A Genetic Algorithm with an incomplete representation for the Job Shop Scheduling Problems

Y. Song†, JG. Hughes†, N. Azarmi‡ and C. Voudouris‡
† School of Info. & Soft. Eng., University of Ulster at Jordanstown, BT37 0QB
Email: (y.song, jg.hughes)@ulst.ac.uk , (nader.azarmi, chrisv)@bt.com
‡ BT Intelligent System Research Group, pp12, MLB1, BT Labs, Ipswich, IP5 3RE

Abstract

Most of the GA approaches for job shop scheduling problem (**JSSP**) represent a solution by a chromosome containing the sequence of all the operations and decode the chromosome to a real schedule from the first gene to the last gene. There are two common problems for this kind of GAs, namely, high redundancy at the tail of the chromosome and little significance of rear genes on the overall schedule quality. GA-operators (e.g. the 1-point, 2-point crossover, and some mutation operators, etc.) applied on the real part of the chromosome (only involving the change of the real part of a chromosome) are less likely to create good offsprings, i.e., most likely a waste of evolution (time). In this paper, we propose a genetic algorithm with an incomplete representation (the number of genes is less than the number of operations) and apply it to the JSSPs. In our approach, the most important and the largest part of a schedule is decoded from a chromosome and the rest of the schedule is completed by a simple heuristic rule.

1. Introduction

The Job-shop scheduling problem ($n/m/J/C_{max}$) is concerned with ordering *n* jobs on *m* machines [7]: each job is composed of an ordered list of *m* operations and must be processed during a uninterrupted time period of a given length on each of the *m* machines. The objective is to minimise the makespan, i.e. the maximum of job completion times. Since the publication of Davis' paper in 1985 [5] a lot of research in the field of production scheduling with GAs has been done. The main difficulty in this subject arises from the question of how to represent the problem in the algorithm, which is known to be most important for genetic search [4]. So far, researchers have proposed many representation approaches together with different kinds of GA operators to tackle the problems. A tutorial survey of the representation approaches in the literature can be found in [6]. Among them, some of the most well-known representation approaches are the priority rule-based representation [3], the random keys representation [2], and the operation-based representation, such as [1, 4]. Some result comparison of these approaches is given in [2].

The incomplete representation (IR) is mainly based on the operation-based representation. The difference is that its chromosomes use fewer genes than those of the normal operation-based representation. In IR-GA, the number of genes is less than the number of operations of the JSSP, and the most important and the largest part of a schedule is decoded from a chromosome and the rest of the schedule is completed by a simple heuristic rule (more details and examples can be seen in the next section). The normal length (no. of total operations) of the chromosome used in most GA approaches is reduced by 20-30% in our approach. This looks like we are considering the original problem as a problem with a smaller size compared with other complete-representation methods. Theoretically, this kind

of simplification may affect the optimal (or best) result of the GA approach when there is no running time limit. In real world, optimal scheduling is not very important as most of the real-word scheduling problems are too large to find their optimal solutions within reasonable time and the dynamic nature of the problem makes frequent rescheduling necessary. In most of the cases we need to find a good or near-optimal solution for a problem in a reasonable time, i.e., to balance the computation time and solution quality. The proposed incomplete representation is designed to find as good average result as the complete-representation with less evolving generation and to be used in cases where fast approximate solutions are needed.

2. **The incomplete representation (IP) for JSSP**

In order to implement and test the IP approach, we used the GA encoding and decoding approach proposed by Fang et al. [1]. In Fang's approach, a *n*-job and *m*-machine JSSP is represented by a chromosome with a string containing *n* x *m* chunks which are large enough to hold the largest job number for the problem. Each chunk is interpreted as the first unscheduled operation of the job specified by the job number at the junk (if all operations of the job have been scheduled, then interpreted it as the next job. See more details in [1]), and the first unscheduled operation belonging to the job at each position in order is scheduled on the designated machine at the earliest time it can be started. For example, following is a typical chromosome for the 3x3 JSSP in table1:

