2002 International Computer Symposium (ICS2002)
Workshop on Artificial Intelligence
December 18-21, 2002

Solving Non-sharable Machine and Resource Scheduling Problem using Simulated Annealing Algorithm

Shih-Tang Lo and Yueh-Min Huang[+]

Department of Information management, Kun Shan University of Technology, Department of Engineering Science, National Cheng Kung University[+], Taiwan, R.O.C.

Email: edwardlo@mail.ksut.edu.tw, raymond@mail.ncku.edu.tw[+]

Tel:06-2732726 Fax:06-2732726

Abstract

Scheduling problems exist in many applications, and most of them have demonstrated their complexities as NP complete. A lot of schemes have been introduced to solve scheduling problems, such as linear programming, artificial neural network and fuzzy logic. Among them, simulated annealing is a highly effective means of obtaining an optimal solution which is capable of preventing the local minimum. In this paper we try to use simulated annealing algorithm to solve a non-sharable machine and resource-based scheduling problem which closely reflects a scheduling problem in the real world. We assume that both machine and resource are not sharable, and each job to process requires machine and resource. In contrast to the most scheduling problems which only resolve the machine problem, job process time and deadline, we believe that resource constrains are critical issues, although it will make scheduling problem more complicated. In this work, we try to use a genetic algorithm(GA)-based simulated annealing algorithm to solve the scheduling problem in reducing the penalty of resource factor involved. Simulation results demonstrate that our method not only can solve machine and resource constraint problem, but also can work effectively.

Key words: simulated annealing, scheduling, genetic

Solving Non-sharable Machine and Resource Scheduling Problem using Simulated Annealing Algorithm

Shih-Tang Lo and Yueh-Min Huang+

Department of Information management, Kun Shan University of Technology, Department of Engineering Science, National Cheng Kung University, Taiwan, R.O.C+

edwardlo@mail.ksut.edu.tw, raymond@mail.ncku.edu.tw+

Abstract

Scheduling problems exist in many applications, and most of them have demonstrated their complexities as NP complete. A lot of schemes have been introduced to solve scheduling problems, such as linear programming, artificial neural network and fuzzy logic. Among them, simulated annealing is a highly effective means of obtaining an optimal solution which is capable of preventing the local minimum. In this paper we try to use simulated annealing algorithm to solve a non-sharable machine and resource-based scheduling problem which closely reflects a scheduling problem in the real world. We assume that both machine and resource are not sharable, and each job to process requires machine and resource. In contrast to the most scheduling problems which only resolve the machine problem, job process time and deadline, we believe that resource constrains are critical issues, although it will make scheduling problem more complicated. In this work, we try to use a genetic algorithm(GA)-based simulated annealing algorithm to solve the scheduling problem in reducing the penalty of resource factor involved. Simulation results demonstrate that our method not only can solve machine and resource constraint problem, but also can work effectively.

Key words: simulated annealing , scheduling, genetic

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.

This fact implies that an optimal solution for a large-scale scheduling problem is quite time-consuming. Hopfield first used an artificial neural network (ANN) to solve optimization problems [1]. Since that, Hopfield networks have been successfully applied to a variety of problems. Willems and Rooda translated the job-shop scheduling problem into a format of linear programming and, then, mapped it into an appropriate neural network structure to obtain a solution [2]. Foo and Takefuji employed integer linear programming neural networks to solve the scheduling problem by minimizing the total starting times of all jobs with a precedence constraint [3]. Meanwhile, Zhang *et. al.* proposed a neural network method based on linear programming. In the proposed algorithm, preemptive jobs are scheduled on the basis of their priorities and deadlines [4]. The above investigations concentrated on the preemptive jobs (processes) executed on multiple machines (multiprocessor) with job transfer allowed by an ANN. Also in [5], Hanada and Ohnishi developed a parallel algorithm based on a neural network for preemptive task scheduling problems by allowing a task transfer among machines. As generally known, most scheduling problems are combinatorial, thereby ensuring the optimization process by a ANN. However, the mentioned above neural networks 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 [6]. In addition, most scheduling problems are limited to the preemptive and migratory processes on a multiprocessor and, therefore, only consider the timing constraint. In some multiprocessor systems, task scheduling does not include only a timing constraint [7]. For instance, the *display system* on an advanced avionics system may consist of two or more display processors. Each processor is responsible for different tasks containing the timing constraint without allowing task migration between processors. Restated, tasks do not use the same resource simultaneously. To facilitate the pilot's control action, all tasks must be properly scheduled to provide the pilot with some useful information. Otherwise, a hazardous situation is inevitable.

