Production Scheduling for Parallel Machines Using Genetic Algorithms

Chichang Jou[1], Hsin-Chang Huang[2]

[1] Tamkang University, Department of Information Management (cjou@mail.im.tku.edut.tw)
[2] Tamkang University, Department of Information Management (wind@mail.im.tku.edut.tw)

Abstract

Production scheduling is to seek shortest manufacturing time for all tasks, under the condition that the inventory is stable, the human and machine utilization rate is balanced, and the average customer waiting time is minimized. It normally needs to compromise among the factors of inventory stability, utilization rate, and customer satisfaction. Various heuristic and approximation solutions have been proposed for miscellaneous production scheduling problems. Of them, many scheduling algorithms for single machines with flow shop production have been proven effective. Achievements also have been gained in scheduling for single machines that produce same products and are inventory based. However, the production scheduling problem for parallel machines has been proven as NP-hard. Since many real production machines manufacture different products with the production type of flow shop and build to order, we would like to attack the production scheduling problem for this type of parallel machines. Previous heuristic rules normally consider too few factors in production scheduling. Therefore, we focus on the suboptimal approximation algorithms. Of the many probabilistic approximation algorithms, genetic algorithms take advantage of the good genes during the evolutionary process, so they usually perform better in approaching the optimal solution. In this paper, we propose a modified genetic algorithm that adds steps of selection and bounce to the traditional genetic algorithm to speed up the search of the approximation solution. We design new chromosomes that will make job assignments to machines more flexible. The evolutionary mechanisms are also designed to obtain sub-optimal solutions more rapidly. We also include the considerations of machine idleness cost and machine setup cost in the fitness function to reflect real cases more closely. Finally, we implement a working system and test its applicability using real production data from an electronic factory. We compare our results with those of previous studies and demonstrate that our algorithm has the following advantages: 1. The additional steps of selection and bounce do help our algorithm perform better than traditional genetic algorithms. 2. Our solutions have smaller deviations. 3. Our fitness function is more suitable for real manufacturing cases.

1. Introduction

Production scheduling is to seek shortest manufacturing time for all tasks, under the condition that the inventory is stable, the human and machine utilization rate is balanced, and the average customer waiting time is minimized. It normally needs to compromise among the factors of inventory stability, utilization rate, and customer satisfaction. Various heuristic and approximation solutions have been proposed for miscellaneous production scheduling problems [5,7,8,13,20,22]. Of them, many scheduling algorithms for single machines with flow shop production have been proven effective [2]. Achievements also have been gained in scheduling for single machines that produce same products and are inventory based [5,22]. However, the scheduling problem for parallel machines has been proven as NP-hard [10,12], mainly due to the additional choice of which machine a job should be assigned to. Since many real production machines manufacture different products with the production type of flow shop and build to order, we focus our research on the production scheduling for this type of parallel machines. Heuristic rules normally consider too few factors in production scheduling. Thus, we focus on the suboptimal approximation algorithms. Of the many probabilistic approximation algorithms, genetic algorithms take advantage of the good genes during the evolutionary process, so they usually perform better in approaching the optimal solution.

We propose a modified genetic algorithm that adds steps of selection and bounce to the original genetic algorithm to speed up the search of the approximation solution. The selection step will sort the chromosomes based on their fitness function values, and then assign them a probability distribution in favor of the good chromosomes, so as to obtain a better chance of producing good next generations. The bounce step will help us to jump out of the local optimum to prevent premature convergence by checking the number of generations with the same suboptimal solution. Based on Cheng's genetic algorithm [10] for production scheduling of parallel machines, we redesign their allocating genes with distinguishable allocating genes, which are a group of special symbols, '*', supplemented with a subscript number, so that the solution space could be more easily explored. We improve the evolutionary mechanisms accordingly to approach the optimal solution faster. Besides the earliness and tardiness considerations, we include factors of variable machine idleness cost and machine setup cost in the fitness function. The adding is mainly due to our observation that many production managers need to adjust the output schedule during the off seasons to reduce the operating cost, and that most jobs do not start immediately after their predecessor jobs are finished. Finally, we implement a working system and use real production data from an electronic factory to verify its effectiveness. We compare our results with those of previous studies.

