

Solving distributed FMS scheduling problems subject to maintenance: Genetic algorithms approach

Felix T.S. Chan^{a,*}, S.H. Chung^a, L.Y. Chan^a, G. Finke^b, M.K. Tiwari^c

^a*Department of Industrial and Manufacturing Systems Engineering, The University of Hong Kong, Pokfulam Road, Hong Kong*

^b*LEIBNIZ—IMAG, University Joseph Fourier, France*

^c*Department of Manufacturing Engineering, National Institute of Foundry and Forge Technology, India*

Received 7 October 2005; accepted 25 November 2005

Abstract

In general, distributed scheduling problem focuses on simultaneously solving two issues: (i) allocation of jobs to suitable factories and (ii) determination of the corresponding production scheduling in each factory. The objective of this approach is to maximize the system efficiency by finding an optimal planning for a better collaboration among various processes. This makes distributed scheduling problems more complicated than classical production scheduling ones. With the addition of alternative production routing, the problems are even more complicated. Conventionally, machines are usually assumed to be available without interruption during the production scheduling. Maintenance is not considered. However, every machine requires maintenance, and the maintenance policy directly affects the machine's availability. Consequently, it influences the production scheduling. In this connection, maintenance should be considered in distributed scheduling. The objective of this paper is to propose a genetic algorithm with dominant genes (GADG) approach to deal with distributed flexible manufacturing system (FMS) scheduling problems subject to machine maintenance constraint. The optimization performance of the proposed GADG will be compared with other existing approaches, such as simple genetic algorithms to demonstrate its reliability. The significance and benefits of considering maintenance in distributed scheduling will also be demonstrated by simulation runs on a sample problem.

© 2006 Elsevier Ltd. All rights reserved.

Keywords: Distributed scheduling; Flexible manufacturing systems; Genetic algorithms; Maintenance

1. Introduction

The significance of distributed scheduling (DS) has been recognized by many researchers and industrialists in recent years because of the changes in the mode of today's production environment. Single factory production in traditional manufacturing has been gradually replaced by multi-factory production due to the trend of globalization. These factories may be geographically distributed in different locations, which allow them to be closer to their customers, to comply with the local laws, to focus on a few product types, to produce and market their products more effectively, and to be responsive to market changes more quickly [1,2]. Each factory is usually capable of manufac-

turing a variety of product types. Some may be unique in a particular factory, while some may not. In addition, they may have different production efficiency and various constraints depending on the machines, labor skills and education levels, labor cost, government policy, tax, nearby suppliers, transportation facilities, etc. This induces different operating costs, production lead time, customer service levels, etc. in different factories [3–6].

DS problems are much more complicated than the scheduling problems in single factory. In general, it mainly involves two issues: (i) allocation of jobs to suitable factories and (ii) determination of the production scheduling in each factory [3,7]. Once a job is allocated to a factory and processed, it is usually unable or uneconomical to transfer this work-in-progress part to another factory for the remaining operations. Since production scheduling depends on the job allocation results, the total operating

*Corresponding author. Tel.: +2859 7059; fax: +2858 6535.

E-mail address: ftschan@hkucc.hku.hk (F.T.S. Chan).

cost, makespan, order fulfillment, etc. will be different. The complexity of the problem greatly increases.

At production level, machine maintenance is inevitable. It directly influences the production rate, product quality, machine availability, utilization ratio, etc. If the production schedule obtained from DS does not consider maintenance, the planning determined will be seriously interrupted because of the mismatch among various processes. Consequently, the system reliability will be damaged and the purpose of DS will not be achieved.

The objective of this paper is to propose a new idea named genetic algorithms with dominant genes (GADG) to deal with DS problems subject to machine maintenance constraint. The function of the dominant genes (DGs) is to identify and record the best genes and the corresponding structure in the chromosome. A new encoding of chromosome is also specially designed to deal with the machine maintenance constraint. This paper is divided into the following sections. Section 2 gives a literature review. Section 3 presents the DS problem subject to preventive maintenance. Section 4 presents the proposed DGs, and its crossover mechanism. Section 5 analyzes and discusses the performance of the DGs. Section 6 shows the significance of considering maintenance in DS. Lastly, the paper is concluded in Section 7.

2. Literature review

The main purpose of DS is to maximize the system reliability and the resources utilization through collaboration among different supply chain activities. Distributed systems can be considered as a set of processes which have to be executed in different nodes (locations), and subject to various constraints such as time, capacity, tooling, etc. [8–11]. Each entity has to share the available resources and collaborate with each other in order to achieve the objective. The task scheduling problem consists of defining a schedule that can meet all timing and logical constraints of the tasks being scheduled, and in general, it has been classified as NP-complete [12].

DiNatale and Stankovic [13] applied simulated annealing algorithm to distributed static systems, in which tasks are periodic and have arbitrary deadlines, precedence, and exclusion constraints. They present a general framework consisting of an abstract architecture model and a general programming model. Recently, Jia et al. [3,14] have proposed a modified genetic algorithm (GA) to solve DS problems. They proposed an encoding of chromosome, crossover mechanism, and two mutation mechanisms. Their modified GA has been compared with other classical scheduling approaches and obtained satisfactory results. However, their chromosomes are designed for fixed production routing. For alternative production routing, Chan et al. [15] proposed a new encoding mechanism. They also proposed a DGs approach, which demonstrated its ability to enhance the optimization reliability. There are many other heuristic approaches that can be found in

Barroso et al. [7], Santos et al. [16], Tindell et al. [17], etc. However, in the knowledge of the authors up to this moment, there is a lack of paper which takes account of machine maintenance in DS.

