ELSEVIER

Artificial Intelligence in Engineering 13 (1999) 399–403

ARTIFICIAL INTELLIGENCE in ENGINEERING

www.elsevier.com/locate/aieng

A genetic algorithm for minimizing the makespan in the case of scheduling identical parallel machines

Liu Min*, Wu Cheng

Department of Automation, Tsinghua University, Beijing 100084, People's Republic of China

Received 1 February 1998; received in revised form 15 June 1999; accepted 23 June 1999

Abstract

Identical parallel machine scheduling problem for minimizing the makespan is a very important production scheduling problem, but there have been many difficulties in the course of solving large scale identical parallel machine scheduling problem with too many jobs and machines. Genetic algorithms have shown great advantages in solving the combinatorial optimization problem in view of its characteristic that has high efficiency and that is fit for practical application. In this article, a kind of genetic algorithm based on machine code for minimizing the makespan in identical machine scheduling problem is presented. Several different scale numerical examples demonstrate the genetic algorithm proposed is efficient and fit for larger scale identical parallel machine scheduling problem for minimizing the makespan, the quality of its solution has advantage over heuristic procedure and simulated annealing method.

Keywords: Identical parallel machine; Genetic algorithm; Simulated annealing; Heuristic procedure; Combinatorial optimization; Scheduling problem

1. Introduction

Identical parallel machine scheduling problem for minimizing the makespan has been proved to be a NP problem [1,2]. It is traditionally solved by operational methods such as dynamic programming, branch and bound method, integer programming [3–8] etc. These methods can give an optimal solution for a reasonably sized problem, however, in the case of a large scale problem, 'curse of dimensionality' limits the applications of mathematical optimization techniques. Heuristic procedure [9] is suitable for identical parallel machine scheduling problem of small scale, but in case of processing objects of larger scale, heuristic procedure, for example, Largest Processing Time (LPT) method, is not yet effective enough, especially the accuracy of the solution need improving. Genetic algorithm (GA) [10–15] has been applied in those fields such as combinatorial optimization successfully in view of its characteristic such as near optimization, high speed, and easy realization. And it has also shown great advantages in solving industrial production scheduling problem. In this article, the authors propose a kind of genetic algorithm method based on machine code for the identical parallel machine scheduling problem. The computational results were compared with the LPT and simulated annealing method (SA) [16]. It shows that genetic algorithm method proposed here is suitable for larger scale identical parallel machine scheduling problem for minimizing the makespan, and has advantages of short computation time and high adaptability, the quality of its solution has advantage over LPT and SA.

In Section 2, the problem is mathematically formulated. The detailed optimization algorithm of the problem is explained in Sections 3 and 4, to demonstrate the searching performance, numerical results of different scale problems are shown by comparing with those of GA, LPT and SA.

2. Problem formulation

In this study, the identical parallel machine scheduling problem for minimizing the makespan is defined as follows: there are n independent jobs and m identical machines, each job has its fixed processing time. The processing job can be completed by either of the machines. We want to find the smallest schedule, in which we determine the number of the job processed on each machine to make the whole processing time shortest. First, suppose that $t(j)$ is the processing time of job j. For the sake of convenience for solving the problem by genetic algorithm, in this article we put forward a concept of processing time in a broad sense, that is, we define that $x(i,j)\mathrm{t}(j)$ is the broadly processing time of the job j processed on machine i. Obviously, if $x(i,j) = 1$, it shows

* Corresponding author.

0954-1810/99/$ - see front matter
PII: S0954-1810(99)00021-7

that job j is processed on machine i; if $x(i,j)=0$, it shows that job j is not processed on machine i. Because the objective function is the total time required for the completion of all the jobs, the completion time of machine i has nothing to do with the processing sequence. Therefore we can assume that n jobs are processed on every machine in natural number's increasing sequence just as $1 \rightarrow 2 \rightarrow \cdots \rightarrow n$, and its broadly processing time is $x(i,j)t(j)$. Since one job can only be processed on one machine, the matrix $\mathbf{X}$ composed by $x(i,j)$ has the following natures:

1. all elements are '0' or '1';
2. each row has only one element valued 1;
3. the number of elements valued 1 is n.

