



Complex Adaptive Systems, Publication 2
Cihan H. Dagli, Editor in Chief
Conference Organized by Missouri University of Science and Technology
2012- Washington D.C.

Modified Genetic Algorithm for Flexible Job-Shop Scheduling Problems

Wannaporn Teekeng*, Arit Thammano

*Computational Intelligence Laboratory, Faculty of Information Technology,
King Mongkut's Institute of Technology Ladkrabang, Bangkok 10520 Thailand*

Abstract

This paper proposes a modified version of the genetic algorithm for flexible job-shop scheduling problems (FJSP). The genetic algorithm (GA), a class of stochastic search algorithms, is very effective at finding optimal solutions to a wide variety of problems. The proposed modified GA consists of 1) an effective selection method called “fuzzy roulette wheel selection,” 2) a new crossover operator that uses a hierarchical clustering concept to cluster the population in each generation, and 3) a new mutation operator that helps in maintaining population diversity and overcoming premature convergence. The objective of this research is to find a schedule that minimizes the makespan of the FJSP. The experimental results on 10 well-known benchmark instances show that the proposed algorithm is quite efficient in solving flexible job-shop scheduling problems.

Keywords: Flexible Job-Shop Scheduling Problems; Genetic Algorithm; Fuzzy Roulette Wheel Selection; Hierarchical Clustering

1. Introduction

The flexible job-shop scheduling problem (FJSP) is one of the hardest combinatorial optimization problems. It is an extension of the classical job-shop scheduling problem (JSSP). The difference between the two is that FJSP allows each operation to be processed on more than one machine. However, different machines require different processing time for each operation. Comparing to the JSSP, FJSP is more complex and more difficult to solve. Recently, a number of metaheuristic approaches, such as Genetic Algorithms (GA) [1], Tabu Search [2], Simulated Annealing (SA) [3], Particle Swarm Optimization (PSO) [4], and other approaches including Fuzzy Logic [5] have gained a lot of attention from researchers in the area.

GA has been applied to many combinatorial problems such as job scheduling and the travelling salesman problem. Some relevant works are: Pezzella et al. [6] proposed an algorithm that integrates different strategies for

* Corresponding author. Tel.: +66-2-723-4964; fax: +66-2-723-4910.
E-mail address: w_teekeng@hotmail.com.

generating the initial population, selecting the individuals for reproduction, and reproducing new individuals. Zhang et al. [7] developed new concepts for generating a high-quality initial population, called Global selection (GS) and Local selection (LS). Al-Hinai and ElMekkawy [8] proposed hybridized genetic algorithm architecture for the FJSP. In their algorithm, the efficiency of the GA is enhanced by integrating it with an initial population generation algorithm and a local search method. Ho et al. [9] proposed a new architecture based on the GA for learning and evolving FJSP called learnable genetic architecture (LEGA).

In this paper, we propose three new operations for the genetic algorithm. First, a new selection method, called fuzzy roulette wheel selection, is proposed and used in selecting chromosomes to form a mating pool. Second, a new crossover operator which employs the concept of a hierarchical clustering algorithm is used in crossover the mating pair. Third, a new mutation operator is applied to the offspring to maintain population diversity and overcome premature convergence.

This paper is organized as follows: Section 2 briefly describes the flexible job-shop scheduling problems. In Section 3, the proposed algorithm is presented. The experimental results are given in Section 4. Section 5 provides the conclusion to the paper.

2. Problem definition

The Flexible Job-Shop Scheduling Problem (FJSP) is an extension of the classical job shop scheduling problem. It allows an operation to be processed by any machine out of a set of available machines. The objective of FJSP is to schedule a set of R jobs $J = \{J_1, J_2, \dots, J_R\}$ on a set of S machines $M = \{M_1, M_2, \dots, M_S\}$ so that the makespan (C_{\max}) is minimized. Each job may have a different number of operations. Each operation $O_{i,j}$, the j^{th} operation of the i^{th} job, can be processed on any of the available machines. An example of FJSP is given in Table 1. Each row refers to an operation; each column refers to a machine and cells are processing times. As illustrated in Table 1, for example, the 3rd operation of the 2nd job is only allowed to be processed on M_1 and M_3 . The symbol “-” means that the machine cannot execute the corresponding operation.

