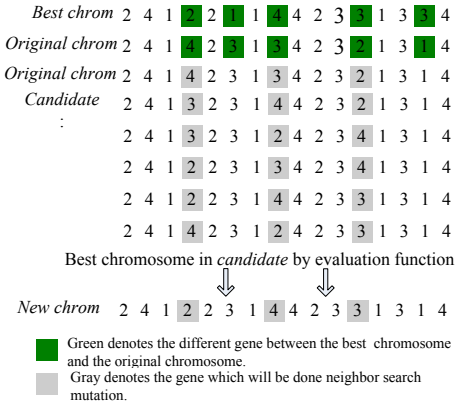

Figure 3. The mutation of the operation

The stop criterion of the iteration of CGA

If the number of the iteration arrive to the maximal iteration or no changes in the result were reported after a certain number of consecutive iterations.

4. EXPERIMENTAL RESULTS

Simulink tool: matlab7.0. Hardware environment: CPU processor 1.6GHz, 512M memory. The benchmark data is taken from S. Lawrence (1984), which is downloaded from the website of ORLibrary.

The basic parameters: $Popsize = 20$; $p_c = 0.7$; $p_m = 0.1$; the updating frequency of belief space =5. Table 2 shows the results of the different size instances after running 20 times respectively, in which GA denotes the canonical genetic algorithm，CGA denotes the proposed cultural genetic algorithm. Table 3 presents the data of CULT showed in the reference [4]. It is pointed out that the data attained by CULT is the result after running 200000 times, while $MaxIter^2$ denotes the number of the iteration when it gets the optimal value.

Table 2 the column of $MaxIter^1$ lists the iterations of every instance. When la01,la06 and la11 run 50 times in turn, they can get the best value, while la16 and la21 run 5000 and 10000 times respectively. Compare it with the column of $MaxIter^2$ in table 2, when CULT get the best value, the iterations of every instance is more than CGA. In this case, CGA attain the solution with all squares that the iteration of CULT is 200000 times.

From table1 we can see that CGA outperform GA apparently when their iterations is the same. Canonical genetic algorithm is hard to get the best solution, even if it gets, the probability of it is less than CGA. With the exception of la21, CGA all gets the best solution, in which la06 and la11 all get the best solution in 20 running. And all of the running time of CGA is less than it of GA. Therefore we can conclude the performance of CGA outperforms it of GA.

Table 1. Comparison of canonical genetic algorithm with cultural genetic algorithm

Instance	Size	BKS	GA				CGA				MaxIter1
			BV	AV	WV	AT(s)	BV	AV	WV	AT(s)	
LA01	10×5	666	667	673.4	705	3.6	666	666.9	675	2.3	50
LA06	15×5	926	926	936	956	9.82	926	926	926	4.61	50
LA11	20×5	1222	1222	1236	1245	14.5	1222	1222	1222	6.2	50
LA16	10×10	945	960	976.6	993	775.6	945	948.1	956	428.8	5000
LA21	15×10	1046	1147	1154	1161	1105	1051	1078	1103	722.3	10000

Notice：BKS denotes the best known solution. BV denotes the best value. AV denotes the average value. WV denotes the worst value. AT(s) denotes the average time of running every instance. MaxIter[1] denotes the maximal iterations of GA and CGA.

Table 2. The data of CULT in the reference [4]

Instance	Size	CULT			
		BV	AV	WV	MaxIter2
LA01	10×5	666	666.5	668	500
LA06	15×5	926	926	926	100
LA11	20×5	1222	1222.3	1224	100
LA16	10×10	945	962.8	990	10000
LA21	15×10	1059	1093	1114	200000

Figure 5 show the convergence curve of GA and CGA when solving the instance of la16,in which GA get the local optimal point(968)then can not get more better solution while CGA get the global optimal point in higher speed in the 762 iteration.

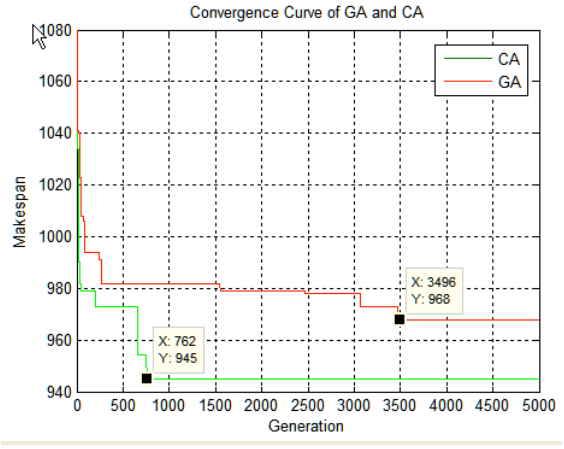

Figure 4. One running process of GA and CA

5. CONCLUSIONS

The paper proposes a new cultural algorithm for job shop scheduling problem on the base of canonical genetic algorithm.

