Available online at www.sciencedirect.com

ScienceDirect

Computers & Industrial Engineering 53 (2007) 313–320

computers & industrial engineering

www.elsevier.com/locate/dsw

Integration of genetic algorithm and Gantt chart for job shop scheduling in distributed manufacturing systems

H.Z. Jia, J.Y.H. Fuh *, A.Y.C. Nee, Y.F. Zhang

Department of Mechanical Engineering, National University of Singapore, 9 Engineering Drive 1, Singapore 117576, Singapore

Available online 15 June 2007

Abstract

In a distributed manufacturing environment, jobs in a batch could usually be manufactured in several available factories and thus have multiple alternative process plans. This paper presents a new approach to determine good combinations of factories (process plans) to manufacture the jobs and in the meantime generate good operation schedules. A genetic algorithm (GA), integrated with Gantt chart (GC), is proposed to derive the factory combination and schedule. The integration of GA–GC is proved to be efficient in solving small-sized or medium-sized scheduling problems for a distributed manufacturing system. Multiple objectives can be achieved, including minimizing makespan, job tardiness, or manufacturing cost. An illustrative example is given to demonstrate and evaluate the performance of the GA–GC approach.
© 2007 Published by Elsevier Ltd.

Keywords: Genetic algorithm; Gantt chart; Production scheduling

1. Introduction

Traditionally, process planning focuses on the selection of the machines and set-up of machine parameters for job operations, while production scheduling deals with the assignments of machines as well as manufacturing times to process the job operations. With the prevailing trends of globalized manufacturing in recent years, the production of a part would have several feasible process plans, each for a different geographically distributed factory. Accordingly, job scheduling in such a distributed manufacturing system becomes more difficult than conventional job scheduling.

Determining an efficient schedule for general job shop problems has been the subject of research for several decades. The generation of scheduling plans for job shops has been literarily addressed using optimization approaches (for example, Potts & Wassenhove, 1985; Schrage & Baker, 1978; Yano & Kim, 1991) or approximation approaches (for example, Dauzere-Peres & Paulli, 1997; Mastrolilli & Gambardella, 1998; Verma & Dessouky, 2000). Among the various solution approaches to different scheduling problems, there has been an increasing interest in applying genetic algorithms (GAs) to solve the engineering optimization including

* Corresponding author. Fax: +65 6779 1459.
E-mail address: mpefuhyh@nus.edu.sg (J.Y.H. Fuh).

0360-8352/$ - see front matter © 2007 Published by Elsevier Ltd.
doi:10.1016/j.cie.2007.06.024

manufacturing scheduling (for example, Fang, Ross, & Corne, 1993; Hou, Ansari, & Ren, 1994; Zalzala & Fleming, 1997; Park, Choi, & Kim, 2003; Srikanth & Barkha, 2004). Frequently, genetic algorithms are able to outperform many optimization or approximation approaches.

When dealing with the scheduling problem in a distributed manufacturing system, in which one job could be manufactured in several available factories, and thus have multiple alternative process plans, the commonly used genetic algorithms appear not flexible enough to choose a good factory (process plan) combination and in the meantime give good job processing schedule. Considering the above aspects, this paper addresses an innovative GA, which applies crossover operator once and mutation operator twice to simultaneously improve the factory selections and job sequence selections. When integrated with Gantt chart (GC), the proposed genetic algorithm could generate good scheduling results with different objectives including makespan minimization, cost minimization, etc. in a reasonable computational time. Jia, Nee, Fuh, and Zhang (2003) proposed a modified GA to solve the job shop scheduling problems in a distributed manufacturing environment. The GA developed in this paper is an extension of the pervious work and it uses the integration of the GA with the GC in order to solve the small-sized or medium-sized scheduling problems in an more efficient and effective manner. The rest of the paper is organized as follows. In Section 2, we introduce the characteristics of a general distributed manufacturing system. In Section 3, the proposed GA is described in detail. Moreover, the integration of the GA and the GC is presented. Section 4 gives an illustrative example to demonstrate the use of the proposed GA in solving the scheduling problems for distributed manufacturing systems. Finally, we give conclusions in Section 5.