Maintenance policy influences the machine availability and the machine utilization ratio. The purpose of maintenance management is to reduce the effect of breakdown and maximize the facility availability at minimum cost [18–20]. Machine age is an important measurement in maintenance because it affects the inspection time, repairing time, production rate, product quality, failure rate, etc. After each time of maintenance, the machine age has to be adjusted. It will then again induce a new set of inspection time, repairing time, production rate, and product quality [21,22]. Kenne and Boukas [23] proposed a two-level hierarchical control model to deal with the production and preventive maintenance planning control problem for a multi-machine FMS. Chan et al. [24] also proposed a total productive maintenance (TPM) methodology for an electronic manufacturing company, aiming to increase the availability of the existing equipment.

Applying pure mathematical optimization approach to determine an optimal solution may not be efficient. In many cases, scheduling problems are classified as NP-hard. Therefore, applying heuristic methodology to obtain a near optimal solution in a relatively shorter period is more appreciated and practical. Among different heuristic approaches, GAs are recognized as an appropriate and efficient approach. Many references can be found. For example, Cheung et al. [25] gave a detailed tutorial survey on papers using GAs to solve classical Job-Shop scheduling problems (JSP) in their Part I survey. In Part II, they reviewed papers using hybrid GA to tackle JSP [26]. Jain and ElMaraghy [27] proposed a GA to solve single process plan scheduling (SPPS) problems. Cavalieri and Gaiardelli [28] applied a hybrid GA, which combines GA with dispatching rule (Earliest Due Date), to solve multi-objective scheduling problems. Sakawa [29] combined GA with fuzzy logic to model the uncertainties of production lead time and order due date in scheduling problems. More references can be found, for example, Mori and Tseng [30], Jawahar et al. [31], Ghedjati [32].

To strategically strengthen the genetic search in different phases of evolution, many researchers proposed different kinds of Adaptive GAs for their particular problems. During the genetic evolution, the genetic parameters, such as population size, crossover rate, and mutation rate will change strategically [28,33,34]. Michalewicz [33] proposed a non-uniform mutation which allows the operator to search through the solution space uniformly in the beginning stages to prevent prematurity of the solution pool, and then locally in the later stages for fine local tuning. González and Fernández [34] adopted a dynamic population size approach to replace the old chromosomes with new ones to maintain the diversity of solution pool.

3. Problem description

In DS problem, when the network receives a number of jobs (N), it has to determine how to allocate them to the suitable factory $F(= 1, 2, \dots, l)$ and generate the corresponding production schedule(s). Each factory has H_l number of machines and can produce all product types with different efficiency. Each job has up to N_i number of operations, and each operation can be performed on more than one suitable machine (but not all) with different processing time.

Assuming that each machine is subject to a hypothetical maintenance scheme and has a maximum machine age M , as shown in Fig. 1, the machine age equals the cumulated processing time of operations. If the machine age reaches M , maintenance has to be carried out right after the completion of the current operation. After each maintenance, the machine age will be reset to 0. The problem is expressed in the following notations:

- χ_{il} binary integer, defined as 1 if job i is allocated to factory l , otherwise 0.
- δ_{ijkhl} binary integer, defined as 1 if operation j of job i occupied time slot k on machine h in factory l , otherwise 0.
- γ_{ijhl} binary integer, defined as 1 if machine h in factory l will be maintained after operation j of job i , otherwise 0.
- F number of factory.
- N number of jobs.
- N_i number of operations of job i .
- H_l number of machines in factory l .
- D_{li} travel time between factory l and job i .
- K time horizon under consideration.
- T_{ijhl} processing time of operation j of job i on machine h in factory l .
- A_{hl} age of machine h in factory l .
- M maximum machine age.
- S_{ij} starting time of operation j of job i .
- E_{ij} ending time of operation j of job i .
- C_i completion time of job i .

In the problem, it is assumed that F , N , N_i , H_l , and T_{ijhl} are given. The decision variables are χ_{il} , δ_{ijkhl} , and γ_{ijhl} . With the solution χ_{il} , δ_{ijkhl} , and γ_{ijhl} obtained, the value of

S_{ij} , E_{ij} , and C_i can be calculated. The objective is to minimize the makespan of jobs.

Objective $Z : \min(\max\{C_i\})$. (1)

Completion time (C_i) of job i equals the summation of the completion time of the last operation (N_i) of job i and the travel time between factory l and job i as defined as

$$C_i = E_{iN_i} + \sum_l D_{li}\chi_{il}. \quad (2)$$

The problem is subject to the following constraints:

Precedence constraints:

$$S_{ij} \geq E_{i(j-1)} \quad (i = 1, 2, \dots, N; j = 1, 2, \dots, N_i). \quad (3)$$

Processing time constraints:

$$E_{ij} - S_{ij} = \sum_{hl} \chi_{il} T_{ijhl} \quad (i = 1, 2, \dots, N; j = 1, 2, \dots, N_i). \quad (4)$$

Operation constraints:

$$\sum_{khl} \delta_{ijkhl} \geq 1 \quad (i = 1, 2, \dots, N; j = 1, 2, \dots, N_i). \quad (5)$$

Processing operation constraints:

$$\sum_{hl} \delta_{ijkhl} \leq 1 \quad (i = 1, 2, \dots, N; j = 1, 2, \dots, N_i; k = 1, 2, \dots, K). \quad (6)$$

Machine capacity constraints:

$$\sum_{ijl} \delta_{ijkhl} \leq 1 \quad (k = 1, 2, \dots, K; h = 1, 2, \dots, H_l). \quad (7)$$

Factory constraints:

$$\sum_l \chi_{il} = 1 \quad (i = 1, 2, \dots, N). \quad (8)$$

In the above constraints, constraint (3) defines that each operation can only start upon the completion of its preceding operation. Constraint (4) defines that once an operation starts, it will be finished without interruption. Constraint (5) forces each operation to be carried out on one machine throughout the horizon. Constraint (6) forces each operation to be only carried out on one machine at each time unit, and constraint (7) forces each machine to carry out only one operation at each time unit. Constraint