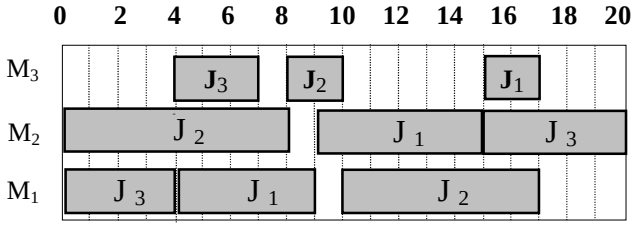
3	2	3	1	2	1	2	1	3

Table 2 is a schedule decoded from the above chromosome and the last three genes have no effects on the schedule. Any values for the last three genes will be decoded as the same schedule as table 2. The greyed part of the chromosome completely decides the schedule. There are a lot of chromosomes like this in the searching space, and most of them are not good ones. In these cases, when GA operators are applied on the rear part, it is unlikely to create new or better solutions. The IP-GA proposed in this paper is based on this observation. Instead of searching a whole pattern as above, IP-GA searches for a pattern as the greyed part of the chromosome and so use fewer genes than the complete representation. Figure 1 shows the difference between the complete and the incomplete GA representation.

Table 1: a 3 job and 3 machine JSSP. (M_i , n): stands for an operation to be performed on machine i for n time units.

Job name	1st operation	2nd operation	3rd operation
J_1	(M_1 , 5)	(M_2 , 6)	(M_3 , 2)
J_2	(M_2 , 8)	(M_3 , 2)	(M_1 , 7)
J_3	(M_1 , 4)	(M_3 , 3)	(M_2 , 5)

Table 2: A schedule (solution) decoded from the typical chromosome for the sample problem. The makespan is 20 (optimal).



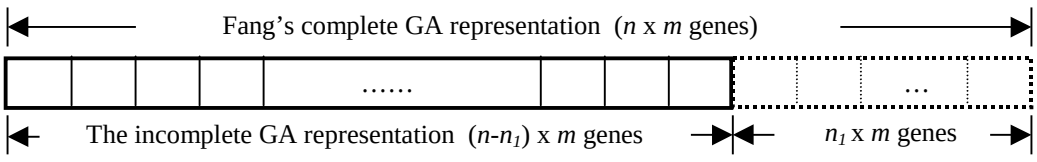


Figure 1. The difference of complete and incomplete GA representation

The incomplete representation discards the n_1 x m genes at the rear part of the chromosome which are known having high redundancy and little significance. In the incomplete representation, the first (n - n_1) x m operations are decoded according to the chromosome and the rest n_1 x m operations are decoded by a simple heuristic rule. The heuristic rule used in this approach selects the next activity (from those unscheduled by GA) with minimal latest end time from those can be started first and have a minimal earliest start time. Initial experimental result show that n_1 / n should be less than 30% and the exact value should be slightly varied with specific problems (n_1/n is problem specific). The heuristic rule doesn't take more time than the GA to decode the rest of the (n - n_1) x m operations.

3. Initial Experimental Results

We have implemented the IP-GA in Visual C++ 5.0 and used Matthew's GA-lib version 2.4 at http://lancet.mit.edu/galib-2.4/ as our starting point. For convenience of implementation we used ILOG Solver 3.2 and ILOG Scheduler 2.2, which are both C++ libraries, to assist the decoding process. However, ILOG libraries are not directly involved with problem solving and have no effect on the final performance. The approach has been tested on the two benchmark problems, FM20(20x5) and FM10(10x10) that can be obtained from the OR-Library at http://mscmga.ms.ic.ac.uk/jeb/orlib/jobshopinfo.html. The GA operators and parameters used in the experiment are described in table 3, and the GA-objective is the same as makespan of the schedule, the maximum completion time of all the jobs. 100 runs have been made for each of the test to get the average result and the initial population was randomly created with random seed. Table 4 and table 5 give the results of our experiment.

