Indian Journal of Fundamental and Applied Life Sciences ISSN: 2231-6345 (Online)
An Online International Journal Available at http://www.cibtech.org/jls.htm
2011 Vol. 1 (4) October-December, pp.386-390/Mohammad Akhshabi

Research Article

USING A PARALLEL GENETIC ALGORITHM FOR MINIMIZING THE MAXIMUM LATENESS ON UNIFORM PARALLEL MACHINES

***Mohammad Akhshabi**
Department of Industrial Engineering, Tonekabon Branch, Islamic Azad University, Tonekabon, Iran
**Author for Correspondence*

ABSTRACT
This paper considers the uniform parallel machine scheduling problem which is to minimize the maximum lateness. This problem is equivalent to the uniform parallel machine scheduling problem, which is to minimize the maximal completion time of n jobs whose release times are zero, processing times depend on the speed of the machine to which they are assigned, and their delivery times are different. This problem is NP-hard, even if the machines' speeds are identical and all the delivery times equal to zero. In this paper, a parallel GA is employed, an experimental study has been carried out to evaluate our methods.

Key Words: *Parallel Machine Scheduling, Maximum Lateness, Parallel GA*

INTRODUCTION
We consider the uniform parallel machine scheduling problem to minimize the maximal lateness. It is described generally as Qm||Lmax according to the standard three parameter notation proposed by Graham *et al.(1979)*. Each scheduling problem is denoted by the standard three-field notation $\alpha|\beta|\gamma$. The first field α describes the scheduling type, the second field β is reserved for the information and conditions of scheduling, while the third field γ contains the performance criteria. This problem is NP-hard, even if its simpler P2||Cmax case, in which there are only 2 machines and all of the jobs have the same due date (1977). This problem is equivalent to the uniform parallel machine problem with delivery times to minimize the maximal completion time, i.e. Qm|qj|Cmax problem. In many practical scheduling problems, the delivery times must elapse after the jobs are processed on the machine. The delivery times may be brought by the transportation times or the need of some special products, such as the course of cooling for steel and iron products or the course of drying for painted products. The uniform parallel machine scheduling problems are fundamental to numerous complex real-world applications.

Although much research has been devoted to the parallel machine scheduling problems with identical machine speeds, little research has been done on uniform parallel machines. Koulamas and Kyparisis (2000) showed that an extension of the EDD (Earliest Due Date first) rule to the Qm||Lmax problem yields a maximum lateness value does not exceed the optimal value by more than pmax, where pmax is the maximum job processing time. They also showed that the LDT (Largest Delivery Time first) heuristic is a $(m-1)s_1/\sum_{i=1}^{m} s_i$ +1-approximation algorithm for Qm|qj|Cmax problem, where m is the number of the machines, si is the speed of the i-th machine, and s1 is the maximum speed. Dessouky (1998) considered Qm|rj, pj = 1|Lmax problem, in which the job processing times are identical. He proposed six simple heuristics, and then developed a branch-and- bound procedure which could solve the problems within 5 machines and 80 jobs within a reasonable time.

Some other papers considered the corresponding identical parallel machine scheduling problem in which the machines have the same speed. The literature in recent years mainly focused on the problem with unequal release dates, i.e. Pm|rj|Lmax problem. Carlier (1987)and Néron *et al.*, (2008) developed some exact branch-and-bound algorithms for the problem. Vakhania (2004) and Gharbi and Haouari (2007) considered the development of heuristics. Mastrolilli (2003) and Carlier and Pinson (1998) proposed some approximation algorithms. Eren (2009) considered the m-identical parallel machine scheduling problem with a learning effect to minimize the maximum lateness. He proposed a model, which can optimally solve the problems with 18 jobs and 4 machines within 7000 s on a personal computer with Pentium IV/2 512 Ram. For the single machine with minimizing the maximum lateness,

recent literature mainly considered some extensive problems. For example, Wu *et al.*, (2007) considered the single-machine maximum lateness minimization problem with a learning effect. The simulated annealing algorithm they proposed outperforms the traditional heuristic algorithm in terms of quality and execution time for a large number of jobs. It can solve the problems with 200 jobs within 21.24 s on a Pentium IV personal computer. Uzsoy and Velasquez (2008) addressed the problem of scheduling a single machine subject to family dependent set-up times in order to minimize maximum lateness. The incomplete dynamic programming heuristic they developed can solve the problems with 50 jobs within 334.21 s on a Pentium II, 266 MHz notebook with 64 MB of RAM.

Considering the NP-hardness of our scheduling problem, we introduce a hybrid genetic algorithm approach to generate the near-optimal solutions with high quality.Only a few papers have focused on parallel machine scheduling problems.

The remainder of this paper is organized as follows. In the next section, we describe Qm||Lmax problem and Qm|qj|Cmax problem. In the third section, we consider Variable proposed parallel genetic algorithm. The results are presented in final Section.

Problem description

The problem under consideration is the problem of scheduling uniform parallel machines so as to minimize the maximal lateness. We are given a set of n jobs, j_1, …. , j_n, each of them has to be scheduled without interruption on one of m machines, M_1,…..,M_m. Machine M_i (i = 1, . . . ,m) has a speed s_i (s_i > 0). Without loss of generality, we assume that $s_1 \geq s_2 \geq \ldots\ldots\ldots \geq s_m$. A machine can process at most one job at a time, and a job can run on only one machine at a time. All jobs and machines are available at time 0. If a job J_j is processed on a machine Mi, it will take a positive processing time p_{ij} and $p_{ij} = p_j/s_i$, here p_j is the length of job J_j. Each job has a distinct due date d_j for j = 1,… ,n. The objective is to determine a schedule so that the maximum lateness L_{max} = max L_j $1 \leq j \leq n$ is minimized, where $L_j = c_j - d_j$ is the lateness and c_j is the completion time of job J_j. Because the objective function L_{max} is not always positive in Qm||Lmax problems, the equivalent form Qm|qj|Cmax is frequently considered in the literature. Let $q_j = d_{max} - d_j$for all j and d_{max} = max d_j , $1 \leq j \leq n$ is the maximum due date (1977).