In our previous work, we used ANN and Normalized Mean Field Annealing (MFA) to solve multiprocessor scheduling problem, and proved our algorithm to be more efficient to converge [7][8].

On the other hand, GA has been considered as a powerful heuristic search method to solve combinational optimization problems. Although they provided good quality of schedules, their execution times are significantly higher than those of other alternatives [9][10]. Hajri Introduced a controlled GA based on fuzzy logic and belief function to solve job-shops scheduling problems [11]. They used an efficient representational scheme including heuristic rules to represent the solution. Dessoaly investigated the hybrid approach combing GA with ANN to solve the short-term generation scheduling. They showed the hybrid technique had higher potential to enhance performance, accelerate converge, and overcome the problems associated with the traditional GA approach [12].

Simulated Annealing (SA) was introduced by Metropolis in 1953 [13], its basic concept came from one state transfer to another state in chemical production processes. Then it was applied to approximate the solution of very large combinatorial optimization problems [14].

The original Metropolis scheme is that: An initial state of a thermodynamic system is chosen at energy *E* and temperature T. Holding T constant, the initial configuration is perturbed and the change in energy $\triangle E$ is computed. If the change in cost function is negative, the transition is unconditionally accepted; if the cost function increases the transition is accepted with a probability based upon the Boltzmann distribution. This process is then repeated sufficient times to give good sampling statistics for the current temperature. The temperature decremented and the entire process repeated until a frozen state is achieved at $T=0$.

SA is one of the best methods to solve a widely complex problems and a lot of solution combinations. Consequently the four key ``ingredients'' for the implementation of simulated annealing are:

(1). The definition of configurations;

(2). A generation mechanism, i.e. the definition of a neighborhood on the configuration space;

(3). The choice of a cost function;

(4). A cooling schedule.

In this paper, we modify the energy function presented in [7][8], introduce the resource base concept into our scheduling problem, and apply the GA-based representation approach with SA algorithm to find a better solution.

2. Scheduling problem

Our scheduling problem domain considers *N* jobs (or processes), *M* machines (or processors) and *R* resources. T is the maximum completion time of the jobs. The machines and resources are heterogeneous and non-sharable in our case. These *N* jobs are divided into *M* groups, each group of jobs must be processed on dedicated machine and using a specify resource. There are some assumptions in our scheduling problem domain. First, the execution time of each job is predetermined. Second, each machine can perform at most one job at any time, and each job cannot be interrupted during it perform (non-preemptive). Finally, in each time, only one job can be performed at one machine using one resource. Our objective is to find a schedule with no conflict on these jobs in executing, base on non-sharable machine and resource.

Here we introduce our GA base representation method. For simplicity we use a matrix M_{ijkp} to denote the job i executed on machine j at certain time k using the resource p. This matrix is built like a gene of GA, which is easy to implement, and it will be used in our continue research. If job i is processing on machine j at certain time k using the resource p, then $M_{ijkp} = 1$, else $M_{ijkp} = 0$.

(1) Non-sharable machine constraint

Since a machine j can only execute one job i at a certain time k, the energy term can be defined as follow:

$$A=\sum_{k=1}^{T} \sum_{p=1}^{R} M_{ijkp}, i = 1..N, j = 1..M \text{ ....(1)}$$

$$\begin{cases} E'_{ij} = 1, & if\ A > 1 \\ E'_{ij} = 0, & if\ A \le 1 \end{cases}$$

If the value is greater than 1, it mean that there are more than two job using same resource at the same time.

(2). Non-sharable resource constraint

We defined the other energy function for non-sharable resource constraint as follow:

$$B=\sum_{i=1}^{N} \sum_{j=1}^{M} M_{ijkp}, k = 1..T, p = 1..R \text{ ....(2)}$$

$$\begin{cases} E'_{kp} = 1, & if\ B > 1 \\ E_{kp} = 0, & if\ B \le 1 \end{cases}$$

When one job is assigned on a dedicated machine at time k using resource p, the energy will greater than 1 if there is still another job has to do at the same time. If this machine only process one job at time k, the energy value equals 1. If there is no job be processed at time k the energy value is 0,

(3) Timing constraint

In our work, we only consider the non-sharable machine and resource. There is no permission for one machine running more than one job at the same time. Each job has to be processed between the arrival time and deadline. Due to the content limitation, we only focus the effort on the coupling of the machine, resource and time problem. We believed that it is easy to incorporate other constrains into our scheduling problem. Thus the timing constraint energy is defined below.

$$EH'_{ij} = \sum_{k=1}^{T} H(G_{ijk}), i = 1..N, j = 1..M \text{ .... (3)}$$

$$G_{ijk} = Job_i - arrival \quad time - k = k - Job_i \quad deadline, \quad if \quad M_{ijkp} = 1, p = 1..R$$

$$\begin{cases} H(G_{ijk}) = 1, & if \quad G_{ijk} > 0 \\ H(G_{ijk}) = 1, & if \quad G_{ijk} \le 0 \end{cases}$$

For each job, it has arrival time and deadline. If a job completion time is over the deadline, we set the energy value to 1.