Table 3. GA operators and parameters used in the experiment

Basic GA	*Steady-State*	Crossover Rate	*0.8*
Scaling Method	*Linear Scaling*	Mutation Rate	*0.005*
Crossover Method	*2-Point Crossover*	Replacement Rate	*10%*
Mutation Method	*Swap Mutation*	Selection Scheme	*Rank Selector*

Table 4. Average results on FM20

* p x g / * n_1/n	50 x100	100 x20	150 x30	200 x400	250 x500	300 x600
0%	1369.5	1333.4	1321.3	1303.24	13015	1295.6
20%	1359.2	1327.1	1312.0	1303.0	1295.4	1288.3
30%	1376.2	1342.7	1326.0	1314.2	1306.0	1297.5

Table 5. Average results on FM10

p x g / n_1/n	50 x100	100 x200	150 x300	200 x400	250 x500	300 x600
0%	1076.2	1050.3	1046.0	1035.5	1030.3	1019.1
20%	1045.9	1024.2	1019.5	1013.1	1008.3	1001.5

(* n_1/n: percentage of genes cut from the complete representation, p x g : *population* x *generation*)

In table 4, the best average results are obtained by n_1/n = 20%. However, the results of n_1/n=30% is not as good as those of n_1/n=0 (the complete representation, i.e. Fang's basic approach). This is probably because we cut the tail too much for this problem. In table 5, there is a great improvement on average results by incomplete representation over complete representation. The average results of incomplete representation with *population* x

generation of 50x100, 100x200 and 150x300, are comparable with those of complete representation with *population* x *generation of* 150x300, 250x500 and 300x600 respectively. In this case, the incomplete representation is much more efficient than the complete representation.

4. Conclusion

We propose a genetic algorithm with an incomplete representation and apply it to the job-shop scheduling problems. Initial experimental results show that the incomplete representation is promising and useful when GA is needed to find near-optimal or reasonable solutions in a reasonable computation time. By cutting the genes with high redundancy and little significance at the rear part of the chromosome, we believe GA could be more efficient because useless (most likely) GA operators applied on the rear part of the chromosome are automatically avoided. However, this approach still need more tests on more benchmark problems with different GA parameters and how to decide the exact n_1/n (the radio of discarded genes to the total number of genes in a complete representation) of the incomplete GA representation for a particular problem is still unknown.

Acknowledgements

This work is partially supported by The Overseas Research Students Awards Scheme. The authors would like to thank Matthew at MIT who made his GAlib available and J E Beasley at Imperial College who maintains the OR-Library.

References

1. H. Fang, Peter Ross and Dave Corne. A promising genetic algorithm approach to job-shop scheduling, rescheduling, and open-shop scheduling problems. Proc. Of the Fifth International Conference on Genetic Algorithms, Urbana-Champaign, IL, USA, July 1993, pp.373-382.

2. B.Norman and J.Bean, Random keys genetic algorithm for job-shop scheduling, Engineering Design & Automation Vol. 3, No. 2, pp.145-156, 1997.

3. U. Dorndorf and E. Pesch, Evolution based learning in a job-shop environment, Computers and Operations Research, Vol. 22, pp.25-40, 1995.

4. C. Bierwirth, A generalized permutation approach to job-shop scheduling with genetic algorithms. OR SPEKTRUM, Vol. 17, N0. 2-3, pp.87-92, 1995.

5. L. Davis, Job-shop scheduling with genetic algorithm., Proc. Of the 1st Int. Conf. On Genetic Algorithms, Lawrence Erlbaum Associates (1985) pp. 136-140.

6. R. Cheng and M. Gen, A tutorial survey of job-shop scheduling problems using genetic algorithms—I. Representation, Computers ind. Engng., Vol. 30, No. 4, pp. 983-997, 1996.

7. S. French, Sequencing and Scheduling, Halstead Press; 1982.