2. Description of the distributed manufacturing system

In a distributed manufacturing system, multiple factories can be selected to manufacture the jobs. Each factory that is available to process the jobs may have different machines and machine available times. In addition, each job involves many processing operations based on the features of the part and has different process plans, each suited for an available factory. To generate the manufacturing schedule in such a distributed manufacturing environment, two problems need to be considered:

- The selection of the factory (process plan) for each job; and
- The assignments of the machines and the times to process the operations of each job.

To address these two problems, we use the following notations to represent the elements in the distributed manufacturing system including the jobs, machines, and factories.

J_i = the ith job in the job batch;
J_iO_j = the jth operation of job i;
F_k = the kth factory that is available to process the jobs;
F_kM_q = the qth machine in factory k;
$PP - J_iF_k$ = the process plan of factory k in processing job i.

With these notations, the relationship among the jobs, machines, and factories in a distributed manufacturing system can be depicted as follows in Fig. 1.

3. Proposed genetic algorithm

3.1. Representation of selected factories and job operations

In a distributed manufacturing environment, where jobs will be dispatched to many factories, the encoding of the scheduling problem becomes more complex since the chromosomes have to comprise more information including the selected factory for every job as well as the job's operation sequence. One advantage of the GA is that it treats the factors of a problem as genes, and then arranges the genes in all possible ways to obtain the

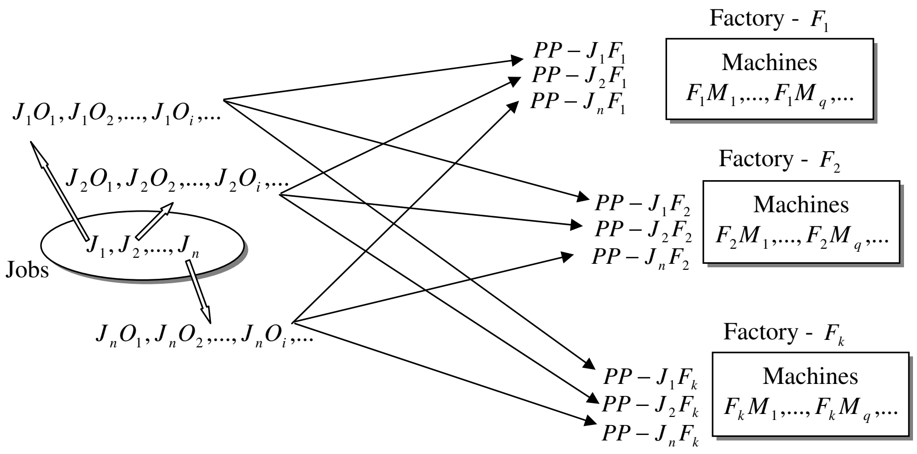


Fig. 1. Relationship among jobs, process plans, machines, and factories.

good solution. To deal with the distributed manufacturing system, a gene in the GA must comprise two determinant factors, i.e., the selected factory with the corresponding process plan and the job operation. In the proposed GA, we use a three digit string as a gene to represent a factory and an operation of a job. The first digit is the identification number of the selected factory, which is used to process the job, and the next two digits represent an operation of a job ID. Thus, the operations of all the jobs can be represented by a random combination of genes, which is called chromosome in GA terminology, as long as the combination comprises the selected factories as well as the operations of all the jobs.

Every job is bound to a series of sequential operations. To make the chromosome comply with the precedence of the operations, all the operations of a job are represented using the same symbol, i.e., the job ID, and then interpreted according to the order of occurrence in the sequence for a given chromosome (Gen & Cheng, 1997). Thus, every chromosome can be understood to be an encoded feasible schedule for a specific distributed scheduling problem, with the sub-schedules for all the factory participants. For example, the chromosome, '103-202-202-101-204-103-202-101', indicates that there are all together two factories and four jobs. The IDs of the factories are '1' and '2', the first digit of each gene, and the IDs of the jobs are '01', '02', '03', and '04', the last two digits of each gene. Job 01, 02, 03, and 04 will be, respectively, manufactured in factory 1, factory 2, factory 1, and factory 2. According to every job gene's appearing times in the chromosome, it can be concluded that these four jobs, respectively, have two, three, two, and one operations. The processing sequence of the job operations is in accordance with their genes' appearing sequence in the chromosome.

