Using Genetic Algorithm to Solve Sequence Dependent Setup Time Jobs Scheduling Problem

Shih-Tang Lo
Department of Engineering Science, National Cheng Kung University
edwardlo@mail.ksut.edu.tw

Ruey-Maw Chen
Department of Electronic Engineering National Chin-yi Institute of Technology
raymond@ncit.edu.tw

Yueh-Min Huang
Department of Engineering Science, National Cheng Kung University
huang@mail.ncku.edu.tw

Abstract*-Scheduling problems exist in many applications and most of them have demonstrated as NP complete. A lot of schemes introduced to solve scheduling problem. Among them, genetic algorithm (GA) is successfully used for similar problems in the past, and leading toward much more successful on some NP-complete domain than others. In this paper, we presented a genetic algorithm to solve "sequence dependent setup time" jobs scheduling problem. The objective of this paper is to minimized the make span and (or) maximum machine utilization. Simulation results demonstrate that the proposed method can find a good solution and is subjected to user requirements, if the user has different objective considerations.*

Keywords: Scheduling, genetic algorithm (GA), Setup time

1. Introduction

Many applications involve the concepts of scheduling, such as communications, routing, and production planning and task assignment in multi-processor. Most problems in these applications are categorized into the class of NP-complete problems. It would take a lot of time to get an optimal solution, especially for a large-scale scheduling problem.

Most of the studies dealing with scheduling have been confined to optimization of a single criterion. Many researchers have considered a variety of scheduling problems with a single performance measure such as the average flow time, maximum completion time or tardiness [1-2].

Scheduling decisions frequently involve considerations of more than one criterion [3-6]. The bi-criteria scheduling problems are generally divided into two classes. First one involves minimized one criterion subject to the constraint that the other criterion has to be minimized. Second class of problems, both criteria are considered equally important and the problem involves finding efficient schedulers [7]. The problem considered in this paper belongs to this class.

This study considers the problem of scheduling N jobs on identical parallel machines, with processing time $p_1, p_2,\ldots,p_n$, respectively. These jobs have different job types. The setup time is small, if one job and its immediate successor job are the same job type. Setup time between jobs make machine idle, increasing the producing cost. Nevertheless, the setup time is always omitted or included in processing time on most scheduling problems. Therefore, this paper focuses on job sequence arrangement under setup time consideration.

There are various researches in parallel machine scheduling problem. This paper addressed sequence dependent setup time jobs scheduling problem on unrelated parallel machines. The minimization of complete time, tardiness and make-span are the common objectives on most study. Karp[8] and HO[9] had showed that the minimization of total tardiness in identical machine was NP-hard problem[10].

Hopfield started using an artificial neural network (ANN) to solve optimization problems [11]. Since that, Hopfield networks have been successfully applied to a variety of applications. As generally known, most scheduling problems are combinational, thereby ensuring the optimization process by an ANN . However, these researches are basically non-adaptive networks, of which the neural unit connection weights and biases must be prescribed before applying of the networks to a particular problem. These networks also have drawbacks such as failing to converge to a valid solution, inability to locate the global minimum and poor scaling properties due to the use of quadratic energy function. Additionally, most scheduling problems are limited to the preemptive and migratory processes on a multiprocessor and, therefore, only consider the timing constraints. Simulated Annealing (SA) was introduced by Metropolis in 1953 [12]. SA is also to solve a widely complex problems and a lot of solution combinations [13], but time consuming is a significant concern. Furthermore, GA has been considered as a powerful heuristic search method to solve combinational optimization problems [14].

In our previous work, we studied multiprocessor scheduling problem using ANN and Normalized Mean Field Annealing (MFA) [15-16]. In this paper, we utilized a genetic algorithm to solve sequence dependent setup time jobs scheduling problem. Simulation results demonstrate that suggested method finds a good solution and is subject to user requirements, if the user has different objective considerations. If the deadline time constraint is more important, then the user can increase the weight factors α or (and)γ, otherwise increase the weight factors β or (and)γ of setup time and idle time constraint. Therefore, the investigated scheduling system is adaptive to user requirement.