2. Related Work

Based on the scheduling methods, there are two approaches to solve the production scheduling problems:

(1) Heuristic rules:

Graves [15] and Cheng [8] have shown that based on the factors of ordering types and machine load, the following rules are used frequently for single machines: SPT (shortest processing time), EDD (earliest due date), and SLK (least slack time). To extend these rules to parallel machines, they need to handle the job assignment issues.

(2) Approximation algorithms:

These algorithms include Simulated Annealing [1,18], Hill Climbing [6], Tabu Search [4,5] and Genetic Algorithms [13,14,20]. They probabilistically approached the optimal solution iteratively to obtain a suboptimal approximate solution for computationally complex problems. One characteristic of these approximation algorithms is that they all need a fitness function to judge how well is the approximation. They would then probabilistically modify the configuration of existing good approximate solutions, and continue the search of the optimal solution until the result is satisfactory. Genetic algorithms usually perform better in approaching the optimal solution, since they take advantage of the good genes during the evolutionary process.

In applying to real cases, Cheng [9] considered the cost of earliness and tardiness in evaluating the effectiveness of a schedule. Webster [23] adapted the genetic algorithms to solve the production scheduling of single machines, but assumed fixed cost of earliness and tardiness. De [11], Lee [19], Ahmed [3], and Adamopolulos [2] then proposed various production scheduling algorithms with variable earliness and tardiness costs. Hall [16] proved that under the environment of single machines, fixed due date, and variable cost of earliness and tardiness, the problem of production scheduling is NP-complete. Cheng [10] and Gao [12] pointed out that the production scheduling problem for parallel machines under certain environments is NP-hard. Sridharan [22] included the cost of machine idleness in the influencing factors of production scheduling. We consider the following factors in the scheduling: earliness, tardiness, machine setup, and machine idleness.

In adapting genetic algorithms to solve production scheduling problem for parallel machines, Cheng [10] used the job numbers and a special symbol, '*', as genes. Suppose there are *m* production machines. The number of the special symbols in their algorithm is *m-1*, and each linear sequence of the genes models a chromosome. The special symbol '*' is for allocating jobs to machines sequentially. Except for the case of mutation, all their other evolutionary mechanisms keep the position of allocating genes. That does not consider all possible combinations of genes. Thus, the initial solution for each run has a strong impact on the result of their algorithm. We replace their allocating gene with distinguishable allocating genes, which are a group of genes, '*', supplemented with a subscript number, so that the solution space would be more easily explored. We then improve the evolutionary mechanisms accordingly.

3. Our Genetic Algorithm and its Modules

We present our genetic algorithm in this section. We will elaborate the idea of distinguishable allocating genes, the evolutionary mechanisms, our fitness functions, our two additional steps of selection and bounce, and how to obtain the initial solution to start the algorithm.

3.1 Design of Genes

Genetic algorithms [14,17,21] were proposed by Holland in 1975. They simulate Darwin's *Nature Selection* evolutionary theory by encoding the factors of a problem by chromosomes, where each gene represents a characteristic of the problem. The combinations of genes are evolved through the evolutionary mechanisms so that the chromosome could approach the optimal solution generation by generation.

There are three evolutionary mechanisms: crossover, mutation, and reproduction. Crossover combines two good chromosomes and generates next-generation chromosomes preserving good characteristics. Mutation reorganizes the combination of genes in a chromosome randomly so that new combination of genes could appear in the next generation. This will help the search to jump out of local optimal solutions. Reproduction is to copy a chromosome to the next generation directly so that good chromosomes in different generations could cooperate in the evolution.

Our algorithm first chooses arbitrary initial solution according to some heuristic rule and transforms it into a chromosome. Details about this part are discussed in Section 3.5. Then chromosomes in each generation are sorted by their fitness function values. Chromosomes are then chosen probabilistically as the parents for generating the next generation through the three evolutionary mechanisms. During the evolutionary process, we will record the best chromosome up to each generation, which is the suboptimal solution so far.

Suppose there are *n* jobs to be assigned to *m* machines. A chromosome is modeled by a sequence of *n+m-1* distinct genes, which consists of *n* job genes and *m-1* distinguishable allocating genes, '*', with subscripts from *1* to *m-1*. For example, suppose we need to allocate 9 jobs to 4 machines. The following chromosome, "1 2 3 $*_2$ 4 5 $*_1$ 6 7 8 $*_3$ 9", would allocate jobs 1, 2, 3 to machine 1, jobs 4, 5 to machine 2, jobs 6, 7, 8 to machine 3, and job 9 to machine 4. The subscript numbers in allocating genes are to distinguish the allocating genes, not to associate them with the machines.