3.2. Genetic operators

Much of the power of GA arises from the recombination of genes, including crossover, mutation, and invasion, which explores all the possible solution space. For the GA proposed in this paper, an initial chromosome population will be first generated randomly. Then, two genetic operators, crossover and mutation, are used for the recombination of genes, which is called offspring generation.

3.2.1. Crossover operator

The procedures for the crossover operation is described as follows, and illustrated by an example.

(1) Randomly choose two chromosomes as parents and exchange a partial string (genes) in the two parents to generate two offspring chromosomes.
(2) Delete or add genes to legalize the offspring chromosomes, which should comprise all the operations of all the jobs and inherit the genetic traits of their parents.

Note that there can be different ways to legalize the offspring chromosomes after the genes have been exchanged in the parent chromosomes. In the proposed GA, we always add the missing genes (job operations) and delete the redundant genes at the end of each offspring chromosome (Table 1).

3.2.2. Mutation operator

The primary purpose of mutation is to introduce variation and help bring back some essential genetic traits, and also to avoid the pre-mature convergence of entire feasible space caused by some super chromosomes.

The traditional mutation procedure of the GAs is to exchange the positions of a number of randomly selected genes or change the randomly selected genes to different ones. However, for the distributed manufacturing system, mutating gene only once is not enough to investigate different job operation sequences as well as different factory selection combinations. To address this problem, we adopt the mutation operator twice in the proposed GA, one for job operation sequence mutation and the other for factory selection mutation. The procedures are depicted as follows and an example is given in Table 2.

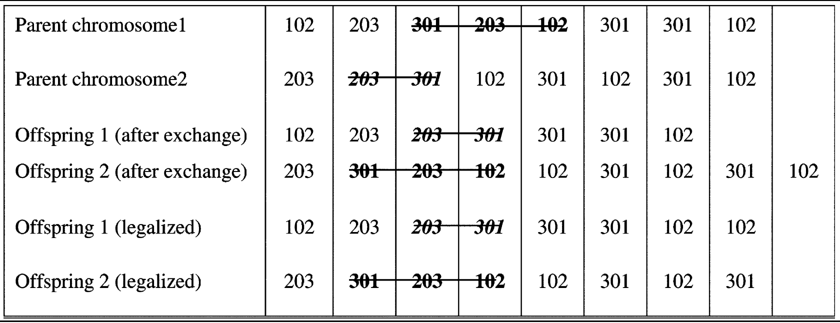
(1) Choose a number of gene pairs stochastically and then permute their positions (mutation 1);
(2) Randomly select one gene (job) from the chromosome;
(3) For the selected gene (job), randomly select a factory that is available to process the job;
(4) Change the first digit (represents the ID of the selected factories) of all the genes, which are the same as the gene in step 2, to the newly selected factory ID number (mutation 2).

In the two mutation procedures, mutation 1 is set to be a local GA operator, which has an effect only within the selected chromosomes. It aims to investigate different job operation sequences. Mutation 2 is set to be a global GA operator, and thus an effect to all the chromosomes in the generation. It is used to explore the different factory selection combinations for processing the jobs.

3.3. Chromosome fitness evaluation

This procedure calculates the fitness or the relative fitness value of the chromosome for elite chromosome selection proposes.

Table 1
A crossover example



Parent chromosome1	102	203	~~**301**~~	~~**203**~~	~~**102**~~	301	301	102	
Parent chromosome2	203	~~***203***~~	~~***301***~~	102	301	102	301	102	
Offspring 1 (after exchange)	102	203	~~***203***~~	~~***301***~~	301	301	102		
Offspring 2 (after exchange)	203	~~**301**~~	~~**203**~~	~~**102**~~	102	301	102	301	102
Offspring 1 (legalized)	102	203	~~***203***~~	~~***301***~~	301	301	102	102	
Offspring 2 (legalized)	203	~~**301**~~	~~**203**~~	~~**102**~~	102	301	102	301	