(1) Unlike the genetic algorithm with random, this algorithm form the mutual inherence between belief space and population space;

(2) Introduce the heuristic in the initial population generator to make it have a better initial solution;

(3) Introduce k-nearest neighbor method in the process of selection operator to calculate the similarity to insure the partial better chromosome to enter into the subpopulation;

(4) Introduce neighbor search mutation in the process of mutation operator to reduce the number of iteration.

By means of the analysis as mentioned, we can conclude that cultural algorithm which is the strategy of evolution on the base of knowledge is effective to solve the job shop scheduling problem. From another aspect the validity of 'No Free Lunch theorem[13] is proved, which is that knowledge of the problem domain can help the search process to get better results.

In the future we will integrate culture algorithm and tabu search, or local search and so on to get better performance. What's more, we may apply this algorithm to the other scheduling problem, such as flow shop scheduling, the scheduling with parallel machine.

ACKNOWLEDGEMENT

This work is supported by National Natural Science Foundation of China (NSFC) Grant # 70771008 and #70371057.

6. REFERENCES

[1] A.S.Jain and S. Meeran. Deterministic job-shop scheduling: Past, present and future [J]. European Journal of Operation Research 113(2) (1998) 390-434.

[2] M.R. Garey, D.S. Johnson, R. Sethi. The complexity of flow shop and job-shop scheduling [J]. Mathematics of Operations Research 1(2) (1996) 117-129.

[3] Reynolds, R.G. An Introduction to Cultural Algorithms, Proc. 3rd Annual Conference on Evolutionary Programming, River Edge, NJ World Scientific (1994) 131-139.

[4] Daniel Cortés Rivera, Ricardo Landa Becerra, Carlos A. Coello Coello. Cultural algorithms, an alternative heuristic to solve the job shop scheduling problem [J]. Engineering Optimization, 2007, 39 (1): 69 – 85.

[5] N.B. Ho, J.C. Tay. GENACE: An efficient cultural algorithm for solving the flexible job-shop problem[C]. In: Proceedings of the Congress on Evolutionary Computation CEC2004, 2004:1759–1766.

[6] Nhu Binh Ho, Joc Cing Tay and Edmund M.-K. Lai. An effective architecture for learning and evolving flexible job-shop schedules [J]. European Journal of Operational Research, 2007, 179(2):316-333.

[7] T. Hastie, R. Tibshirani, J. Friedman, the Elements of Statistical Learning [M], Springer, New York, 2001 (Chapter 13).

[8] Hengyang Tang, Chuangli Zhao. The theory of sequencing [M].Beijing, science publishing Company.2000:7.

[9] M. Pinedo. Scheduling Theory, Algorithms, and Systems [M]. Prentice-Hall, Englewood Cliffs, NJ. 2002:159.

[10] Christian Bierwirth. A generalized permutation approach to job shop scheduling with genetic algorithms [J]. Springer Berlin / Heidelberg. 1995, 17: 87-92.

[11] R. Varela, C.R. Vela, J. Puente and A. Gómez. A knowledge-based evolutionary strategy for scheduling problems with bottlenecks [J]. European Journal of Operational Research. 2003,145 (1): 57–71.

[12] Y. Tsujimura, M. Gen and R. Cheng. Improved Genetic Algorithms for Solving Job-shop Scheduling Problem [J].Engineering Design and Automation, 1997, 3（2）:133-144.

[13] David H. Wolpert, William G. Macready. No free lunch theorems for optimization [J]. IEEE Transactions on Evolutionary Computation, 1997, 1(1):67-82.

BIOGRAPHICAL SKETCH

WANG Weiling, born in 1979, is currently a PhD in School of Economics and Management University of Science and Technology Beijing, China. Her research interests include production scheduling and controlling, metaheuristic algorithms design and analysis.

LI Tieke, born in 1958, is currently a professor in School of Economics and Management University of Science and Technology Beijing, China. His research interests include advanced manufacturing and management, production scheduling and controlling, metaheuristic algorithms and so on.