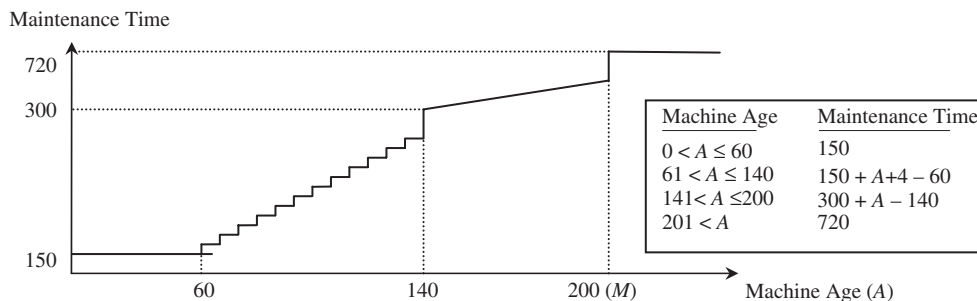


Fig. 1. Maintenance time related to machine age.

Table 1
Model parameters of example 1

Job 1 oper.	m/c	Process time	Job 2 oper.	m/c	Process time	Job 3 oper.	m/c	Process time	Job 4 oper.	m/c	Process time	Job 5 oper.	m/c	Process time
1	1	7	1	1	8	1	1	10	1	2	9	1	1	10
1	3	4	1	2	12	1	2	15	1	3	5	1	3	15
2	2	3	2	3	4	1	3	8	2	1	6	2	2	7
3	1	3	3	1	7	2	2	2	2	3	2	2	3	14
3	3	6	3	2	14	2	3	6	3	2	7	3	1	5
4	1	2	4	1	8	3	1	2	3	3	12	3	2	8
4	2	4	4	3	4	3	3	4	4	1	9	4	1	4
						4	1	6	4	2	6	4	2	6
						4	2	3	4	3	3	4	3	8

(8) forces all the operations of a job to be finished in the same factory.

Table 1 shows a classical single factory scheduling problem with alternative production routing obtained from Lee and DiCesare [35]. They applied Petri Nets combining with some heuristic search to obtain 439 makespan. Recently, Kumar et al. [36] have applied Ant Colony approach to obtain 420 makespan. For benchmarking the proposed algorithm, we will first apply simple GA (SGA) to obtain a solution in the next section.

4. SGA

4.1. Encoding of chromosome

The encoding of chromosome is modified from Jia et al. [14]. Each chromosome represents a possible solution of the allocation of jobs into factory, and the production scheduling of each factory, as shown in Fig. 2a. Each chromosome consists of $\sum_i N_i$ number of genes. In our encoding, each gene composes of four parameters, representing factory, machine, job, and operation (FMJO), while Jia’s encoding composes of FMJ only. Fig. 2a shows a sample solution of the allocation of 3 jobs (each with 3 operations) into 2 factories (each factory has 3 machines), and the production scheduling.

In Fig. 2a, the first gene (1233) represents that O3 of J3 is allocated on M2 in F1. The scheduling priority of jobs on machines is arranged in order, from the highest priority on the left to the lowest one on the right. Therefore, O3 of J3 (1233) will be scheduled before O2 of J1 (1212) on M2 in F1. The advantage of modification of this encoding allows us to model in alternative routing problems. Assuming O3 of J3 can also be performed on M3, it can be represented as (1333) as shown in Fig. 2b.

4.2. Crossover operator

During crossover, a pair of chromosomes will be randomly selected according to the Roulette Wheel selection approach, and a number of genes will be randomly selected according to a predefined crossover

1233-1212-1111-2221-1113-2123-2322-1131-1332
(a)

1333-1212-1111-2221-1113-2123-2322-1131-1332
(b)

Fig. 2. (a) A sample encoding of chromosome. (b) A sample encoding of alternative routing.

	P1:	1233	-	1212	-	1111	-	2221	-	1113	-	2123	-	2322	-	1131	-	1332
	P2:	2223	-	2111	-	2213	-	2321	-	2122	-	2312	-	1131	-	1332	-	1233
Step																		
1	Of1:	2223	-	0000	-	0000	-	2321	-	2122	-	0000	-	0000	-	1131	-	0000
2	Of1:	2223	-	1233	-	1212	-	2321	-	2122	-	1111	-	1113	-	1131	-	1332
Step																		
1	Of2:	1233	-	0000	-	0000	-	0000	-	0000	-	0000	-	0000	-	1131	-	1332
2	Of2:	1233	-	2223	-	2111	-	2213	-	2321	-	2122	-	2312	-	1131	-	1332

Fig. 3. A sample crossover of simple GA.

P1: 1233 - 1212 - 1111 - 2221 - 1113 - 2123 - 2322 - 1131 - 1332
Of1: 1212 - 1233 - 1111 - 2221 - 1113 - 2123 - 2322 - 1131 - 1332
(a)

P1: 1233 - 1212 - 1111 - 2221 - 1113 - 2123 - 2322 - 1131 - 1332
Of1: 1233 - 2122 - 1111 - 2221 - 1113 - 2123 - 2322 - 1131 - 1332
(b)

Fig. 4. (a) A sample procedure of Mutation 1. (b) A sample procedure of Mutation 2.

rate. Fig. 3 shows a sample crossover, in which the first gene is selected to cross over. Since crossover may generate invalid chromosome, such as redundant operations and inconsistent factory, to avoid this, the selected genes and its related genes (in the same job number) will be inherited to the offspring. For example, in Step 1, Of1 is inherited the selected and related genes from P2, and Of2 from P1. Then the remaining genes will be inherited to the offspring from left to right in Step 2, such that Of1 is inherited the remaining from P1, and Of2 from P2.