3.2 The Evolutionary Mechanisms

(1) Crossover

Our crossover mechanism is accomplished through the following steps, and Fig. 1 demonstrates an example:

a. Randomly choose two chromosomes with good fitness function values.

b. Randomly produce a sequence of *n+m-1* flags, which are valued as either 0 or 1.

c. In the first chromosome, those genes with flag 0 are kept in its original position, while those genes with flag 1 are reordered according to their order in the second chromosome.

Step 1: Chromosome1:	1	2	3	$*_1$	4	5	$*_2$	6	7	8	$*_3$	9
Flags:	0	1	0	1	0	1	0	0	1	1	1	0
Selected:	1		3		4		$*_2$	6				9
Not Selected:		2		$*_1$		1			7	8	$*_3$	
Step 2: Chromosome2:	1	3	8	4	2	$*_3$	$*_2$	6	$*_1$	7	9	5
Selected:			8		2	$*_3$			$*_1$	7		5
Reordered:		8		2		$*_3$			$*_1$	7	5	
Step 3: New Chromosome:												
Combined:	1	8	3	2	4	$*_3$	$*_2$	6	$*_1$	7	5	9

Fig. 1: Example of the Crossover Mechanism of our Genetic Algorithm

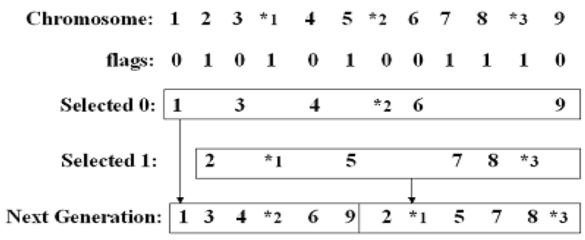
(2) Mutation

Our mutation mechanism is accomplished through the following steps, and Fig. 2 demonstrates an example:

a. Randomly choose one chromosome with good fitness function value.

b. Randomly produce a sequence of *n+m-1* flags, which are valued as either 0 or 1.

c. Those genes with flag 0 would then be shuffled to the front of the sequence with their order reserved. Similarly, those genes with flag 1 would be shuffled to the back of the sequence with their order reserved.



Chromosome: 1 2 3 *1 4 5 *2 6 7 8 *3 9
flags: 0 1 0 1 0 1 0 0 1 1 1 0
Selected 0: 1 3 4 *2 6 9
Selected 1: 2 *1 5 7 8 *3
Next Generation: 1 3 4 *2 6 9 2 *1 5 7 8 *3

Fig. 2: Example of our Mutation Mechanism

(3) Reproduction

This mechanism just copies the chromosome of the parent to its next generation.

3.3. The Fitness Function

As in Section 3.1, suppose we have n jobs to be assigned to m machines. The following are the basic definitions of symbols used in the fitness function:

S_s : starting time of schedule s

F_s : finishing time of schedule s

J : the set of jobs to be scheduled: $J_1, J_2, \ldots, J_i, \ldots, J_n$

M : the set of machines to which jobs will be assigned, $M_1, M_2, \ldots, M_j, \ldots, M_m$

M_s : the number of machines used in schedule s

U_s : machine utilization ratio in schedule s, which is equal to M_s / m

I_s : total machine idle time in schedule s

R_s : machine idleness ratio in schedule s, which is equal to $I_s / (F_s - S_s)$

t_{ij} : job processing time for job J_i in machine M_j

d_i : due date for job J_i

s_{ij} : machine setup time for job J_i using machine M_j

p_j : working hours per day for machine M_j

B_{si} : beginning date for job J_i in schedule s

C_{si} : closing date for job J_i in schedule s. Suppose in s, J_i is assigned to machine M_j. C_{si} is equal to $B_{si} + [\,(s_{ij} + t_{ij}) / p_j\,]$

L_{si} : lag of job J_i, defined as the difference between J_i's finishing date and due date. It is equal to $C_{si} - d_i$.