The objective function can be expressed with the following function form:

$$\min G = \min\left\{\max\left[\sum_{j=1}^{n} x(1,j)t(j),\ldots,\sum_{j=1}^{n} x(k,j)t(j),\ldots,\sum_{j=1}^{n} x(m,j)t(j)\right]\right\}$$

where

$$G = \max\left[\sum_{j=1}^{n} x(1,j)t(j),\ldots,\sum_{j=1}^{n} x(k,j)t(j),\ldots,\sum_{j=1}^{n} x(m,j)t(j)\right]$$

is the makespan.

3. Algorithm

3.1. Genetic algorithm

3.1.1. Coding

If the above binary-coding is used for making up chromosome directly, the digit of the chromosome chain will reach $n \times m$, in addition, genes are not independent of each other, it is very difficult to get the feasible solution when the chromosomes are crossed. For this reason, this article sets the line number of element valued 1 in each row of matrix $\mathbf{X}$ as genes. By this way, genes are independent with each other, and we can adopt natural number to code them directly. The gene code are $k_1, k_2, \ldots, k_j, \ldots, k_n$, where $k_j \in [1, m]$ and is a repeatable positive integer number. Because the gene code is also the number of the machine on which every job is processed, the method is named as genetic algorithm based on the machine code.

3.1.2. Select initial population

Generate N positive integer coded n-digit chromosome chains randomly, and set them as the initial population.

3.1.3. Calculate $x(i,j)$

$\forall j \in [1,n]$ and $i \in [1,m]$, if $k_j = i$, then $x(i,j) = 1$, else $x(i,j) = 0$.

3.1.4. Reproduce

Because the article's purpose is to get the minimal solution of the problem, in order that genetic algorithm could have better chances for the surviving individuals with relatively higher fitness, the article obtains the fitness function by means of transforming the objective function, i.e. let

$$F(i) = \alpha \exp(-\beta G(i)),$$

where α, β is positive real number and the selection tactics is roulette wheel selection [17]. Assume that $P(i)$ is the selection probability of individual i, then

$$P(i) = \frac{F(i)}{\sum_{k=1}^{n} F(k)}, \qquad i = 1, 2, \ldots, N.$$

Let

$$S(0) = 0,$$

$$S(i) = P(1) + P(2) + \cdots + P(i), \qquad i = 1, 2, \ldots, N.$$

First generate N random real numbers ζ_s which are uniformly distributed between 0 and 1, that is, $\zeta_s \in U(0,1)$, $s = 1, 2, \ldots, N$.

If $S(i-1) < \zeta_s < S(i)$, then make individual i the parents for next generation. The objective function is:

$$G(i) = \max\left[\sum_{j=1}^{n} x(1,j)t(j),\ldots,\sum_{j=1}^{n} x(k,j)t(j),\ldots,\sum_{j=1}^{n} x(m,j)t(j)\right]$$

3.1.5. Crossover

We carry out the crossover with combination of two genes in proper order. Because the coding method used in this paper makes it completely independent of the genes, between each other, if $k_j \in [1,m]$, the crossover method could adopt common two-point crossover method whose advantages are that making message exchange between genes is more abundant and that the best solution can be obtained quickly.

Assuming a crossover between individual A and B, select two positions from chromosome chain randomly, and exchange chromosome between this two positions, then there are two child chromosome chains generated.

Crossover:

$$A_1|A_2|A_3 \times B_1|B_2|B_3 \Rightarrow A_1|B_2|A_3, B_1|A_2|B_3$$

where signal $|$ represents the position of crossover point selected randomly, and signal $\times$ represents crossover operation.

Table 1
Processing time of every job on machine

Job j	Processing time $t(j)$
1	6
2	6
3	4
4	4
5	4
6	3
7	3

3.1.6. Mutation

Because the coding method used in this article allows different genes to take the same positive integer value, in order to improve the efficiency of the mutation, this paper puts forward a new mutation method.

First generate a one-digit positive integer, $k_j \in [1, m]$ randomly, then replace the old one when mutating. If k_j is equal to the old one, then select a new positive integer again until they are different, the efficiency of the mutation could be improved greatly using the method.