2. Problem Definition and Characteristics

Most scheduling problems are limited to the preemptive and migratory jobs on machines, therefore, only consider the timing constraint. Suppose the system has *M* identical parallel machines and *N* jobs with different types. A job is non-segmented and non-preemptive. The execution time of each job is predetermined. The set up time matrix is predefined base on the job type characteristics. The machine is idle while setting up or completes the jobs before the maksspan, causing low utilization. The set up time is small or zero if concessive job has the same job type. Given these assumptions, this work is to find a schedule, which to minimize maximum complete time and to maximize the machine utilization. Hence, the job processing sequence and makepan is the most important factor on scheduling problem. This paper uses Genetic Algorithm (GA) to resolve such scheduling problem; considerations are listed as follows:

(1) Each job has only one operation; no jobs can be processed on more than one machine simultaneously
(2) Machines are all identical. Each machine can process one job at a time.
(3) Jobs' processing time is known, and setup time is determined by job type sequence

3. Genetic Algorithms

According to Darwin's theory of survival of fittest, a genetic algorithm is an optimization process based on the evolution of nature. Using the natural selection and exchange of genetic information, the species with the optimal fitness govern the word. The genetic algorithm is a search algorithm on the basis of biological principles of crossover, mutation and selection process. This algorithm maintains a population of candidate solutions that evolves over time and ultimately converges. Individuals in population are represented with chromosomes. Each individual has a numeric fitness value obtained from fitness function that measures how well of this solution [17].

The basic GA contains three parts:

(1) Chromosome representation involves the parameters combination of handling problems, and different genetic representation depicts different solutions of problems. A Binary representation is usually used for the coding of each solution.
(2) Fitness function can determine the category of the parameters combination within a set. GA uses this information to determine which combinations of parameters would survive.
(3) Genetic operators are the kernels of GA. Their primary task is to synthesize new combinations of parameters in accordance with the fitness function of each parameters combination to achieve the target of evolution. GA contains three operators. The selection operators select the fittest individuals of the current population to serve as parents of the next generation. The crossover operator chooses randomly a pair of individuals and exchanges some part of the information. The mutation operator takes an individual randomly and alters it.

The structure of GA is an iteration composed of a selection step followed by a sequence of crossovers and mutations procedures. Finally, GA is executed until some termination state is achieved, such as the number of iterations, execution time, result stability, etc. This paper is deal with sequence dependent job setup time and machine utilization problem. The proposed algorithm is using most common genetic operator by the past research to verify our scheduling problem. The following are the GA operators in this scheduling system.

3.1 Genetic Expressions and Initial Population.

The coding of individual δ is composed of m strings $(\delta_1,\delta_2,\ldots\ldots,\delta_{m-1},\delta_m)$, δ_j is the jobs sequence on machine *j*. There is one to one correspondence between machines and strings, where each string represents the jobs assigned to some specific machine. Restated, string δ_j represents a list of jobs to machine *j*. Figure 1 shows an example of a coding for three machines, job *1,3,4* are processed in machine 1.

$$\left(\underbrace{1,3,4}_{\delta_1} ,*, \underbrace{5,2}_{\delta_2} ,!*, \underbrace{6,7}_{\delta_3} \right)$$

Figure 1 Genetic expression for 3 machines

Since the initial population has been produced randomly in most GA researches, it always requires

longer search time to obtain an optimal solution. There is a lot of scheduling system using heuristic rule to generate the initial population. But this will decreases the diverse of individuals for searching an optimal solution. In this investigated scheduling system, the initial population is generated randomly.

3.2 Crossover Operator

The crossover operator focuses on creating alternative solutions around the best solution. Generally, crossover is an operation used to generate a new string (i.e., child) from two parent strings. It provides a mechanism for parameters to mix and match through random processes. The operation of crossover operator is described as follows (Figure. 2). :

(1) Using a random process to select two parents individual sequence string *Parent1* and *Parent2* that have different jobs schedules from the old generation.
(2) Generate *k* as random number between *1* to *N* to exchange string into new offspring.
(3) Select the first *k* members of *Parent1* and the last *N* -*k* of *Parent2* save them in the new offspring.
(4) If the new offspring is not feasible, some jobs may be duplicated. We replace these duplicated jobs by the randomly selected short jobs. Job *1* and *4* are duplicated, job *2* and *6* are short. Replace one of job *1* with job *2* or *6*; job *4* is done in the same way.

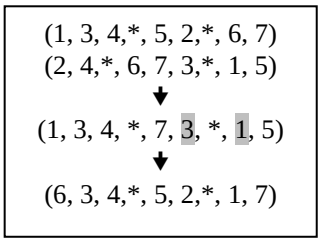

Figure. 2 Crossover operation (if *k*=4)

3.3 Mutation Operator

Mutation operation is used to randomly select one chromosome with a pre-specified probability. In order to avoid the risk of staying in the local maximum, a mutation operator is proposed herein. Exchange two randomly choused jobs of an individual string. Figure.3 describes the suggested mutation operation in this study. The operator of this operation is consists of the following steps:

(1) Randomly generate two integers *k* and *s* between 1 to *N*
(2) Swap jobs that are in the *k*th and *s*th position in an individual sequence.