E_{si} : earliness of job J_i in schedule s, which is equal to $max\ \{0, -L_{si}\}$

T_{si} : tardiness of job J_i in schedule s, which is equal to $max\ \{0, L_{si}\}$

$\alpha_i(E_{si})$: earliness cost function for job J_i in schedule s

$\beta_i(T_{si})$: tardiness cost function for job J_i in schedule s

γ : weight for machine setup cost

λ : weight for machine idleness cost

Π: the set of the pairs of closing date and due date , (C_i,d_i), for all jobs J_i

The goal of the evolutionary procedure is to find a minimal fitness function value. For a schedule s, let C, d, be the vectors of closing date and due date for each job, and U, R the corresponding utilization ratio and idleness ratio. Our fitness function is defined as follows:

$$F(C,d,U,R) = \sum_i [\alpha_i(E_{si}) + \beta_i(T_{si})] + \gamma U_s + \lambda R_s$$

Note that the earliness cost, $\alpha_i(E_{si})$, and tardiness cost, $\beta_i(T_{si})$, could represent the inventory cost for the early finished stocks and the penalty for the lateness, respectively. Since their cost is normally proportional to their value, they must be monotonic polynomial functions. We would like to find C^*, d^*, U^*, and R^* such that

$$F(C^*, d^*, U^*, R^*) = \underset{(C,d)\in\Pi}{Min}\{F(C, d, U, R)\}$$

3.4. The Amended Genetic Algorithm

In designing this amended genetic algorithm, our idea is to converge quickly but not prematurely [20]. The premature convergence is shown as being trapped in local optimal solution. In our amended genetic algorithm, we introduce two additional steps: *selection* and *bounc*e. Fig. 3 shows these new steps highlighted. In the traditional genetic algorithm, by a fair probability, two chromosomes are selected as parents from senior generations. However, not all senior generation populations have good chromosomes. In our selection step, all the current generation chromosomes are sorted according to their fitness function values. They are assigned a probability in the fashion that chromosomes with lower fitness values are given a higher probability of being chosen for the next evolution. Then a random number generated will be mapped to the appropriate chromosome that complies with the above probability distribution. This not only ensures the solution quality but also avoids being trapped into local optimum. The bounce step is added after the evaluation of solutions in each new generation. It checks the number of successive generations with unchanged best approximate solution so far. When this number reaches a preset threshold, like 5, we consider that a sign of premature convergence. The current best solution is then bounced out of the local search space to increase the diversity of the solution space. This brings new variations to the population, like the effect of mutation, and will accelerate the convergence speed.

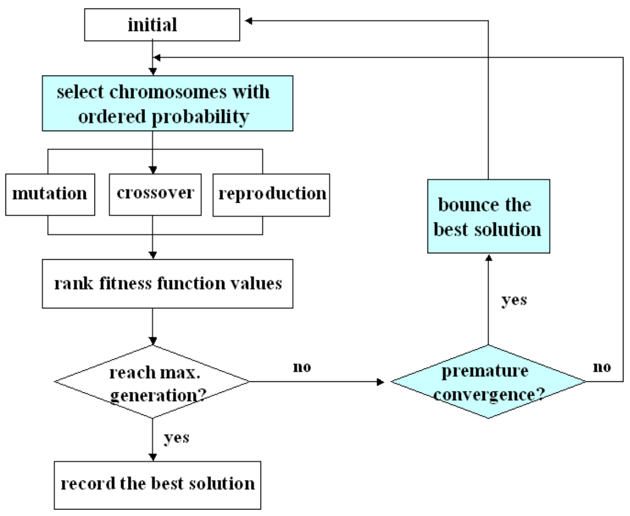
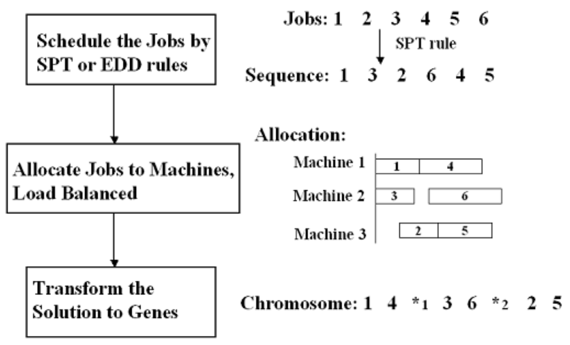