Finally, our energy function is defined as the follow equation:

$$\text{Energy}=c_1\sum_{i=1}^{N}\sum_{j=1}^{M}E'_{ij}+c_2\sum_{k=1}^{T}\sum_{p=1}^{R}E'_{kp}+c_3\sum_{j=1}^{M}\sum_{i=1}^{N}EH'_{ij}\ \ldots\ldots\ldots\ldots(4)$$

The energy will be zero if there is a feasible solution.

Here we show an example using data in Table 1 without time constraints. If we adopt the scheduling depicted in Figure 1, there is no legal solution existed. That is because the resource R_2 is shared by *job$_1$* and *job$_6$* at time 3 to 4, the energy is 2. If we adopt the scheduling depicted in Figure 2, there is a feasible result since the energy is 0. The resource R_2 is not shared by *job$_1$* and *job$_6$* at the same time.

Table1.Example with 6 jobs/3 machines/2 resources

job	*machine*	*time*	*resource*
Job1	M1	2	R2
Job2	M1	3	R1
Job3	M2	4	R1
Job4	M2	3	R2
Job5	M3	2	R1
Job6	M3	3	R2

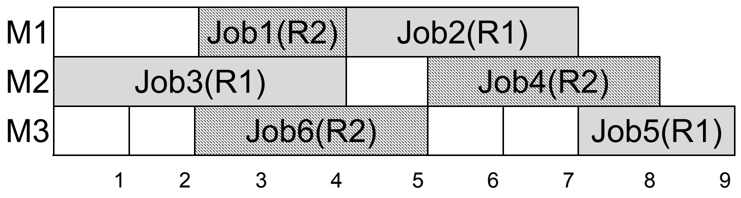

Figure1. An example is not a feasible scheduling result for Table 1

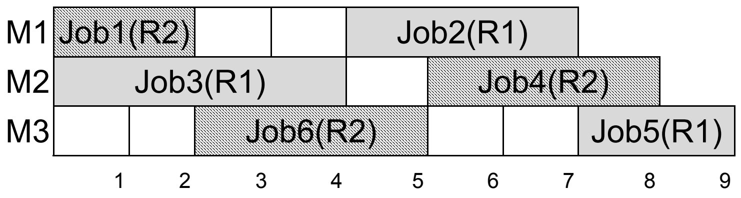

Figure 2. A feasible scheduling result of Table 1

3. Algorithm framework

The scheme of our SA algorithm is shown in Table 2, the algorithm begins with the creation of an initial solution, some jobs are chosen randomly (base on the number of job) to reschedule for each generation, and the energy value is computed based on equation (4). If the energy is lower than before, then we accept the new solution. But if the energy is higher than before, we might accept the solution base on the probability *Pa*, if *Pa*< δ . This algorithm will be terminated until the energy goes to zero. If energy is zero then output the schedule result.

Table 2.Simulated annealing algorithm

```
Initial a solution $S$, compute $E$
Set the initial $T, k, r$
While $E$ <> 0
    $S'$=Random choose one job to reschedule
    Compute energy $E'$ and $\Delta E=E'-E$
    if $\Delta E<0$ then accept $S=S', E=E'$
            else Compute the $\delta = e^{-\frac{\Delta E}{kT}}$
                Accept the new solution when $Pa<\delta$
    decrease the temperature $T=T*r$
```

We can only conclude that the scheduling result is a feasible solution. We can't prove that it is the optimal one. However, we will use some weight function or heuristics to get a near optimal solution in our future work. Our cooling schedule is defined as $T = T * r$, $0<r<1$.

4. Numerical experiment

There are some examples examined for different situations in our experiment. First we show the job is not segmented, it means that the job can not be preempted until its finish. And we set *T=T*0.95* for the cooling process, the results shown in Figure 2. Figure 3 and Figure 4 show the schedule results for the job that can be segmented. Figure 5 shows the scheduling results using the data in Table 4 for the job with time constraint (arrival time/deadline). Figures 6 and 7 are the energy evolution result which the time constraints is include or exclude. It shows that there is higher probability to get a worse answer when the temperature gets higher. Figure 8 is our cooling scheduling.