Table 2
A mutation example

Chromosome (before mutation 1)	102	203	***301***	***203***	***102***	301	301	***102***
Chromosome (after mutation 1)	102	203	***102***	***102***	***203***	301	301	***301***
Chromosome (before mutation 2)	102	203	**102**	102	203	301	301	301
Chromosome (after mutation 2)	**302**	203	**302**	**302**	203	301	301	301

3.3.1. Objective

The efficiency of the manufacturing system depends on many factors such as machine utilization, tardiness, manufacturing makespan, cost, etc. One or more of these factors are employed to measure the efficiency of manufacturing systems. In the current competitive market, more focuses of manufacturing are put on finishing the jobs in the minimal time with least cost. As such, in this paper, the objectives of scheduling are targeted as minimizing makespan, job tardiness, or production cost. Based on the problems that are specifically considered, the production cost may include the manufacturing cost only or include both the manufacturing cost and the transportation cost since the parts need to be delivered among the factories.

3.3.2. Chromosome translation using Gantt chart

Chromosome translation is the process of converting the chromosome into a real schedule. In this research, the GC is used for mapping the chromosome into a readable schedule and it is particularly efficient and effective for small-sized and medium-sized scheduling problems. Because every job is bound to a sequence of operations, when mapping the chromosome into the GC schedule, the following constraints should be satisfied.

(1) The mapped gene (operation), which could be manufactured in a machine different from the job's preceding operation, must be put in a position behind its preceding operation;
(2) The scale of each gene in the chart must be in accordance with the manufacturing time or cost of the job operation. Such a scale-based mapping will give convenience for the overall chromosome fitness evaluation;
(3) Because each machine can merely process one operation at one time, any gene in the GC cannot overlap with other genes;
(4) In order to obtain the minimal makespan or cost, all the mapped genes on the Gantt chart schedule must be as compact as possible.

The complete chromosome translating process is depicted in Fig. 2.

3.3.3. Gantt chart schedule evaluation

Based on the scheduling objective that is of interest, the fitness of the mapped chromosome can be readily evaluated from the Gantt chart. If makespan minimization is the objective, the fitness will be the completion time of the last operation in the Gantt chart. If the cost minimization is the objective, then the fitness would be the sum of the manufacturing cost and idling cost of all the machines, plus the necessary transportation cost for the job delivery among the factories.

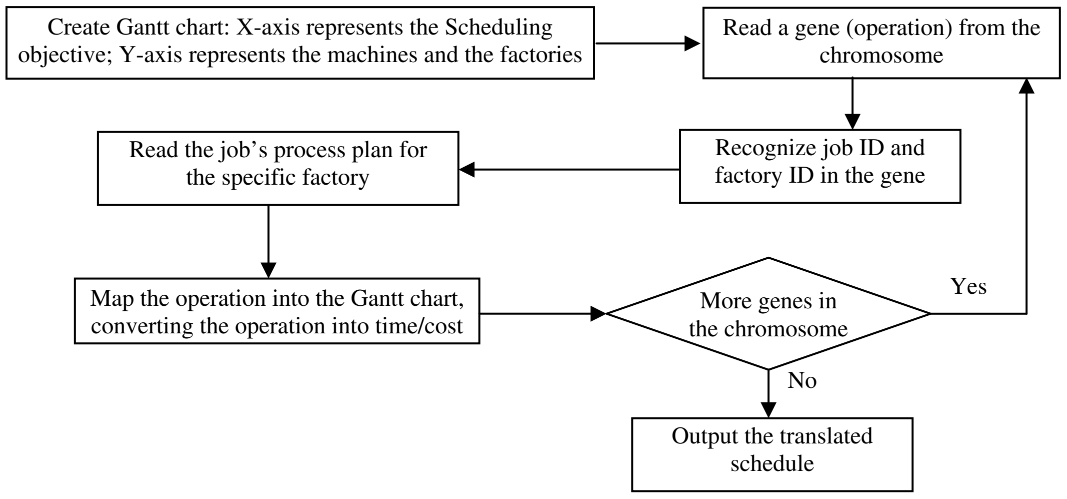

Create Gantt chart: X-axis represents the Scheduling objective; Y-axis represents the machines and the factories
Read a gene (operation) from the chromosome
Recognize job ID and factory ID in the gene
Read the job's process plan for the specific factory
Map the operation into the Gantt chart, converting the operation into time/cost
More genes in the chromosome
Yes
No
Output the translated schedule

Fig. 2. Procedures of the chromosome translation using Gantt chart.