Fig. 3: Our Genetic Algorithm

3.5. Initial Solutions

The initial solution to start our modified genetic algorithm is obtained through the use of heuristic rules, like SPT or EDD. Fig.4 shows an example, where the initial job sequence by applying SPT rule is: 1 3 2 6 4 5. Then according to the machine load balance principle, jobs 1,4 are assigned to machine 1, jobs 3, 6 to machine 2, and jobs 2, 5 to machine 3. The schedule would then be transform to obtain the initial chromosome as: 1 4 $*_1$ 3 6 $*_2$ 2 5.


Schedule the Jobs by SPT or EDD rules
Jobs: 1 2 3 4 5 6
SPT rule
Sequence: 1 3 2 6 4 5
Allocate Jobs to Machines, Load Balanced
Allocation:
Machine 1
1
4
Machine 2
3
6
Machine 3
2
5
Transform the Solution to Genes
Chromosome: 1 4 *1 3 6 *2 2 5

4. System Implementation

We implement a production scheduling system using the modified genetic algorithm in Section 3. In this section, we demonstrate the architecture and major user interface of the system, and then exhibit the performance comparison with other algorithms.

4.1. System Architecture

Fig. 5 exhibits the main architecture of our system, following the one proposed by Gao [12], with modifications to fit the already existing systems of a real production plant.

Fi

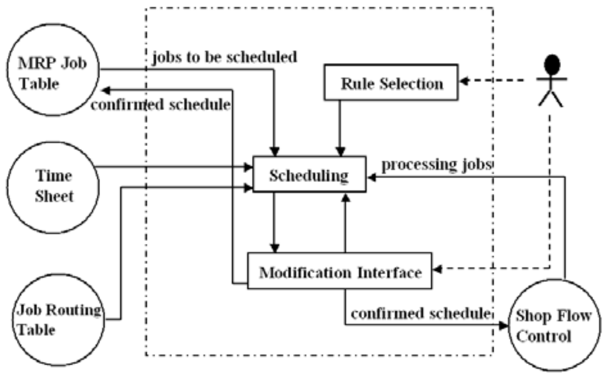


Fig. 5: Architecture of our Production Scheduling System

The following explains the main functions of the system:

(1) Data Communication:

The inputs are the jobs to be scheduled from the MRP system and the information about currently processing jobs from shop flow control. Additional information are the time sheet for the productivity and the job routing table for required processing time of each job in each machine. The output could be verified by the production manager. Confirmed schedules are then sent to the MRP system and shop flow control.

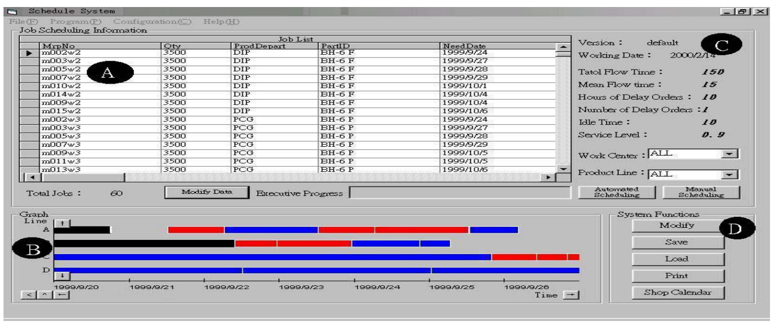
(2) Human Interface:

We provide interface to select the rules for the initial solution for the genetic algorithm. Parameters of the fitness function, as well as the weights of earliness and tardiness, could be adjusted according to previous experiences and various requirements. The schedule generated by the system could be manually adjusted and tested to obtain better performance.

(3) Result Presentation:

In addition to automatic data feed to the MRP system and shop flow control, the results are also printed out as reports. They are presented both graphically and in a tabular form.