4.3. Mutation operator

There are two types of mutation. In Mutation 1, a pair of genes will be randomly selected and swapped, as shown in Fig. 4a. The purpose of this mutation is to reschedule the

scheduling priority of job’s operations. For example, after the swap, the production priority on *F1*’s *M2* is that *O3* of *J3* (1233) is rescheduled to be produced after *O2* of *J1* (1212).

In Mutation 2, some genes will be randomly selected and mutated, as shown in Fig. 4b. The number of mutated genes is governed by the predefined mutation rate(s). The selected gene will randomly change in the *F* or *M* parameter. The purpose of this mutation is to increase the diversity of the chromosomes.

4.4. Elitist strategy

To prevent the loss of the best chromosome during evolutions, the best chromosome will be identified and recorded. If the best chromosome is lost or becomes weaker after evolution, it will be inserted back into the mating pool for the next evolution.

4.5. Prevention of premature and local search

To prevent the prematurity of the solution pool and the algorithm from searching around a local optimal, the similarity of the chromosomes will be checked in each evolution. If the similarity evaluated is larger than a threshold, a certain number of chromosomes will be selected and replaced by new ones. The similarity is evaluated by

$$\text{Similarity} = \text{Identical}/\text{Comparison}, \tag{9}$$

	C1	C2	C3	C4
R1:	1111	– 1212	– 1121	– 1222
R2:	1111	– 1212	– 2121	– 2222
R3:	1111	– 1212	– 1121	– 1222
R4:	2111	– 2212	– 1222	– 1121

Fig. 5. A sample of similarity checking.

where Identical is the number of identical pair and Comparison the total number of comparisons.

Fig. 5 shows a sample of similarity checking of 4 chromosomes. Each column will be individually evaluated. In column C1, the gene in R1 will be compared with those in R2, R3, and R4 with a total of 3 comparisons, among which, 2 comparisons show to be identical (R1 = R2 and R1 = R3). Similarly, the gene in R2 will be compared with those in R3 and R4, then, R3 with R4. As a result, Identical equals 3 and Comparison equals 6. The similarity is evaluated to be 0.5, which is equal to the probability of selecting 2 identical genes to undergo crossover. When any one of the columns’ similarity exceeds a certain value, new chromosomes will be inserted.

4.6. Optimization results

The objective of this section is to testify the performance of SGA by comparing it with other existing approaches Petri Nets [35] and Ant Colony [36] in Lee and DiCesare’s Model [35]. As mentioned in the Section 1, many literatures reviewed that different crossover and mutation rates in different problems would have different performance. Therefore, we separately apply single point (5%), 25%, and 50% rates aiming to measure their influences. Each crossover and mutation rate has been individually run for 50 times to measure the deviation of the solutions obtained. The solution pool consists of 100 chromosomes, and the number of evolution is 1000, which is long enough to obtain a steady stage as shown in Fig. 6.

Table 2
Results obtained from simple GA for Model 1

Rate (%)	Avg.	Std. dev.	MIN
5	413	20.2	360
25	394	16.733	380
50	394	11.402	380

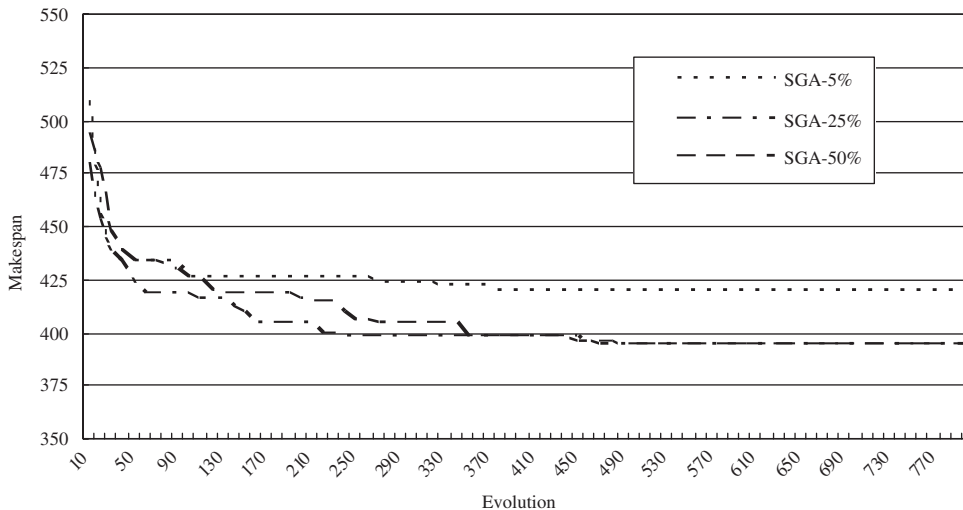


Fig. 6. Results obtained in each evolution.

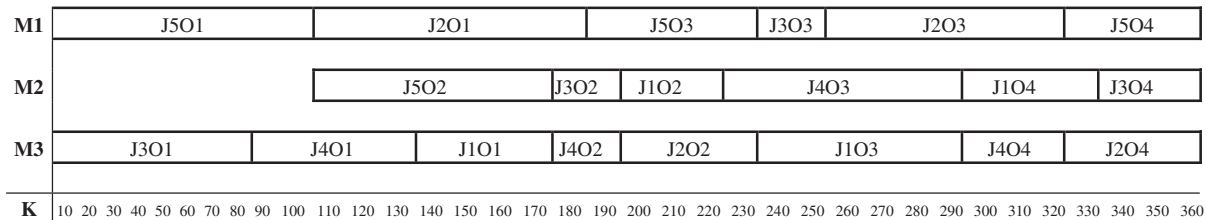


Fig. 7. Production scheduling obtained for Lee and DiCesare’s Model by simple GA.