Table 1. Example of the flexible job-shop scheduling problem.

Job	Operation	Machines		
		M_1	M_2	M_3
J_1	$O_{1,1}$	2	4	8
	$O_{1,2}$	5	4	7
	$O_{1,3}$	2	-	-
J_2	$O_{2,1}$	10	11	10
	$O_{2,2}$	6	5	-
	$O_{2,3}$	2	-	6

The following hypotheses for the FJSP, which are previously stated in [10], are made in this paper:

- All machines are available at time $t = 0$.
- All jobs are released at time $t = 0$.
- For each job, the sequence of operations is predetermined and cannot be modified.
- Each machine can only execute one operation at a time.
- Each operation $O_{i,j}$ must be processed without interruption on one of the S machines.
- The objective of the FJSP is to minimize the makespan.

3. The proposed algorithm

The proposed algorithm is developed according to the concept of the genetic algorithm. The detailed procedure of the proposed algorithm is as follows:

3.1. Generate the Initial Population

As shown in Fig 1, the structure of the chromosome used in this paper consists of two components. The first component of the chromosome represents the sequence of operations to be processed while the second component contains the list of machines used in executing the operations in the first component.

For example, consider the 2-job, 3-machine problem given in Table 1. An initial population of 4 chromosomes is randomly created as shown in Fig 2. Each chromosome contains 12 genes. The numbers 1 and 2 which appear in the first component of the chromosome stand for jobs J_1 and J_2 respectively. Since each job has three operations, the numbers 1 and 2 therefore occur three times in the first component of the chromosome. On the other hand, the numbers 1, 2, and 3 which appear in the second component of the chromosome stand for machines M_1 , M_2 , and M_3 respectively. According to the chromosome 4 in Fig 2, the second component is [3 2 3 2 2 1]. The first gene in the second component 3 means that the 1st operation of the 2nd job is to be processed by the machine M_3 .

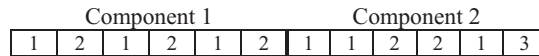


Fig. 1. Schematic representation of the chromosome structure

Chromosome 1	1	2	2	2	1	1	2	2	2	1	1	1
Chromosome 2	1	2	1	1	2	2	1	1	1	1	1	3
Chromosome 3	2	1	1	2	2	1	2	1	2	2	2	1
Chromosome 4	2	2	2	1	1	1	3	2	3	2	2	1

Fig. 2. Initial chromosomes

3.2. Fitness Evaluation

In this step, a fitness function is applied to evaluate the fitness of all chromosomes in the population. In line with the main objective of the FJSP, the fitness function used in this study is the reciprocal of the makespan of the chromosome.

3.3. Selection

In the genetic algorithm, parent chromosomes are selected with a probability related to their fitness. Highly fit chromosomes have a higher probability of being selected for mating than less fit chromosomes. In order to improve the performance of the genetic algorithm, the new selection method, called the fuzzy roulette wheel selection, is proposed in this paper. The concept of the fuzzy roulette wheel selection is that two chromosomes which are similar in content are placed next to each other with some degree of overlap in the roulette wheel [11]. The similarity between two chromosomes is measured based on the number of genes on one chromosome whose position and content are identical to those of the other chromosome. To be more specific, the similarity value of a pair of chromosomes is the summation of 1) the number of the same $O_{i,j}$ positions that both chromosomes have and 2) the number of the operations $O_{i,j}$ on both chromosomes that have chosen the same machine. For example, in Table 2, the population consists of 4 chromosomes; each chromosome represents a different possible solution. The following paragraphs show how to create the fuzzy roulette wheel selection.