The main user interface of the system is shown in Fig. 6:



Job List

MrpNo	Qty	ProdDepart	PartID	NeedDate
m002w2	3500	DIP	BH-6 F	1999/9/24
m003w2	3500	DIP	BH-6 F	1999/9/27
m005w2	3500	DIP	BH-6 F	1999/9/28
m007w2	3500	DIP	BH-6 F	1999/9/29
m010w2	3500	DIP	BH-6 F	1999/10/1
m014w2	3500	DIP	BH-6 F	1999/10/4
m009w2	3500	DIP	BH-6 F	1999/10/4
m015w2	3500	DIP	BH-6 F	1999/10/6
m002w3	3500	PCG	BH-6 P	1999/9/24
m003w3	3500	PCG	BH-6 P	1999/9/27
m005w3	3500	PCG	BH-6 P	1999/9/28
m007w3	3500	PCG	BH-6 P	1999/9/29
m009w3	3500	PCG	BH-6 P	1999/10/5
m011w3	3500	PCG	BH-6 P	1999/10/5
m013w3	3500	PCG	BH-6 P	1999/10/6

Total Jobs : 60

Executive Progress

Version : default
Working Date : 2000/2/14
Tatol Flow Time : 150
Mean Flow time : 15
Hours of Delay Orders : 10
Number of Delay Orders : 1
Idle Time : 10
Service Level : 0. 9
Work Center : ALL
Product Line : ALL

Fig. 6: User Interface of our Production Scheduling System

In Fig. 6, area A is the job table area, containing information about jobs before or after the scheduling. Attributes include job number, beginning date, due date, quantities, etc. Area B is the graphical representation of the result schedule, where colors of the jobs represent their status. A job in black color indicates it is currently processing. A job in blue color means it will be finished on time. A job in red color means it will past due date. Production managers could use the color information to adjust the result. Area C is the performance indicators, including total processing time, average processing time, total tardiness time, number of jobs past due date, total machine idle time, utilization rate, etc. Area D is for related supporting functions, including manual adjustment, saving schedules, loading previously generated but not yet confirmed schedules, and printing reports.

4.2. Experimental Results

We collected the MRP job table, time sheets, and job routing table from an electronic factory that produces motherboards and barebones. We simulate a production case task that needs to flow through workstations of SMT, DIP, and PCG. The workstations have 4, 3, 3 parallel assembly lines separately, and each workstation has 15 jobs to schedule. For the earliness and tardiness cost functions, we designed three functions for the jobs according to their priorities:

(1) For low priority jobs, we define:

$$\alpha_i(E_{si}) = 2 \qquad \& \qquad \beta_i(T_{si}) = 5$$

(2) For general jobs, we define:

$$\alpha_i(E_{si}) = 2E_{si} + 2 \qquad \& \qquad \beta_i(T_{si}) = 5T_{si} + 5$$

(3) For high priority jobs, we define:

$$\alpha_i(E_{si}) = 2E_{si}^2 + 2E_{si} + 2 \quad \& \quad \beta_i(T_{si}) = 5T_{si}^2 + 5T_{si} + 5$$

The weights of machine setup cost and machine idleness cost are set as: γ=40 and λ = 80. Therefore, for each workstation, the fitness function is as follows:

$$F(C, d, U, R) = \sum_{1 \le i \le 15} [\alpha_i(E_{si}) + \beta_i(T_{si})] + 40U_s + 80R_s$$

We set the number of evolutions as 50, and set the number of populations in each generation as 50. Fig. 7 shows our results after 10 test runs with four possible combinations with regard to whether the selection or bounce steps are chosen:

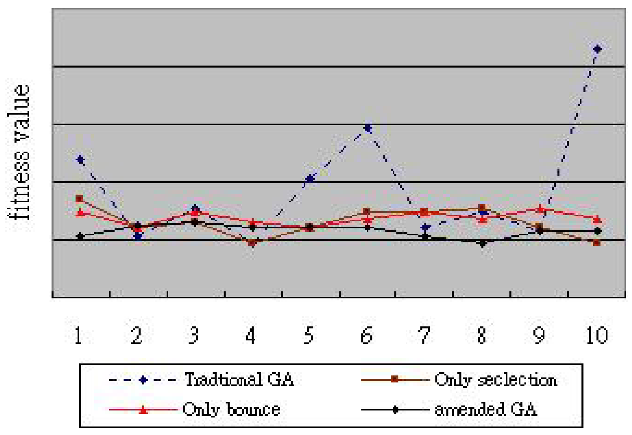


Fig 7. The Distribution of Best Solution in Different Runs

Table 1 shows the average and standard deviation of fitness function values with regard to the two additional steps:

Table 1: Effects of the Two Additional Steps

	normal genetic algorithm	add selection step	add bounce step	add both selection and bounce steps
average fitness function value	1.12	1.03	1.04	1.00
standard deviation of fitness function values	21.18	5.07	2.55	2.11
(standard deviation) / average fitness function value	15.33%	4.02%	2.00%	1.71%

From Table 1, we find that both selection and bounce steps help lower the average fitness function values. In addition, they also help reduce the standard deviation of the fitness function values. When both steps were added, the result is the best of the four combinations.