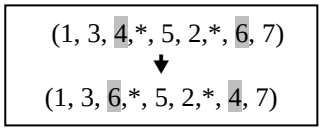

Figure. 3 Mutation Operation

3.4 Fitness Function

Minimizing the *machine utilization* (*setup time, idle time*) and *make span* of the job scheduling is the goal of this study. Accordingly, fitness function is defined correlating to those targets as depicted below. In this paper, we introduce idle time in fitness function. The idle time indicates that the machine does nothing, hence decreases the machine utilization. The fitness value of each individual sequence is computed by fitness function with respective weights. α ,β and γ are weight factors based on the user's requirements. Using different weights, one may expand the dimensions of the objective function and provide solutions that incorporate multiple objective criteria.

$$f(e) = \frac{1}{\alpha * MakeSpan + \beta * SetupTime + \gamma * IdleTime}$$

Figure.4 Fitness function

- *Makespan* is the time of the last job in the sequence leaves the system. *Makespan* can be computed as the complete time of the *n*th job in the sequence.

  $C_i = C_{ji} + S_{ji} + P_i$

  Where
  - C_i is the complete time of the *i*th job in the sequence
  - S_{ji} is the setup time between the *j*th job and the *i*th job (the *j*th job is an intermediate predecessor of *i*th job) in the sequence, in which the setup time is predefined in setup time matrix.
  - P_i is the processing time of the *i*th job in the sequence.
- *Setup tTime* is computed by the summation of S_{ji} showed in the sequence.
- *Idle time* is computed as difference between total machine capacity and total processing time plus setup time. The idle time is dependent to makespan and setup time.

3.5 Selection Operation

The selection is done using a *Roulette Wheel* principle. Thus, the better fitness of an individual, the higher probability of it will be selected. The process of creating new generation continues until a given number of generations are achieved or the cost of a given solution is achieves an acceptable level. In this study, the termination conditions is set to number of generations equal 50 or the best solution will never get improved after 10 times.

4. Simulation considerations and results

To evaluate the scheduling system, the simulation involved 15 jobs and 3 machines example (Table 1). Jobs are categorized to three types *A, B,* and *C*. Setup time matrix is depicted in Table 2. The first column indicates the processing job's type and the first row represents the immediately successive job's type. The values in the matrix are setup time. For example, the setup time is 3 if processing job's type is "*B*" and the immediate successive job's type is "*A*". The setup time is 0 when the job types in a row are the same. Three different cases corresponding to the different scheduling objectives are simulated.

First one is the typical case; make span is the only scheduling objective (α=1, β=0, γ=0) (Fig 5). In this simulation result we found that the make span is not minimum, and the machine utilization is equal to 75/(3*35)≅0.806. The machine utilization is lower than the others case. The second case is to maximum the machine utilization (i.e. minimum the total set up time and idle time, set α=0, β=1, γ=1) (Fig 6). We found that the total set up time and idle time is minimized as possible. The last case is set α=0.2, β=0.8, γ=0.8 (Fig 7). In this case the machine utilization is .75/(3*28)≅0.892 after the number of generation is 46, the best case in these three cases. All cases set the mutation rate, crossover rate as 0.02 and 0.4 respectively. The simulation results show that jobs have the same type seems to be arranged to together as possible if taking setup time into account as shown in Fig. 6 and 7. One job which type is "*A*" will not be scheduled immediate before one job which type is "*C*" (the setup is 6). And in Figure 7, job *8,12,2,5,3* which type "*B*" are scheduled in the same machine *2*.

Figure 8 show the simulation results of makesapn after 50 generations. It shows that using proposed fitness function gets a better solution in different weight factor simulation results. If setting α=0, β=1, γ=1, the simulated scheduling result is better than setting α=1, β=0, γ=0. Table 3 shows the same conclusions in different simulation runs.

Figure 9 demonstrates the resulting machine utilization based on proposed fitness function. The case of considering makespan (α=1, β=0, γ=0) yields the worse results than that of other cases. Accordingly, the system is more efficient than only concerning makespan. The proposed algorithm converges in about 15 generations, if apply to other job cases, different setup time matrix or no setup time included.

In this study, we employed a new idea to design a scheduling system; the scheduling database is created. We translate the scheduling constrains into database table. The setup time matrix can be modified by the setup cost, or add setup time cost constraints into fitness function. Because the setup cost really reflect the main concerned by the enterprise managers.