3.2. Simulated annealing method

Simulated annealing is a kind of search algorithm based on Monte Carlo interactive method, it introduces annealing equilibrium problem of thermodynamics into problem solving. Its basic thought is to apply randomness of the algorithm and to increase freedom of optimization of the algorithm and to accept bad solution with a certain probability, thus escaping from local optimum and tending to global optimum.

The solution of the simulated annealing is expressed in the following form based on machine code $k_1, k_2, \ldots, k_j, \ldots, k_n$ such as the above genetic algorithm, where $k_j \in [1, m]$, and is a positive integer.

In order to improve the efficiency of the simulated annealing, we adopt the method that performs two operators 'swap and inverse' randomly in generating new solution.

Swap operator

Old solution	New solution
2451324532	2351324542

(swap digit 2 and digit 9)

Inverse operator

Old solution	New solution
2451324532	2442315532

(inverse from digit 3 to digit 7)

Where the underlined digit is the genes which need transforming.

Table 2
Result of calculation

Problem scale	LPT	SA	GA
7×3	11	10	10

Simulated annealing procedure of the algorithm is as follows:

Set the initial value of the control parameter $T1$, attenuation function α and Markov chain length L
while $N_{\text{gen}} < L$ do
begin
for $i := 1$ to L do
begin
generate random integer r

($r = 0$ or $r = 1$)

if $r = 0$, then generate y

(by means of inverse)

if $r = 1$, then generate y

(by means of swap)

if $T_{\text{ms}}(y) - T_{\text{ms}}(x) < 0$, then $x := y$
else
if $\exp[-(T_{\text{ms}}(y) - T_{\text{ms}}(x))/T1] >$

$random(0, 1)$, then $x := y$

end
$T1 = \alpha T1$
end
$N_{\text{gen}} = N_{\text{gen}} + 1$
end

In this article, maximum reproduction generation N_{genmax} set in advance is used to act as stopping criterion.

4. Numerical results

Three examples will be used to show that the algorithm presented in this article is effective, we have compared the different results by GA, LPT and SA.

Example 1. A small scale identical parallel machine scheduling problem of 7 jobs and 3 machines.

The processing time of each job is listed in Table 1.
The result of Example 1 is shown in Table 2.
The result shows that the best solution for this problem can be obtained both by GA and by SA, but it can not be obtained by LPT. (The best solution for the problem is 10.)

Example 2. A small scale identical parallel machine scheduling problem of 10 job and 2 machines.

The processing time of each job is listed in Table 3.
The result of Example 2 is shown in Table 4.
The result shows that when problem scale is fairly small, the average best solution by running ten times GA is the same as the one by running each GA. The best result can be obtained within several generations. The selection of

Table 3
Processing time of every job on machine

Job j	Processing time $t(j)$
1	3
2	2
3	6
4	4
5	5
6	7
7	8
8	6
9	2
10	6

Table 4
Result of calculation

Problem scale	SA	GA
10×2	25	25

Table 5
Processing time of every job on machine

Job j	Processing time $t(j)$
1	3
2	2
3	6
4	4
5	5
6	7
7	9
8	13
9	4
10	12
11	10
12	8
13	22
14	11
15	8
16	26
17	14
18	6
19	17
20	27
21	11
22	17
23	26
24	16
25	7
26	23
27	15
28	18
29	15
30	13

crossover probability P_c and mutation probability P_m has little influence on the quality of the solution, the best solution can also be obtained by SA.

Example 3. A larger scale identical parallel machine scheduling problem of 30 jobs and 10 machines. The processing time of each job is listed in Table 5.

Table 6
Result of calculation

Problem scale		30×10
T_{avg}	GA	42.3
	SA	46
T_{best}	GA	41
	SA	44

Table 7
Optimal policy of scheduling

Number of job	Number of used machine
1	8
2	10
3	3
4	2
5	1
6	4
7	10
8	6
9	9
10	10
11	10
12	1
13	2
14	7
15	3
16	6
17	2
18	4
19	5
20	4
21	9
22	1
23	9
24	5
25	10
26	3
27	7
28	8
29	7
30	8

The relevant parameters of GA in this article are as follows:

Individual number per generation: $N = 20$,
Max. generations: $N_{max} = 77$,
Crossover probability $p_c = 0.92$,
Mutation probability $p_m = 0.05$.