3.4. Termination conditions

There are two conditions set to terminate the GA iteration. When any of the two conditions is satisfied, the GA will terminate and the best chromosome, together with the corresponding schedule, will be outputted. The two conditions are:

(1) The best found chromosome, which is kept in memory using a variable, has been continuously obtained for 10 chromosome generations;
(2) A pre-defined GA iteration number has been executed.

4. An illustrative example

In this section, we demonstrate the application of the proposed GA–GC approach and evaluate its computational performance by using a distributed scheduling problem that involves 3 factories, 12 machines, and 6 jobs. The detailed scheduling problem is given in Table 3.

In this table, $PP - J_iF_k$ denotes the process plan for job i in factory k; F_kM_q refers to the machine q in factory k, which is used to manufacture the specific operation of a job; and the number in the parentheses represents the time spent on processing the operation.

In order to investigate the case better, some jobs' process plans are intentionally set to be obviously better or poorer than their other parallel process plans. From Table 3, it can be seen that the process plan in the third factory for job 2 and the process plan in the first factory for job 3 are poorer than the two jobs' process plans in other two factories since their manufacturing time is much longer. The process plan in the third factory for job 4, however, is better than its process plans in the first and second factory because of the shorter manufacturing time. Given such a setting, an optimal result for this scheduling problem should not choose the poorer process plans (factories) for job 2 and job 3, but should choose the better process plan (factory) for job 4.

We use the proposed GA to solve the scheduling problem and apply the GC to translate the chromosomes into feasible scheduling solutions and evaluate the fitness of the chromosomes. The obtained final solution from the GA–GC approach is depicted in Fig. 3.

Table 3
A distributed scheduling problem

Jobs	Factory (process plan)	Operation 1	Operation 2	Operation 3	Operation 4
Job-1 (J_1)	Factory 1 ($PP - J_1F_1$)	F_1M_1 (2)	F_1M_3 (1)	F_1M_2 (3)	F_1M_4 (3)
	Factory 2 ($PP - J_1F_2$)	F_2M_1 (1)	F_2M_3 (3)	F_2M_2 (4)	F_2M_4 (4)
	Factory 3 ($PP - J_1F_3$)	F_3M_1 (2)	F_3M_3 (4)	F_3M_4 (2)	F_3M_2 (3)
Job-2 (J_2)	Factory 1 ($PP - J_2F_1$)	F_1M_2 (3)	F_1M_3(2)	F_1M_1 (3)	F_1M_4 (4)
	Factory 2 ($PP - J_2F_2$)	F_2M_2 (2)	F_2M_1 (2)	F_2M_3 (3)	F_2M_4 (4)
	Factory 3 ($PP - J_2F_3$)	***F_3M_1 (10)***	***F_3M_4 (10)***	***F_3M_3 (15)***	***F_3M_2 (8)***
Job-3 (J_3)	***Factory 1 ($PP - J_3F_1$)***	***F_1M_3 (10)***	***F_1M_1 (10)***	***F_1M_2 (10)***	***F_1M_4 (6)***
	Factory 2 ($PP - J_3F_2$)	F_2M_3 (1)	F_2M_2 (2)	F_2M_1 (4)	F_2M_4 (3)
	Factory 3 ($PP - J_3F_3$)	F_3M_4 (3)	F_3M_3 (1)	F_3M_1 (4)	F_3M_2 (2)
Job-4 (J_4)	Factory 1 ($PP - J_4F_1$)	F_1M_2 (6)	F_1M_3 (8)	F_1M_1 (9)	F_1M_4 (10)
	Factory 2 ($PP - J_4F_2$)	F_2M_3 (7)	F_2M_2 (7)	F_2M_1 (8)	F_2M_4 (7)
	Factory 3($PP - J_4F_3$)	***F_3M_1 (2)***	***F_3M_4 (3)***	***F_3M_2 (2)***	***F_3M_3 (3)***
Job-5 (J_5)	Factory 1 ($PP - J_5F_1$)	F_1M_2 (3)	F_1M_4 (2)	F_1M_1 (1)	F_1M_3 (4)
	Factory 2 ($PP - J_5F_2$)	F_2M_4 (3)	F_2M_3 (2)	F_2M_2 (2)	F_2M_1 (3)
	Factory 3 ($PP - J_5F_3$)	F_3M_1 (5)	F_3M_3 (1)	F_3M_2 (2)	F_3M_4 (2)
Job-6 (J_6)	Factory 1 ($PP - J_6F_1$)	F_1M_3 (4)	F_1M_4 (3)	F_1M_1 (2)	F_1M_2 (1)
	Factory 2 ($PP - J_6F_2$)	F_2M_3 (5)	F_2M_2 (3)	F_2M_1 (1)	F_2M_4 (2)
	Factory 3 ($PP - J_6F_3$)	F_3M_4 (2)	F_3M_1 (3)	F_3M_3 (2)	F_3M_2 (3)