Table 1: Simulation jobs data

Job#	Process Time	Job Type
1	5	C
2	5	B
3	3	B
4	7	A
5	8	B
6	7	C
7	2	A
8	6	B
9	2	A
10	6	A
11	5	A
12	6	B
13	2	C
14	8	A
15	3	B
Total processing time	75	

Table 2 Setup time matrix

	A	B	C
A	0	2	6
B	3	0	2
C	1	5	0

Machine	Job# /complete time				
M1	2	8	1	4	3
	5	11	18	26	31
M2	10	11	14	7	12
	6	11	19	21	29
M3	15	5	13	9	6
	3	11	15	18	31

Figure 5 Scheduling result after iteration=46, α=1, β=0, γ=0, Make span= 31, Setup time= 16, Idle time=2

Machine	Job# /complete time					
M1	1	13	10	11	4	
	5	7	14	19	26	
M2	15	12	5	9	7	3
	3	9	17	22	24	29
M3	6	14	2	8		
	7	16	23	29		

Figure 6 Scheduling result after iteration=11, α=0, β=1, γ=1, Make span= 29, Setup time= 9, Idle time=3

Machine	Job# /complete time				
M1	8	12	2	5	3
	6	12	17	25	28
M2	11	10	7	14	9
	5	11	13	21	23
M3	4	15	13	1	6
	7	12	16	21	28

Figure 7 Scheduling result after iteration=46,α=0.2, β=0.8, γ=0.8,Make span= 28,Setup time= 4, Idle time=5

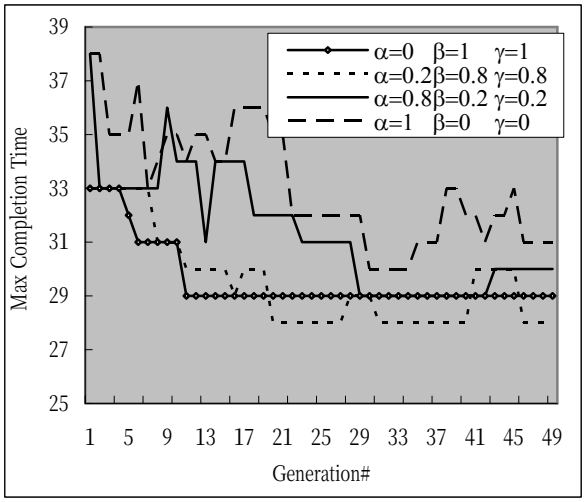

Figure 8 Makespan under different weight factors

5. Conclusions

In this paper, genetic algorithms were applied to a job scheduling problem. Most scheduling problems concern minimizing the make span. The setup time is always is included in processing time or ignored. Such scheduling problem can't reflect a scheduling problem in the real world. In real situation, the user may have dynamic objective based on the environment circumstances. In this paper, we introduced the scheduling algorithm with sequence dependent setup time and make span considerations. And we employed the idle time into fitness function, although the idle time is computed from makespan. It is easily to converge than use makespan only.

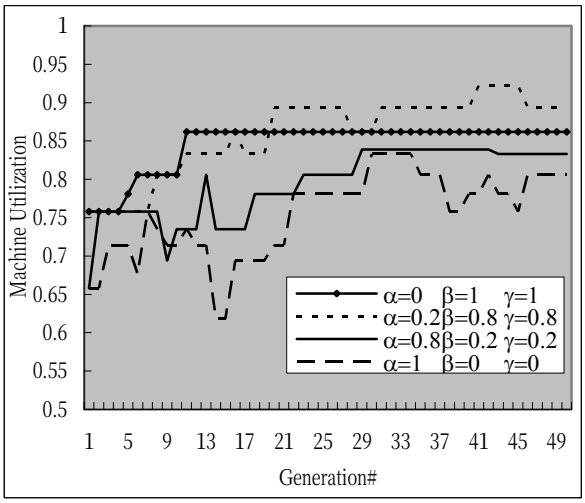

Figure 9 Machine utilization performances

According to simulation results, GA is confirmed to be a better scheme to fit the user requirements. Most researches use problem constrains to solve the scheduling problem. They have to modify the corresponding system case by case. But in real situation, different conditions and scheduling objectives exist in manufacturing company. They have different decision-making consideration. GA are powerful but usually suffer from longer scheduling time. Genetic algorithms can circumvent many inappropriate solutions, and speed up searching efficiency significantly [18]. Another advantage of using genetic algorithms is their flexibility. Accordingly, GA is used in solving many similar scheduling problems. For the constrained problems, the searching efficiency can be improved by means of the proper modification of genetic representations and genetic operators.