Our results with 20 runs were crossly compared with those applied with Cheng's algorithm with regard to whether the allocating genes are distinguishable or identical. The comparisons are shown in Table 2 and Table 3 below:

Table 2: Using Our Fitness Function

allocating genes	distinguishable	identical
ratio of average fitness function value	1.000	1.004
(standard deviation)/ average fitness function value	2.02%	1.91%
best fitness function value	119	120
worst fitness function value	127	127

Table 3: Using Cheng's Fitness Function

allocating genes	distinguishable	identical
ratio of average fitness function value	1.000	1.049
(standard deviation) / average fitness function value	5.39%	6.46%
best fitness function value	120	121
worst fitness function value	137	151

We observe that in applying both our and Cheng's fitness functions, the results with distinguishable allocating genes are better than those with identical allocating genes. That is because the distinguishable allocating genes could make the approximate solution fluctuate more easily so that good result would appear faster. In addition, our fitness function limits the deviation among genes, and that brings the solutions close to the optimal solution. In both the distinguishable allocating genes and identical allocating genes cases, our algorithm produced smaller fitness function values, for both the average and the worst cases.

We also compare the average performance factors of total processing time, total tardiness time, and total idleness time in our algorithm for the cases regarding whether the fitness function take the machine setup and machine idleness cost into consideration. The tests are run 20 times, and the result is shown in Table 4 below.

Table 4: Comparison of Fitness Functions

whether consider machine setup and idleness costs	total processing time	total tardiness time	total idleness time
No	311.810	10.700	27.810
Yes	307.200	10.410	23.220

It is obvious that the consideration of machine setup and idleness costs helps in producing better scheduling results.

5.Conclusion

Too many factors need to be checked for the production scheduling of parallel machines. Solving this problem through traditional optimization algorithms, like dynamic programming, takes too much computing time. That is impractical and not economical. On the other hand, solving through the heuristic rules often produces solutions not close to optimal solutions, due to considering too few factors. We proposed a genetic algorithm solution for this problem, with newly designed genes, evolutionary mechanism, fitness function, and additional steps of selection and bounce in the algorithm. We also applied real production data to verify the effectiveness of our solution.

More specifically, the two steps of selection and bounce were added to the traditional genetic algorithms to speed up the search of the approximation solution. We replaced the identical allocating gene in Cheng's algorithm with distinguishable allocating genes, which are the symbol '*' subscripted with a distinct number to assign jobs to machines. Our design of the evolutionary mechanism allowed these distinguishable genes to change positions. When there are no job genes between two allocating genes, the corresponding machine will be idle in the schedule, so that machine setup cost and human resource cost could be saved. In the fitness function, we included the consideration of machine setup and idleness costs. In reality, more factors should be taken into consideration, and this could be a future research topic. The additional bounce and select steps in our amended algorithm did improve the performance of the algorithm. In addition, the parameters and weights in the fitness function were determined by the requirements and job priorities. They need helps from domain experts and prior experiences. That makes the learning curve of the system high in the beginning.

The following are the future research topics:

(1) defining new modules of genetic algorithms for production scheduling:

The definition of genes and the evolutionary mechanisms could be further refined to make the probabilistic nature of the genetic algorithms more explicit. More factors should be considered for the fitness function to reflect the real production environment. Also, use of genetic algorithms for job scheduling of other types of production machines could be studied.

(2) improving the structure of the genetic algorithms:

In our algorithm, we bounce back to the initial solution to jump out of local optimum. Other methods could be investigated for solving this problem.

(3) application of the genetic algorithms in other fields:

Genetic algorithms could be applied to many other NP-complete and NP-hard problems, like routing, pattern recognition, prediction, equipment arrangement, etc. How to properly modify the genes and evolutionary mechanisms to fit these problems would be very interesting research topics.