Table 3
Parameter and results of models

Model size	<i>F</i>	<i>M</i>	<i>J</i>	Avg.	Std. dev.	MIN	Evolution
Model 1	1	3	5	413	20.2	360	800
Model 2	2	6	10	524	40.6	440	2000
Model 3	5	15	25	712	49.9	600	4000
Model 4	10	30	50	907	57.3	650	6000

From literature, Lee and DiCesare [35] applied Petri Nets and Kumar et al. [36] applied Ant Colony to obtain 439 unit time and 420 unit time, respectively. In this paper, SGA obtains the minimum one, which is 360 unit time in 5% rate, as shown in Table 2 and the production scheduling is shown in Fig. 7. Also, the average makespan obtained in 5%, 25%, and 50% is all better than the existing ones. The results also show that SGA in this case performance is better in 25% and 50% rates. This comparison demonstrates that SGA can obtain satisfactory results.

However, the deviation of the solutions obtained is large. We further increase the complexity of the problem to testify the quality of the solutions by increasing the number of Factory, Machine, and Job as shown in Table 3. All the models are run with single point crossover and mutation rates, and the number of evolution is long enough to reach a steady stage. The results show that the deviation of the solutions increases as the complexity increases from Models 1 to 4. In this connection, we will introduce the idea of DG to overcome it.

5. Proposed DGs in GA

5.1. Encoding of chromosome

The encoding of chromosome is further modified. Each gene composes of 5 parameters, representing factory, machine, job, operation, and domination (FMJOD), as shown in Fig. 8. If the gene is classified as DG, *D* parameter will be denoted by 1, otherwise 0. For example, the first gene (12331) is a dominant gene.

5.2. Functions of DGs

In traditional GA approach, a number of genes will be randomly selected during crossover, governed by a predefined crossover rate(s). This rate(s) may be uniform in SGA or non-uniform in adaptive GA. However, in either case, it is usually difficult to ensure that

12331–12121–11110–22210–11130–21230–23220–11311–13320

Fig. 8. A sample encoding of chromosome.

the important part of the chromosome structure can be selected and inherited to its offspring. Sometimes, those selected genes may not be critical even to its original chromosome structure. Indeed, there is a lack of mechanism to measure and identify which genes are important.

The idea of DGs is to identify and record the best genes in each chromosome, and the corresponding structure. Those genes are classified as DGs, when any changes in the genes can increase the fitness value for that particular chromosome. For example, Fig. 8 shows that the first, second, and eighth genes are DGs. These genes are structured in this order and their own value can increase the fitness value for that particular chromosome. During crossover operation, the whole set of DGs will be selected. Note that this set of DGs may not strengthen the other chromosomes or be similar to the optimal structure. The purpose of this approach is to prevent the loss of any identified potential critical structure and give more chance for them to grow stronger.

5.3. Crossover and mutation operators

In the initial stage of evolution, some genes are randomly assigned as DGs in the initial pool. Each chromosome may contain more than one DG. During evolutions, only those DGs undergo crossover in each pair of parents to generate a pair of offspring. Each offspring reserves most of the genes from one of the parents and inherits only the DGs from another parent. If these inherited DGs increase the strength of the chromosome, they will be denoted as DGs in the offspring, otherwise they will be denoted as normal genes. In the DG approach, crossover rate depends on the number of DGs in each chromosome. It can be divided into 2 cases. If there are no DGs conflicting at the same location of the two parents, and if there are no identical jobs dominating in both parents, it will be classified as parent chromosomes in Case A as shown in Fig. 9, otherwise Case B as shown in Fig. 10.

Crossover in Case A is carried out in three steps, as in Fig. 9.

Step 1: Copy all DGs from P1 and P2 to Of1, and empty those identical to the copied genes in P1.

Case A		P1:	12331	-	12120	-	11110	-	22210	-	11130	-	21230	-	23220	-	11311	-	13320
		P2:	22230	-	21110	-	22130	-	23210	-	21221	-	23120	-	11310	-	13320	-	12330
Step		Of1:	12331	-	00000	-	00000	-	00000	-	21221	-	00000	-	00000	-	11311	-	00000
1		P1:	00000	-	12120	-	11110	-	22210	-	11130	-	21230	-	00000	-	00000	-	13320
2		Of1:	12331	-	12120	-	11110	-	22210	-	21221	-	21230	-	11130	-	11311	-	13320
3				-		-		-		-		-		-		-		-	
		Of2:	12331	-	00000	-	00000	-	00000	-	21221	-	00000	-	00000	-	11311	-	00000
1		P2:	22230	-	21110	-	22130	-	23210	-	00000	-	23120	-	00000	-	13320	-	00000
2		Of2:	12330	-	22230	-	21110	-	22130	-	21221	-	23210	-	23120	-	11310	-	13320
3				-		-		-		-		-		-		-		-	

Fig. 9. Dominant gene crossover of Case A.

Case B		(Location)	P1:	12331	-	12120	-	11110	-	22210	-	11130	-	21230	-	23220	-	11311	-	13320
			P2:	22231	-	21110	-	22130	-	23210	-	21220	-	23120	-	11310	-	13320	-	12330
		(Job)	P1:	12331	-	12120	-	11110	-	22210	-	11130	-	21230	-	23220	-	11311	-	13320
			P2:	22230	-	21110	-	22130	-	23210	-	21220	-	23120	-	11311	-	13320	-	12330

Fig. 10. Dominant genes conflicting in the location and same job.

Step 2: Replace the genes in P1 with those in P2 with identical job number to its DGs.