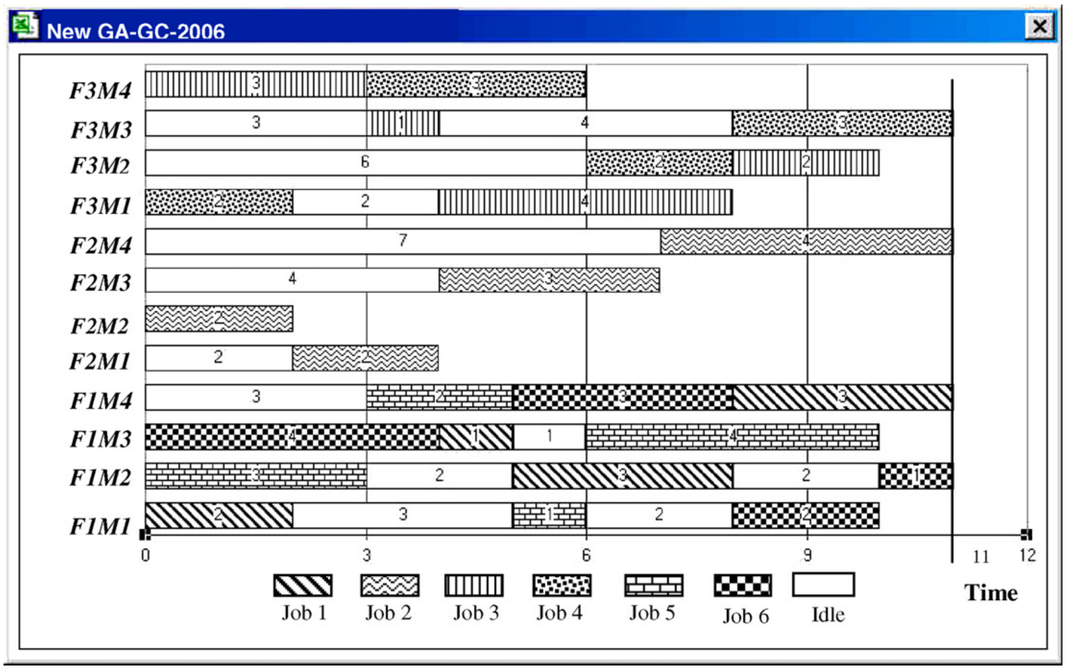


Fig. 3. Scheduling result.

From the results, it can be seen that factory 1 is selected to manufacture job 1, job 5, and job 6; factory 2 is selected to manufacture job 2; and factory 3 is selected to manufacture job 3 and job 4. This result is in accordance with the pervious optimal result analysis. The GA–GC approach selects factory 3 (with the best process plan) for job 4, and does not select factory 3 (with the poorest process plan) for job 2 and the factory 1 (with the poorest process plan) for job 3. Furthermore, it can be seen that, with such a job operation schedule, the makespan for manufacturing all the jobs is 11 time units. All the factories complete the manufacture of the job operations at the same time (i.e., at 11th time unit), and this indicates that the workloads of all factories are well balanced.