Table 2. Some solutions of the example in Table 1

Chromosome	Solutions
1	O _{1,1} , O _{2,1} , O _{2,2} , O _{2,3} , O _{1,2} , O _{1,3} M ₂ , M ₂ , M ₂ , M ₁ , M ₁ , M ₁
2	O _{1,1} , O _{2,1} , O _{1,2} , O _{1,3} , O _{2,2} , O _{2,3} M ₁ , M ₁ , M ₁ , M ₁ , M ₁ , M ₃
3	O _{2,1} , O _{1,1} , O _{1,2} , O _{2,2} , O _{2,3} , O _{1,3} M ₂ , M ₁ , M ₂ , M ₂ , M ₂ , M ₁
4	O _{2,1} , O _{2,2} , O _{2,3} , O _{1,1} , O _{1,2} , O _{1,3} M ₃ , M ₂ , M ₃ , M ₂ , M ₂ , M ₁

In this proposed selection method, the first chromosome is always assigned to the first slot of the roulette wheel. The slot size is proportional to its fitness value. Then the similarity between the first chromosome and the rest of the chromosomes (chromosomes 2, 3, and 4) are calculated. A pair of chromosomes with the highest similarity score is chosen; in this case, the fourth chromosome is selected with a similarity score of 5 as shown in Fig 3. As a result, the fourth chromosome occupies the second slot. The above steps are repeated until all chromosomes are allocated slots in roulette wheel. Next, the overlapping area between the two adjacent chromosomes is determined. The size of the overlapping area is positively correlated with the similarity between the two adjacent chromosomes. Fig 4 shows the fuzzy roulette wheel of the example in Table 2.

In the selection of chromosomes to form a mating pool, first, a random number is generated. Then the membership degree to which a generated random number belongs to each chromosome is calculated. The one with the highest membership degree is chosen. This process continues until a total of Psize chromosomes has been selected.

Chromosome	Solutions
1	O _{1,1} , O _{2,1} , O _{2,2} , O _{2,3} , O _{1,2} , O _{1,3} M ₂ , M ₂ , M ₃ , M ₁ , M ₁ , M ₁
2	O _{1,1} , O _{2,1} , O _{1,2} , O _{1,3} , O _{2,2} , O _{2,3} M ₁ , M ₁ , M ₁ , M ₁ , M ₁ , M ₃
3	O _{2,1} , O _{1,1} , O _{1,2} , O _{2,2} , O _{2,3} , O _{1,3} M ₂ , M ₁ , M ₂ , M ₂ , M ₂ , M ₁
4	O _{2,1} , O _{2,2} , O _{2,3} , O _{1,1} , O _{1,2} , O _{1,3} M ₃ , M ₂ , M ₃ , M ₂ , M ₂ , M ₁

Chromosome	Chromosome	Similarity value (Max score = 12)			Selected pair
		Operation	Machine	Total	
1	2	2	2	4	→ [1,4]
1	3	1	3	4	
1	4	2	3	5	

Fig. 3. The process of assigning chromosomes to slots of a roulette wheel

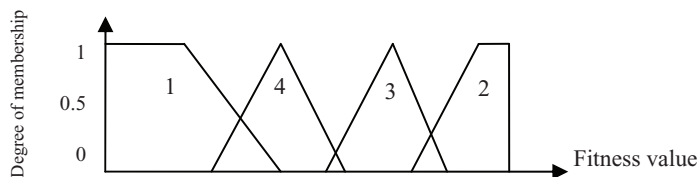


Fig. 4. Fuzzy roulette wheel

3.4. Crossover operation

Next, the proposed model employs the crossover operation to improve the quality of the solutions. This paper proposes a new crossover operator, which employs the concept of the hierarchical clustering algorithm. The followings are steps of the proposed crossover operator:

- 1) By using the fuzzy roulette wheel selection, create a mating pool of size Psize, where Psize is the predefined population size.
- 2) Partition the chromosomes in the mating pool into 2 clusters by using the hierarchical clustering algorithm. In this hierarchical clustering algorithm, the single linkage method as defined in eq. (1) is employed to calculate the similarity between two clusters.

$$S(G, H) = \underset{e \in G, f \in H}{MAX} [s(e, f)] \tag{1}$$

where G and H are any two clusters.

s(e,f) denotes the similarity between chromosomes e and f.

- 3) Randomly select one chromosome from each cluster to form a mating pair. Once a crossover point is determined, a one point crossover technique as shown in Fig 6 is used to create new offspring. This third step is repeated (Psize/2) – 1 more times.