Step 3: Copy the non-emptied genes from P1 to the non-emptied genes in Of1.

Similar steps will be carried out to obtain Of2. One of the advantages of this crossover mechanism is that the best genes will undergo crossover. In addition, the changed genes will be testified whether they can increase the fitness value. If they make contribution to the chromosome, they will be recorded, and inherited to its next generation. Assuming that Of1 is better than P1, then the inherited DGs from P2 will remain its domination. However, if it is weaker than its parents, assuming Of2 is weaker than P2, the inherited DGs from P1 will become normal genes.

In Case B, since P1 and P2 have DGs conflicting in the same location(s) or job(s), as shown in Fig. 10, a selection is required to testify which sets of DGs contribute more to the offspring. The checking will be done by setting one set of DGs as normal genes, and then the other to satisfy the criteria of Case A in Of1A and Of1B respectively setting one set of DGs as normal genes, and then the other to satisfy the criteria of Case A. The stronger offspring generated will be Of1.

The mutation mechanism is similar to the two types of mutation operators discussed in Section 4.3 except that if the offspring is stronger than its parent, the mutated genes will be denoted as DGs.

5.4. Prevention of prematurity and local search

Similarity checking as discussed in Section 4.5 is applied again to prevent the prematurity of solution pool, and the algorithm from searching around a local optimal. How-

ever, in order to take the balance between the local search and global search, the algorithm will strategically increase the diversity of the solution pool. Throughout the evolution, the mutation rate is set as two genes for Mutation 1, and one gene for Mutation 2. This low mutation rate can increase the strength of local search. However, to prevent the algorithm from being trapped in a local optimal, each evolution will have an improvement checking. If there is no improvement found after certain number of evolutions, Mutation 2 will be run with 5% rate to slightly bring the searching out of the existing area. If no improvement is found for another certain number of evolutions, Mutation 2 will be run and the rate will be increased to 10%, and so on. The setting of rates is 5% → 10% → 15% → 25%, and is then reset to 5% after a cycle.

5.5. Optimization results

The objective of this section is to testify the optimization reliability of the proposed GADG by comparing it with SGA as discussed in Section 4. Again, each model will be individually run for 50 times to measure the deviation of the results obtained with the same number of evolution as adopted in SGA. Table 4 shows that the average results obtained improve from Models 1 to 4. In addition, the deviation of the solutions is smaller. The proposed algorithm can also obtain a shorter makespan but SGA cannot. Fig. 11 shows the average makespan of GADG and SGA obtained in Model 4. In the figure, GADG converge slower than different rates of SGA in the initial stage, but steadily improve to obtain a better solution. This comparison indicates that DG improves the quality of the solution obtained and reduces the deviation of solution generated.

Table 4
Results obtained from DG approach

	Avg.	Std. dev.	MIN	Evo.	Improvement		
					Avg. (%)	Std. dev. (%)	MIN (%)
Model 1	374	10.5	350	800	9.4	48.0	2.8
Model 2	440	18.7	400	2000	16.0	53.9	9.1
Model 3	565	20.9	530	4000	20.6	58.1	11.7
Model 4	691	26.8	620	6000	23.8	53.2	4.6

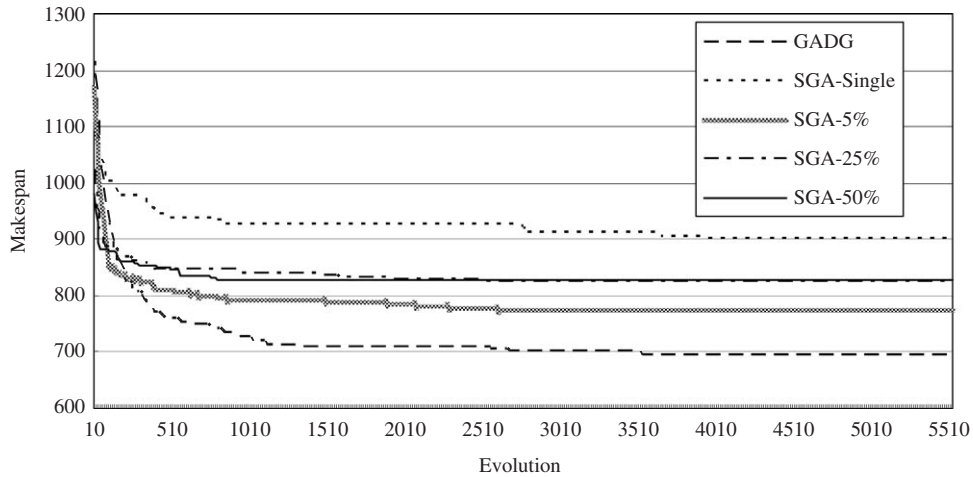


Fig. 11. Performance of DG and SGA in Model 4.

6. DS subject to maintenance

123311–121201–111100–222110–111300–212300–232200–113101–133200

6.1. Encoding of chromosome

Fig. 12. A sample encoding of chromosome.

The optimization reliability of GADG has been demonstrated in Section 5. GADG will be applied to solve the DS problem, which is subject to maintenance as discussed in Section 3. The encoding of chromosome will be further modified to FMJOSD, in which the *S* parameter represents the machine will stop for maintenance (the *S* parameter will be denoted as 1 if the machine will be maintained after the current operation, otherwise 0). For example, the first gene indicates that *M2* in *F1* will be maintained after *O3* of *J3* and it is denoted as DG in Fig. 12.