The CPU time used for the GA–GC approach to solve the problem with an environment of PIII-1.5G/512M-RAM is 10 s. This computational time is 50% shorter than the time (15 s) consumed by using the GA alone (without GC integration) developed by Jia et al. (2003) to solve the same problem under a similar environment. Furthermore, the problem is also solved to optimality by using CPLEX, a commercially available software for linear/integer programming problems, and the same scheduling result has been obtained. However, CPLEX uses a much longer computational time (22 s) to solve the problem.

5. Conclusions

Due to the trend of globalized job outsourcing, job scheduling in a distributed manufacturing environment has become much more complicated than its traditional counterpart. The challenges faced by the scheduling in a distributed manufacturing environment require the job scheduler to not only sequence the job operations in an optimal way, but also select the best factory combinations to manufacture the jobs. In this paper, a modified GA, integrated with GC, is proposed and implemented for dealing with the newly emerged distributed scheduling problems. The information of the selected factories for the jobs and the job operations are encoded, respectively, and then combined to form the genes. The mixture of all the jobs' genes makes up a chromosome that represents a feasible schedule for the distributed scheduling problem, including the sub-schedules for all the available factories. "Once" gene crossover and "twice" gene mutation are employed, respectively, in an attempt to change the operations' processing sequence and the selected factories for the jobs. The GC is applied to efficiently and effectively translate the chromosomes into feasible schedules. Experimental results show that the application of the GC is able to facilitate the chromosome evaluation procedures and thus improve the computational performance of the proposed GA.

References

Dauzere-Peres, S., & Paulli, J. (1997). An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search. *Annals of Operations Research, 70*, 281–306.

Fang, H. L., Ross, P., & Corne, D. (1993). A promising genetic algorithm approach to job-shop scheduling, rescheduling, and open-shop scheduling problems. In *Proceedings of the fifth international conference on genetic algorithms* (pp. 375–382). M. Kaufmann Publishers.

Gen, M., & Cheng, R. (1997). *Genetic algorithms and engineering design*. New York: John Wiley & Sons, Inc..

Hou, E. S. H., Ansari, N., & Ren, H. (1994). A genetic algorithm for multiprocessor scheduling. *IEEE Transactions on Parallel and Distributed Systems, 5*, 113–120.

Jia, H. Z., Nee, A. Y. C., Fuh, J. Y. H., & Zhang, Y. F. (2003). A modified genetic algorithm for distributed scheduling problems. *Journal of Intelligent Manufacturing, 14*, 351–363.

Mastrolilli, M., & Gambardella, L. M. (1998). Effective neighborhood functions for the flexible job shop problem. *Journal of Scheduling, 3*, 3–20.

Park, B. J., Choi, H. R., & Kim, H. S. (2003). A hybrid genetic algorithm for the job shop scheduling problems. *Computers and Industrial Engineering, 45*, 597–613.

Potts, C., & Wassenhove, L. V. (1985). A branch and bound algorithm for the total weighted tardiness problem. *Operations Research, 33*, 363–377.

Schrage, L., & Baker, K. (1978). Dynamic programming solution of sequencing problems with precedence constraints. *Operations Research, 26*, 444–449.

Srikanth, K. I., & Barkha, S. (2004). Improved genetic algorithm for the permutation flowshop scheduling problem. *Computers and Operations Research, 31*, 593–606.

Verma, S., & Dessouky, M. (2000). Single machine scheduling of unit time jobs with earliness and tardiness penalties. *Mathematics of Operations Research, 23*, 930–943.

Yano, C., & Kim, Y. (1991). Algorithms for single machine scheduling problems minimizing tardiness and earliness. *European Journal of Operational Research, 52*, 167–178.

Zalzala, A. M. S., & Fleming, P. J. (1997). *Genetic algorithms in engineering systems*. London: The Institution of Electrical Engineers.