For example, in Fig 5, four chromosomes are divided into two clusters. Cluster 1 consists of 3 chromosomes while cluster 2 consists of 1 chromosome. Two mating pairs are formed: 1-2 and 3-2. Fig 6 shows an example of the one point crossover operation which is performed on the mating pair 1-2.

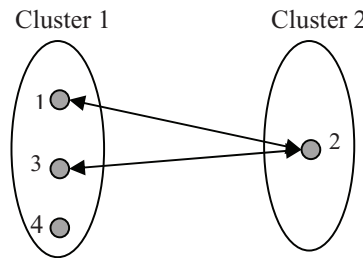


Fig. 5. Mating pair selection

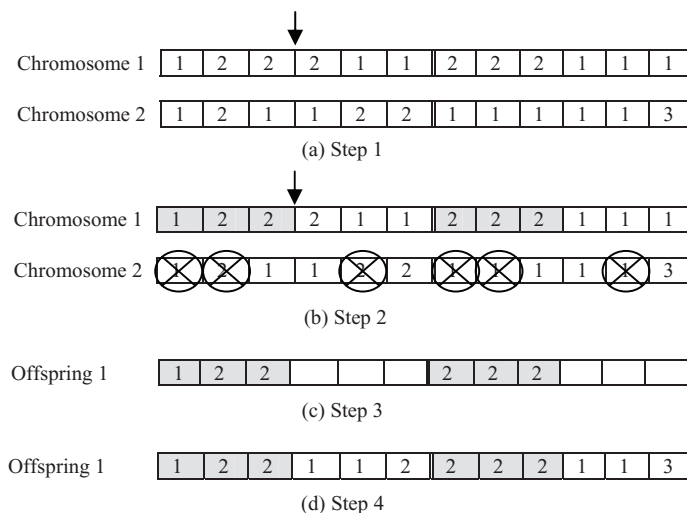


Fig. 6. Procedure of the one point crossover operation

3.5. Mutation operation

In this paper, the new mutation operator is employed to perform the mutation on the offspring. The mutation operation is used in order to maintain the diversity of the population and to overcome premature convergence. For each offspring q , the following mutation procedure is performed:

- 1) Randomly select two genes in the component 1 of the chromosome.
- 2) Swap the selected genes.
- 3) Evaluate the fitness of the mutated offspring. If the mutated offspring is fitter than its parent, it will replace its parent in the population. If not, the mutated offspring will be discarded.
- 4) This procedure is repeated T_q times, where T_q is the number of iterations in which the mutation operation is performed on the offspring q .

$$T_q = \max(1, r/(x_b - x_q)) \quad (2)$$

where r is a random number in the range $[-1,1]$.

x_b is the fitness value of the fittest chromosome in the current generation.

x_q is the fitness value of the chromosome q in the current generation.

3.6. Termination

The best Psize chromosomes are selected from the combined list of current chromosomes and newly generated offspring. Next, the termination criterion is checked. If the criterion is met, the elite chromosome is returned as the best solution found so far. If not, the whole procedure is repeated. The termination criterion used this study is to terminate the iteration of the GA when eighty percent of the population shares the same value.

4. Experimental Results

The proposed algorithm was tested on 10 benchmark problems taken from [12]. The performance of the proposed algorithm is then compared to that of the previous research works by Ho et al. [9], Girish and Jawahar [13, 14], and Xing et al. [15]. The system parameters of the proposed algorithm are set as follows: Psize = 200; Pc = 0.9; Pm = 0.3. The percentage relative error from the best known solution (RE) is used as a measure to evaluate the performance of the proposed method. Table 3 shows the experimental results of the proposed algorithm in comparison to all of the above mentioned methods. The average relative error of the proposed algorithm is the best among the compared methods.

5. Conclusion

This paper proposes a modification to the genetic algorithm for solving the flexible job-shop scheduling problem (FJSP). Three main operations of the genetic algorithm are modified: selection, crossover, and mutation operations. The experimental results show that the performance of the proposed model is significantly improved when these three new operations are used in place of the original operations typically used in the genetic algorithm. A much more complex version of the FJSP, which contains multi-objective, will be studied in the future work.