6.2. Optimization results

The objective of these simulation runs is to demonstrate the significance of considering maintenance during DS. Two sets of simulation runs will be carried out. In Set 1, maintenance will not be considered during DS (as discussed in Section 5). In Set 2, maintenance will be considered during DS. Table 5 shows a sample of the problem parameters with processing time of operation on different machines, and the traveling distance between factories and customers. It has 2 factories and 10 jobs. Assuming the processing time of the operations in *F1* and *F2* is the same,

the selection operator, crossover operator, mutation operator, and similarity checking are similar as discussed in Section 5. The number of evolution adopted is 5000 with the solution pool size of 100. Each set will be run for 50 times individually again to measure the deviation of the results.

Table 6 summarizes the results of the simulation runs for Sets 1 and 2. Assuming that maintenance is not required on machines, the average makespan obtained is 510 unit time. However, if each machine is forced to stop for maintenance after its concurrent operation when the age is older than 200 unit time, the average makespan obtained becomes 1780 unit time. However, when maintenance is considered during DS as in Set 2, the average makespan can be shortened to 1280 unit time, which has 28% improvement.

For example, Fig. 13 shows a sample scheduling obtained from Set 1 with the makespan of 490 unit time, and the corresponding scheduling after maintenance is 1920 unit time as shown in Fig. 14. Fig. 15 shows a sample scheduling obtained from Set 2 with only 1220 unit time. This comparison demonstrates that adequate scheduling maintenance during DS can shorten the makespan by improving the machine utilization ratio, such as in Fig. 14,

Table 5
Problem parameters of a sample distributed scheduling subject to maintenance

Job 1 oper.	m/c	Process time	Job 2 oper.	m/c	Process time	Job 3 oper.	m/c	Process time	Job 4 oper.	m/c	Process time	Job 5 oper.	m/c	Process time
1	1	7	1	1	8	1	1	10	1	2	9	1	1	10
1	3	4	1	2	12	1	2	15	1	3	5	1	3	15
2	2	3	2	3	4	1	3	8	2	1	6	2	2	7
3	1	3	3	1	7	2	2	2	2	3	2	2	3	14
3	3	6	3	2	14	2	3	6	3	2	7	3	1	5
4	1	2	4	1	8	3	1	2	3	3	12	3	2	8
4	2	4	4	3	4	3	3	4	4	1	9	4	1	4
4	4	1	6	4	2	6	4	2	6					
4	4	2	3	4	3	3	4	3	8					
Job 6 oper.	m/c	Process time	Job 7 oper.	m/c	Process time	Job 8 oper.	m/c	Process time	Job 9 oper.	m/c	Process time	Job 10 oper.	m/c	Process time
1	1	7	1	1	8	1	1	10	1	2	9	1	1	10
1	3	4	1	2	12	1	2	15	1	3	5	1	3	15
2	2	3	2	3	4	1	3	8	2	1	6	2	2	7
3	1	3	3	1	7	2	2	2	2	3	2	2	3	14
3	3	6	3	2	14	2	3	6	3	2	7	3	1	5
4	1	2	4	1	8	3	1	2	3	3	12	3	2	8
4	2	4	4	3	4	3	3	4	4	1	9	4	1	4
4	4	1	6	4	2	6	4	2	6					
4	4	2	3	4	3	3	4	3	8					
Travel time	Job													
Factory	1	2	3	4	5	6	7	8	9	10				
1	10	10	10	10	10	10	10	10	10	10	10	10	10	10
2	20	20	20	20	20	20	20	20	20	20	20	20	20	20

Table 6
Optimization results for Sets 1 and 2

	Avg.	Std. dev.	MIN
Set 1			
No maintenance	510	12.87	490
Maintenance	1780	21.83	1920
Set 2			
Maintenance	1280	13.14	1220

the total maintenance time is 6480 (9 × 720) unit time, while Fig. 15 is only 3060 (170 + 320 + 200 + 160 + 200 + 230 + 230 + 340 + 160 + 150 + 150 + 230 + 150 + 160 + 210) unit time. The machine utilization ratio has 53% improvement.

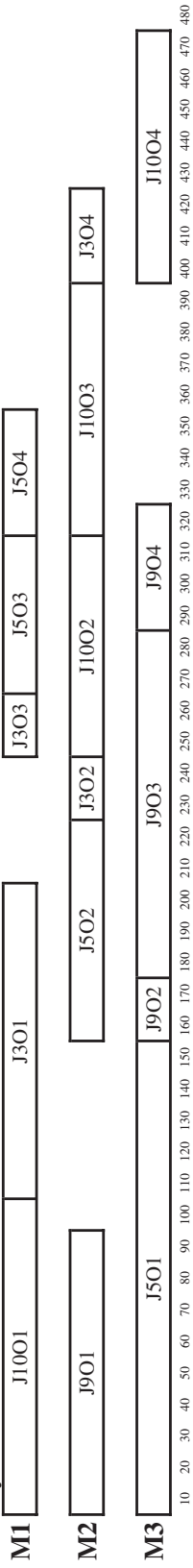
7. Conclusion

In conclusion, this paper proposed a Genetic Algorithm with Dominant Genes (GADG) approach to solve distributed FMS scheduling problem subject to machine maintenance. The idea of Dominant Genes (DGs) and various genetic operators including selection, crossover, and muta-

tion have been presented. A Simple Genetic Algorithm (SGA) approach has been compared with Petri Nets [35] and Ant Colony [36] in Lee and DiCesare’s Model to testify its performance. The optimization results indicate that SGA performs better. However, when the problem size increases, the deviation of the solutions obtained becomes larger. To overcome the problem, DGs is applied. In traditional crossover mechanism, a number of genes will be randomly selected, governed by a predefined crossover rate(s). However, it is usually difficult to ensure that the important part of the chromosome structure can be selected and inherited to its offspring. In the new approach, the proposed DGs identify and record the best genes in each chromosome, and the corresponding structure. The results obtained by GADG have been compared with the ones obtained by SGA. The comparison indicates that GADG improves the quality of the solution, and reduces the deviation of the results obtained. Lastly, GADG has been modified to consider machine maintenance. Two sets of simulation runs have been carried out. Set 1 does not consider maintenance, while Set 2 does. The comparison of the results demonstrates that considering maintenance during DS can shorten the makespan by improving the machine utilization. The average makespan obtained by Set 2 is shorter. In addition, the machine utilization is improved as well.