Table 3.Example with 6 jobs/2 machines/2 resources

job	*machine*	*time*	*Resource*
Job1	M01	2	R02
Job2	M01	3	R01
Job3	M01	4	R01
Job4	M02	3	R02
Job5	M02	2	R01
Job6	M02	3	R02

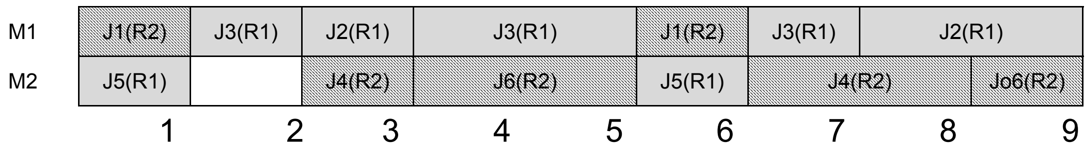

Figure 3. A feasible scheduling result for example of Table 3 which the job can be segmented

Table 4.Example with 10 jobs/4 machines/3 resources

Job	*machine*	*time*	*resource*	*Arrival time*	*Deadline*
Job1	M01	2	R02	1	10
Job2	M01	3	R01	5	14
Job3	M01	4	R01	1	14
Job4	M02	3	R02	1	10
Job5	M02	2	R03	5	14
Job6	M02	3	R02	1	14
Job7	M03	5	R03	1	10
Job8	M03	3	R02	5	14
Job9	M04	3	R02	5	14
Job10	M04	4	R01	1	10

M1	J2(R1)			J2(R1)	J1(R2)	J3(R1)	J1(R2)		J2(R1)	J3(R1)	J3(R1)	J3(R1)		
M2	J5(R3)	J4(R2)	J6(R2)	J4(R2)				J6(R2)	J6(R2)				J4(R2)	J5(R3)
M3		J7(R3)	J7(R3)			J8(R2)	J7(R3)		J7(R3)		J3(R3)	J8(R2)		J8(R2)
M4	J9(R2)		J10(R1)		J10(R1)			J10(R1)		J9(R2)	J9(R2)			J10(R1)
	1	2	3	4	5	6	7	8	9	10	11	12	13	14

Figure 4. A feasible scheduling result for example of Table 4 which the job can be segmented

M1			J1(R2)			J3(R1)		J1(R2)	J3(R1)	J3(R1)	J2(R1)	J2(R1)	J2(R1)	J3(R1)
M2	J4(R2)	J4(R2)		J6(R2)	J6(R2)			J5(R3)	J6(R2)	J4(R2)	J5(R3)			
M3			J7(R3)	J7(R3)		J7(R3)	J8(R2)		J7(R3)	J7(R3)	J8(R2)		J8(R2)	
M4	J1(R1)		J1(R1)		J1(R1)	J9(R2)	J1(R1)					J9(R2)		J9(R2)
	1	2	3	4	5	6	7	8	9	10	11	12	13	14

Figure 5. A feasible scheduling result for example of Table 4 which the job can be segmented include arrival time and deadline