References

[1] Aarts E. and J. Korst; Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing, Wiley, 1989.

[2] Adamopoulos, G.I. and C.P. Pappis; Scheduling Jobs with Different, Job-Dependent Earliness and Tardiness Penalties Using the SLK Method, European Journal of Operational Research, Vol. 88, pp336-344 , 1996.

[3] Ahmed, M.U. and P.S. Sundararaghavan; Minimizing the Weighted Sum of Late and Early Completion Penalties in a Single Machine, IIE Transaction, Vol. 22, pp288-290 , 1990.

[4] Armentano, V.A. and D.P. Ronconi; Tabu Search for Total Tardiness Minimization in Flowshop Scheduling Problems, Computers & Operations Research, Vol. 26, pp219-235 , 1999.

[5] Ben-Daya, M. and M. Al-Fawzan; A Tabu Search Approach For the Flow Shop Scheduling Problem, European Journal of Operational Research, Vol. 109, pp88-95, 1998.

[6] Chen, C.S; Search, http://cindy.cis.nctu.edu.tw/AI/ai1/Main.html

[7] Cheng, T.C.E; A Heuristic for Common Due-date Assignment and Job Scheduling on Parallel Machines, Operational Research Society, Vol. 40, pp1129-1135, 1989.

[8] Cheng, T.C.E. and C.C.S Sin; A State-of -the-Art Review of Parallel-Machine Scheduling Research, European Journal of Operational Research, Vol. 47, pp271-292, 1990.

[9] Cheng, T.C.E. and Z.L. Chen; Parallel-Machine Scheduling Problem with Earliness and Tardiness Penalties, Journal of the Operation Research Society, Vol. 45, pp685-695 , 1994.

[10] Cheng, R. and M. Gen, T. Tozawa; Minmax Earliness/Tardiness Scheduling in Identical Parallel Machine System Using Genetic Algorithms, International Journal of Computers and Industrial Engineering, Vol. 29, pp513-517, 1995.

[11] De, P. and J.B. Ghosh, C.E. Wells; Scheduling to Minimize Weighted Earliness and Tardiness about a Common Due Date, Computers & Operations Research, Vol. 18, pp463-475, 1991.

[12] Gao, Lin and Chengyao Wang, Dingwei Wang, Zhisong Yin, Shuning Wang; A Production Scheduling System for Parallel Machine in an Electrical Appliance Plant, Computers and Engineering, Vol. 35, pp105-108, 1998.

[13] Goldberg, David; Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, 1989.

[14] Gen, Mitsuo and R. Cheng; Genetic Algorithms and Engineering Design, Wiley, 1997.

[15] Graves, Stephen C; A Review of Production Scheduling, Operations Research, Vol.29, pp646-675, 1981.

[16] Hall, N.G. and M.E. Posner; Earliness-Tardiness Scheduling Problems I: Weighted Deviation of Completion Times about a Common Due Date, Operations Research, Vol.39, pp836-849, 1991.

[17] Holland, J. H;, Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, 1975.

[18] van Laarhoven P.J.M. and E.H.L. Aarts; Simulated Annealing: Theory and Applications, Reidel, 1987.

[19] Lee, C.Y. and S.L. Danusaputro, C.S. Lin; Minimizing Weighted Number of Tardy Jobs and Weighted Earliness-Tardiness Penalties about a Common Due Date, Computers & Operations Research, Vol.18, pp379-389, 1991.

[20] Liu, Jiyin and Lixin Tang; A Modified Genetic Algorithm for Single Machine Scheduling, Computers & Industrial Engineering, Vol. 37, pp43-46, 1999.

[21] Obitko, M; Genetic Algorithms, http://cs.felk.cvut.cz/~xobitko/ga/index.html

[22] Sridharan, V. and Z. Zhou; Dynamic Non-Preemptive Single Machine Scheduling, Computers & Operations Research, Vol. 24, pp1183-1190, 1996.

[23] Webster, S. and P. D. Jog, A. Gupta; A Genetic Algorithm for Scheduling Job Families on a Single Machine with Arbitrary Earliness/Tardiness Penalties and an Unrestricted Common Due Date, International Journal of Production Research, Vol. 36, pp2543-2551, 1998.