No Maintenance

Factory 1



Factory 2

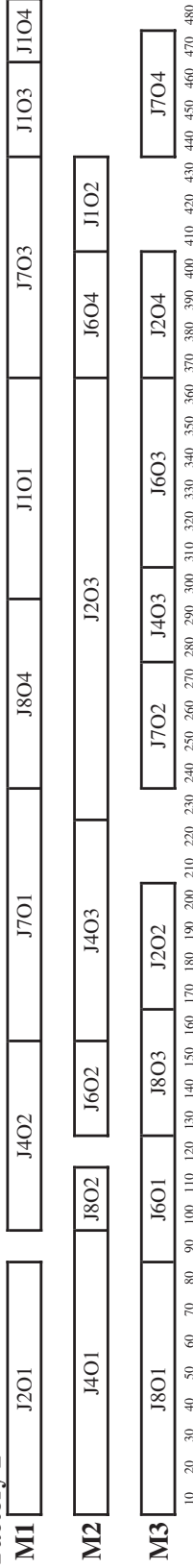
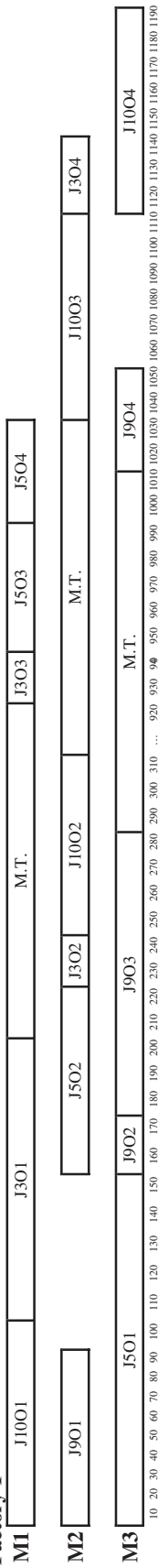


Fig. 13. Result of distributed scheduling without the consideration of maintenance (Set 1).

No Maintenance - Must Stop at 200 unit

time

Factory 1



Factory 2

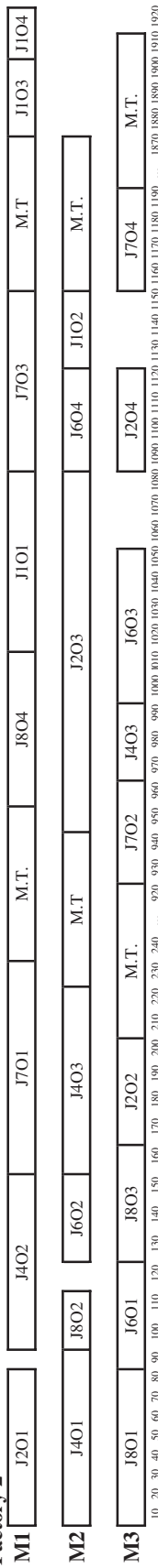


Fig. 14. Distributed scheduling of Fig. 13 after the consideration of the maintenance factor.

- [24] Chan FTS, Lau HCW, Ip RWL, Chan HK, Kong S. Implementation of total productive maintenance: a case study. *Int J Prod Econ* 2005; 95(1):71–94.
- [25] Cheung R, Gen M, Tsujimura Y. A tutorial survey of job-shop scheduling problems using genetic algorithms—I. *Comput Ind Eng* 1996;30(4):983–97.
- [26] Cheng R, Gen M, Tsujimura Y. A tutorial survey of job-shop scheduling problems using genetic algorithms—II. *Comput Ind Eng* 1999;37(1):51–5.
- [27] Jain AK, ElMaraghy HA. Single process plan scheduling with genetic algorithm. *Prod Plan Control* 1997;8(4):363–76.
- [28] Cavalieri S, Gaiardelli P. Hybrid genetic algorithms for a multiple-objective scheduling problem. *J Intell Manuf* 1998;9:361–7.
- [29] Sakawa M. Genetic algorithms and fuzzy multi-objective optimization. Dordrecht: Kluwer Academic Publisher; 2002. p. 188–222.
- [30] Mori M, Tseng CC. Genetic algorithms for multi-mode resource constrained project scheduling problem. *Eur J Oper Res* 1997;100(1): 134–41.
- [31] Jawahar N, Aravindan P, Ponnambalam SG. A genetic algorithm for scheduling flexible manufacturing systems. *Int J Adv Manuf Technol* 1998;14:588–607.
- [32] Ghedjati F. Genetic algorithms for the job-shop scheduling problem with unrelated parallel constraints: heuristics mixing method machines and precedence. *Comput Ind Eng* 1999;73: 39–42.
- [33] Michalewicz Z. Genetic algorithms + data structures = evolution programs. Berlin, Heidelberg, NY: Springer; 1996.
- [34] González EL, Fernández MAR. Genetic optimization of a fuzzy distribution model. *Int J Phys Distrib Log Manage* 2000;30(7/8): 681–96.
- [35] Lee DY, DiCesare F. Scheduling flexible manufacturing systems using petri nets and heuristic search. *IEEE Trans Robot Autom* 1994;10(2):123–32.
- [36] Kumar R, Tiwari MK, Shankar R. Scheduling of flexible manufacturing systems: an ant colony optimization approach. *IMEchE* 2003;217(Part B):1443–53.