Table 3. Experimental results

Problems	n x m	LB	Ho et al. [9]	Girish and Jawahar [13]	Girish and Jawahar [14]	Xing et al. [15]	Proposed Model
MK01	10 x 6	36	40 (11.11)	40 (11.11)	40 (11.11)	39 (8.33)	40 (11.11)
MK02	10 x 6	24	29 (20.83)	26 (8.33)	27 (12.50)	29 (20.83)	27 (12.50)
MK03	15 x 8	204	-	204 (0.00)	204 (0.00)	204 (0.00)	204 (0.00)
MK04	15 x 8	48	67 (39.58)	66 (37.50)	62 (29.17)	65 (35.42)	64 (33.33)
MK05	15 x 4	168	176 (4.76)	174 (3.57)	178 (5.95)	173 (2.98)	175 (4.17)
MK06	10 x 15	33	67 (103.03)	77 (133.33)	78 (136.36)	67 (103.03)	65 (96.97)
MK07	20 x 5	133	147 (10.53)	143 (7.52)	147 (10.53)	144 (8.27)	144 (8.27)
MK08	20 x 10	523	523 (0.00)	523 (0.00)	523 (0.00)	523 (0.00)	523 (0.00)
MK09	20 x 10	299	320 (7.02)	328 (9.79)	341 (14.05)	311 (4.01)	309 (3.34)
MK10	20 x 15	165	229 (38.79)	247 (49.70)	252 (52.73)	229 (38.79)	234 (41.82)
Average Relative Error			23.57	26.08	27.24	22.17	21.15

References

1. D. E. Goldberg, *Genetic Algorithms in Search Optimisation and Machine Learning*, Addison-Wesley, Reading, Massachusetts, 1989.
2. F. Glover, *Tabu Search-Part I*, Operations Research Society of America, J. Comput., 1989.
3. S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, *Optimization by Simulated Annealing*, Sci., vol. 220 (1983) 671-680.
4. J. Kennedy and R. Eberhard, *Particle Swarm Optimization*, Phys.Rev., B.(1995) 13:5344-5348.
5. L. Zadeh, *Fuzzy sets, Information and Control*, 8(3) (1965) 338-353.
6. F. Pezzella, G. Morganti, G. Ciaschetti, *A genetic algorithm for the flexible job-shop scheduling problem*, Computers and Operations Research, 35 (10) (2008), 3202–3212.
7. G. H. Zhang, L. Gao, Y. Shi, *An effective genetic algorithm for the flexible job-shop scheduling problem*, Expert Systems with Applications, 38 (4) (2011), pp. 3563–3573.
8. N. Al-Hinai and T. Y. ElMekkawy, *An efficient hybridized genetic algorithm architecture for the flexible job-shop scheduling problem*, Flexible Services and Manufacturing Journal, Vol. 23, (2011), pp. 64-85.
9. N. B. Ho, J. C. Tay, and E. M. K. Lai, *An effective architecture for learning and evolving flexible job-shop schedules*, European Journal of Operational Research, vol.179, no. 2, (2007) 316-333.
10. I. Kacem, S. Hammadi, and P. Borne, *Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems*. IEEE Transaction on Systems, Man and Cybernetics, (2002) 32(1), 1-13.
11. W. Teekeng and A. Thammano, *A combination of Shuffled frog leaping algorithm and fuzzy logic for Flexible Job-Shop Scheduling Problems*, Procedia Computer Science, Complex adaptive systems, vol.6, (2011) 69-75
12. M. Mastrolilli, *Flexible Job Shop Problem*, <http://www.idsia.ch/~monaldo/fjsp.html>.
13. B. S. Girish and N. Jawahar, *Scheduling job shop associated with multiple routings with genetic and ant colony heuristics*, International Journal of Production Research, (2009), 47(14): 3891-3917.
14. B. S. Girish and N. Jawahar, *A particle swarm optimization algorithm for flexible job shop scheduling problem*, IEEE International Conference on Automation Science and Engineering, (2009), 298-303.
15. L. N. Xing, Y. W. Chen, P. Wang, Q. S. Zhao, and J. Xiong, *A knowledge-based ant colony optimization for flexible Job Shop scheduling problems*. Applied Soft Computing, 10, (2010) 888-896.