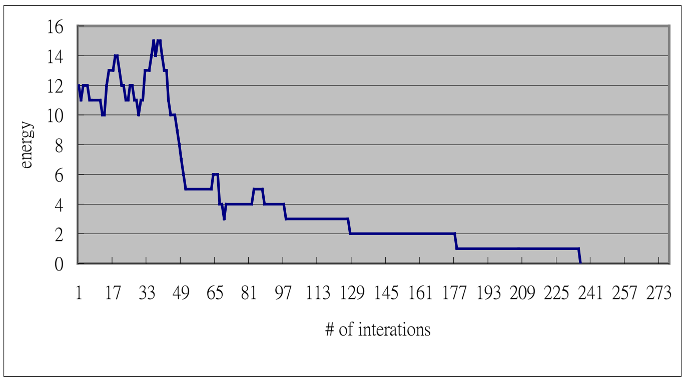

Figure 6. Energy evolution of Simulated Annealing without time constraint

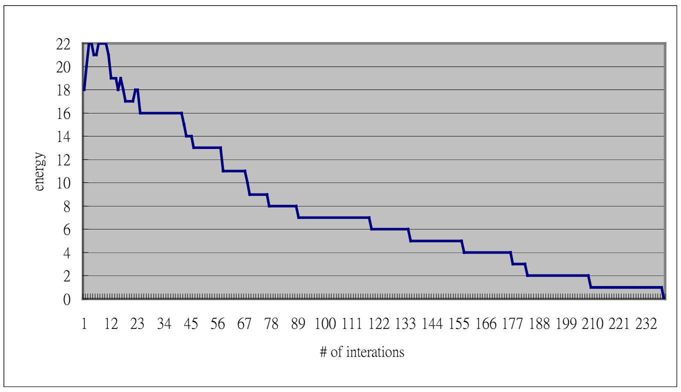

Figure 7. Energy evolution of Simulated Annealing with arrival time/deadline

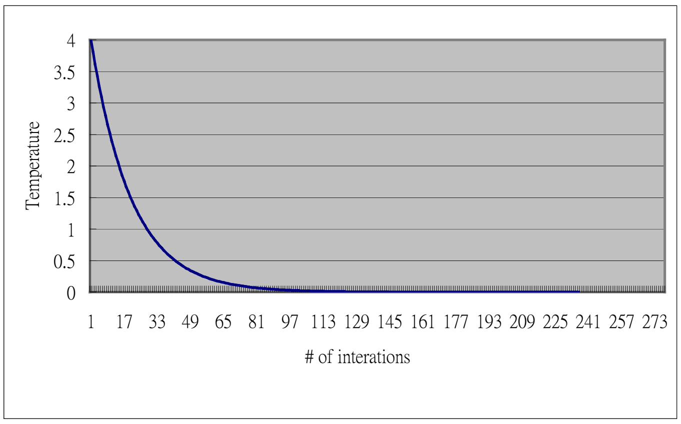

Figure 8. Cooling scheduling of Simulated Annealing (T=T*0.95).

In our simulation, we find the simulating annealing method can find a feasible solution. However the variance of number of execution iterations sometime is high. The initial temperature, cooling process and initial solution are important factor of result.

We conclude that:

(1.) The parameter is significant factors to SA.

(2.) The range of neighborhood answer is an impact to the number of iterations.

(3.) We cannot guarantee to get an optimal solution.

(4.) If the cooling schedules decrease too fast, it may fall into a local minimum solution or non-converge.

(5.) The initial solution sometimes cannot converge in a reasonable number of iterations.

5. Conclusions

In this paper, we have presented a GA-representation SA algorithm for solving the scheduling problem.

(1.) An easy understanding scheduling representation is introduced by using only a four-dimension matrix, which is efficient for designing scheduling problem.

(2.) In our proposed method, the energy function is defined by our assumption. It is more practical than previous studies when the resource consideration is involved.

(3.) We use GA-like solution presentation form, it can be implement easily by a genetic algorithm.

(4.) This algorithm can be adapted to different configurations.

The time complexity is O (N*M*R*T). The simulation results show ours method always can find a desired and feasible solution with an acceptable iterations.

REFERENCES

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

[2] T. M. Willems and J. E. Rooda, "Neural networks for job-shop scheduling," *Contr. Eng. Pract.* vol. 2, no. 1, pp. 31–39, 1994.

[3] Y. P. S. Foo and Y. Takefuji, "Integer linear programming neural networks for job-shop scheduling," in *IEEE Int. Conf. Neural Networks,* 1988, vol. 2, pp. 341–348.

[4] C. -S. Zhang, P. -F. Yan, and T. Chang, "Solving job-shop scheduling problem with priority using neural network," in *IEEE Int. Conf. Neural Networks,* 1991, pp. 1361–1366.

[5] A. Hanada and K. Ohnishi, "Near optimal jobshop scheduling using neural network parallel computing," in *Proc. IECON'93, Int. Conf. Industrial Electronics, Control, Instrumentation,* 1993, vol. 1, pp. 315–320.

[6] 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.

[7] 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, august 1999,pp490-502

[8] 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 Conference*, 1998, pp288 -294

[9] W. Xiao, P. Hao, S. Zhang, and X. Xu ”Hybrid flow shop scheduling using genetic algorithms” in *IEEE proceeding of 3rd world congress on intelligent control and automation,* 2000, pp. 537–541.

[10] 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.

[11] S. Hajri, N.; Liouane,; S.Hammadi,; P. Borne,” A controlled genetic algorithm by fuzzy logic and belief functions for job-shop scheduling,” in *IEEE Transactions on Systems*, Man and Cybernetics, Part B, , Volume: 30 Issue: 5 , Oct. 2000,pp812 –818

[12] A.A.El Dessouky, R.Aggarwal, M.M.Elkateb and F.Li, “Advanced hybrid genetic algorithm for short-term generation scheduling ”, in *IEEE Proc-Gener*. Transm. Distrib, Vol. 148, NO. 6, November 2001, pp. 511-517

[13] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, E. Teller, *Equation*

of State Calculations by Fast Computing Machines, J. of Chem. Phys., Vol. 21, No. 6, pp. 1087-1092, 1953.

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