Future research in this field may combine with other artificial intelligence techniques, thus improving the solution quality and accelerating the calculations. The continued research is to take more consideration in real case, such as job structure (or the job operation precedence) or different machine types.

Table 3 Simulation result

	Weight Factor											
	α=0 β=1 γ=1			α=0.2 β=0.8 γ=0.8			α=0.8 β=0.2 γ=0.2			α=1 β=0 γ=0		
Run	Generation	Makespan	Setup+Idle	Generation	Makespan	Setup+Idle	Generation	Makespan	Setup+Idle	Generation	Makespan	Setup+Idle
1	35	31	18	21	31	18	37	28	9	51	29	12
2	5	30	15	6	30	15	17	29	12	30	32	21
3	6	30	15	8	31	18	29	31	18	25	33	24
4	17	29	12	12	28	9	17	33	24	36	31	18
5	7	30	15	7	31	18	27	29	12	17	30	15
6	7	31	18	7	28	9	75	30	15	12	31	18
7	28	28	9	21	31	18	18	28	9	13	31	18

References

[1] S. Tzafestas, A. Triantafyllakis," Deterministic scheduling in computing and manufacturing systems: a survey of models and algorithms. Mathematics and Computers in Simulation," 35(5):397 – 434,1993.

[2] K.R. Baker, G.D. Scudder," Sequencing with earliness and tardiness penalties: A review," Operations Research ;38(1):22 – 36,1990.

[3] Ruiz-Diaz F, French S, "A survey of multi-objective combinatorial scheduling," Multi-objective decision making. New York, NY: Academic Press; p. 59 – 77. ,1983.

[4] Dileepan P, Sen T. ,"Bicriteria state scheduling research for a single machine," Omega;16:53-9.,2000

[5] Murata T, Ishibuchi H, Tanaka H.," Multi-objective genetic algorithm and its applications to flowshop scheduling.",Computer and Industrial Engineering ;30(4):957 – 69,1996.

[6] Allahverdi A. "The two- and m-machine flowshop scheduling problems with bicriteria of makespan and mean flow time." European Journal of Operational Research;147(2):373 – 96.,2003.

[7] Sang M. Lee and Arben A. Asllani, "Job scheduling with dual criteria and sequence-dependent setups: mathematical versus genetic programming," Omega, Volume 32, Issue 2, April 2004, Pages 145-153, April 2004.

[8] Karp RM. Reducibility among combinatorial problems: complexity of computer computations. NewYork: Plenum Press, pp. 85–103, 1972.

[9] J.C. Ho, Y. L. Chang, "Minimizing the number of tardy jobs for m-parallel machines" in European Journal of Operating Research, vol. 84, Pages343–55.1995;

[10] D.W. Kim, K. H. Kim, W. Jang and F. F. Chen,"Unrelated parallel machine scheduling with setup times using simulated annealing ," in Robotics and Computer-Integrated Manufacturing, Volume 18, Issues 3-4, June-August 2002, Pages 223-231

[11] J. J. Hopfield and D.W. Tank ,"Neural computation of decisions in optimization problems," Biol. Cybern., vol. 52,pp.141-152,1985.

[12] S. Yang and D. Wang, "Constraint satisfaction adaptive neural network and heuristics combined approaches for generalized" in IEEE Transactions on. Neural Networks, 2000, vol. 11, pp. 474–486.

[13] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," Science, vol. 220, pp. 671–680, May 1983.

[14] H. Topcuoglu, S. Hariri , and M.Y. Wu., "Performance-effective and low-complexity task scheduling for heterogeneous computing" in IEEE transactions on parallel and distributed systems, 2002, vol. 13, pp. 260–274.

[15] Y. M. Huang and R. M. Chen, "Scheduling Multiprocessor Job with Resource and Timing Constraints Using Neural Networks," in IEEE Transactions on Systems, Man, and Cybernetics—Part B: Bybernetics, Vol. 29, no. 4 , pp490-502, august 1999

[16] Y.M. Huang and R.M. Chen, " Multiconstraint task scheduling in multi-processor system by neural network," in Tools with Artificial Intelligence, 1998. Proceedings. Tenth IEEE International Conferenc, pp288 -294 e, 1998

[17] Y. Z. Wang ,"Using genetic algorithm methods to solve course scheduling problems," in Expert Systems with Applications, Volume 25, Issue 1, Pages 39-50 , July 2003

[18] Dirk C. Mattfeld and Christian Bierwirth ,"An efficient genetic algorithm for job shop scheduling with tardiness objectives," European Journal of Operational Research, Volume 155, Issue 3, Pages 616-630 ,June 200.