The result shows that the satisfied solution can still be obtained by GA when problem scale become larger, and that the quality of both the best solution and the average solution has advantage over that by SA and has something to do with the selection of p_c and p_m. Others, SA has some weak points

such as long running time and difficulty in selecting cooling parameter when the problem scale becomes larger.

The results of Example 3 are shown in Tables 6 and 7.

5. Conclusion

Since genetic algorithm is used for the TSP problem successfully, it has shown great search advantages in solving NP problem in the scope of combination optimization. Because it is realized easily, and has no demand for differentiability and convex of objective function, its adaptability is extraordinary wide, it is significant to enlarge the application area for genetic algorithm further. The paper makes the study on the application of genetic algorithm in solving identical parallel machine scheduling problem for minimizing the makespan. Results of examples show that the algorithm based on machine code presented in this article is fit for larger scale scheduling problem. Further studies will be focused on selecting crossover probability p_c and mutation probability p_m automatically.

Acknowledgements

This research work was funded by National 863 High-Tech Program (No.863-511-9600-008) and Doctor Station Foundation of the National Education Committee (No. 9500322).

References

[1] Sethi R. On the complexity of mean flow time scheduling. Mathematics of Operations Research 1977;2(4):320–30.
[2] Garey MR, Johnson DS. Computer and intractability: a guide to the theory of NP-completeness, San Francisco: W H Freeman, 1979.
[3] Potts CN. Analysis of a linear programming heuristic for scheduling unrelated parallel machines. Discrete Applied Mathematics 1983;10(2):155–64.
[4] Bernstein D, Pinter RY, Rodeh M. Optimal scheduling of arithmetic operations in parallel with memory, The Annual ACM Symposium on Principles of Programming Languages, New York, 1985.
[5] Luh PB, Hoitomt DJ, Max E. Parallel machine scheduling using Lagrangian relaxation. IEEE International Conference on Computer Integrated Manufacturing, New York, 1988. pp. 244–8.
[6] Narahari Y, Srigopal R. Real-world extensions to scheduling algorithms based on Lagrangian relaxation. Proceedings in Engineering Sciences 21st, 1996. pp. 415–33.
[7] Cheng TCE, Sin CCS. State-of-the-art review of parallel-machine scheduling research. European Journal of Operational Research 1990;47(3):271–92.
[8] Blazewicz J, Dror M, Weglarz J. Mathematical programming formulations for machine scheduling, a survey. European Journal of Operational Research 1991;51(3):283–300.
[9] Blackstone HG, Phillips ET, Hogg GL. The state-of-art survey of dispatching rules for manufacturing job shop operations. International Journal of Production Research 1982;20(1):27–45.
[10] Michalewicz Z. Genetic algorithms + data structure = evolution programs, Berlin: Springer, 1992.
[11] Lae J, Park CH. Application of genetic algorithm to job shop scheduling problems with active schedule constructive crossover. Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, 1995. pp. 530–5.
[12] Gilkinson JC, Rabelo LC, Bush BO. Real-world scheduling problem using genetic algorithms. Computers & Industrial Engineering 1995;29(1–4):177–81.
[13] Biegel JE, Davern JJ. Genetic algorithms and job shop scheduling. Computers & Industrial Engineering 1990;19(1–4):81–91.
[14] Wang PC, Korfhage W. Process scheduling using genetic algorithms. Proceedings of IEEE Symposium on Parallel and Distributed Processing, 1995. pp. 638–41.
[15] Sannomiya N, Iima H. Application of genetic algorithm to scheduling problems in manufacturing processes. Proceedings of the IEEE Conference on Evolutionary Computation, 1996. pp. 523–8.
[16] Davis L, editor. Genetic algorithms and simulated annealing Los Altas, CA: Morgan Kaufmann, 1987.
[17] Brindle A. Genetic algorithms for function optimization: Doctoral dissertation, University of Alberta, 1981.