In this paper, we mainly focus on the Qm|qj|Cmax problem too. Here each job J_j has a distinct positive delivery time q_j that must elapse between its completion on the machine and its exit from the system. Let $c_j$denote the completion time of job J_j on a machine, then for the consumers the effective completion time $C_j = c_j + q_j$. The objective is to minimize the largest Completion time c_{max}= max c_j = max (c_j + q_j). Note that the makespan corresponds to c_{max}= max c_j, and it is different from the maximum completion time here.denote that the job J_j is processed on the machine M_i.c_{max} ($J_{j'} \rightarrow M_{i'}$, $J_j \rightarrow M_i$) is the larger completion time between job $J_{j'}$ and $J_{j'}$, when they are all scheduled on machine M_i, and the job $J_{j'}$ is processed before job J_j. ''c_{max} ($J_{j'} \rightarrow M_{i'}$, $J_j \rightarrow M_i$) '' denotes the larger completion time between job $J_{j'}$ and J_j, when they are proposed on the machine $M_{i'}$ and $M_i$respectively. The sequence of the jobs on the same machine forms a sub-schedule.

The Proposed Parallel Genetic Algorithm

The proposed parallel genetic algorithm involves a master scheduler, which has the processor lists and the job queue. The processors of the distributed system are heterogeneous. The available network resources between processors in the distributed system can vary over time.

The availability of each processor can vary over time (processors are not dedicated can may have other jobs that partially use their resources). Jobs are indivisible, independent of all other jobs, arrive randomly, and can be processed by any processor in the distributed system. The master scheduler runs a sequential GA in which the fitness function evaluation alone is done by slave processors. When jobs arrive they are placed in the unscheduled job queue. They jobs are taken in batches and scheduled. Batch schedulers are shown to have higher performance than immediate schedulers in. When any processor is idle, the processor asks for a job to perform and the job scheduled for that processor (if any) is given to that

processor. All the job data are maintained only in the Synchronous master slave parallelization is used to evaluate the fitness function alone in a distributed fashion. These are the steps in parallelization,

a) A master scheduler which is the processor in charge of scheduling chooses the slaves. This choice is based upon the communication overhead involved and the computational potential of the slave processor. In other words, a processor which is too slow or too remote will not be used as a slave.

b) The master has the population of chromosomes for which the fitness function is to be evaluated.

c) Each slave evaluates the fitness of a fraction (F_i) of the population in the master scheduler.

d) After partitioning the population into fractions, the slave processors receive their fraction of

Chromosomes one at a time evaluate and return the result to the master. This approach is efficient because, it limits the data transfer. In a distributed environment, the slaves may leave the system at any time. So the chromosomes are transferred only just before the calculation is to be performed. The above algorithm has exactly the same properties as a sequential GA, but executes faster. The Pseudo code for the underlying sequential genetic algorithm is shown in below.

Encode the chromosome.
Initialize the population (randomize)
do {Stochastic sampling with partial replacement selection
Cycle crossover
Mutation: randomize and rebalancing}
While (stopping conditions not met)
Return best individual

Encoding the Chromosome

Each job in the batch has a unique identification number. The total number of jobs in the batch is N and total number of processors in M. The unique identification job number of all the jobs allocated to a processor is encoded in the chromosome with -1 being used to delimit the different processor queues.

Table 1: A sample chromosome

5	1	-1	2	-1	3	4

The sample chromosome in Table 1 has a batch size of 5 jobs with 3 processors. This chromosome represents the following job allocation.

Table 2: Job Allocation

Processors	Jobs
1	5,1
2	2
3	3,4

Fitness Function

A fitness function computes a single positive integer to represent how good the schedule is. We use relative error to generate the fitness values. The fitness of each individual in the population is calculated using synchronous master slave parallelization, in other words, by this function itself is computed by the slave processors. Previously assigned, but unprocessed, load for each processor is considered by calculating the finishing time of a processor j.

Cycle Crossover

Cycle crossover is a crossover operator which applies to permutation encoding schemes which need to preserve both the allele value and the allele order of the gene. This operator ensures that, the two offspring will have their gene values taken from the same value and position of either of their parents.

This ensures that the properties of the parents are carried over to the children there by making fitter children possible in table 3 and 4.

Table 3: Parents

3	4	-1	2	-1	5	1
1	-1	3	5	-1	2	4

Table 4: Children

3	4	-1	5	-1	2	1
1	-1	3	2	-1	5	4

The above example uses the randomized locus for start of the start of the cycle as the first position. The cycle formed is 3-1-4-3.The 5 and 2 of the parents, which are not part of the cycle, are swapped to get the resulting children.

Swapping Mutation

An individual in the population is randomly selected and any two jobs in that chromosome are randomly selected and swapped. This approach ensures that all the solutions in the search space are more thoroughly examined.

Stopping Conditions

A maximum of 1000 evolutions are used. The fitness values of the chromosomes obtained after 1000 evolutions did not show considerable improvement. The GA will also stop evolving if one of the processors becomes idle, in which case it will return the best schedule found so far.

RESULTS

To illustrate the effectiveness and performance of the proposed parallel genetic algorithm (PPGA), it is implemented in MATLAB 7 on a laptop with Pentium IV Core 2 Duo 2.53 GHz CPU. The outputs of the PPGA are compared with that achieved by Lingo 8 software. To study the function of maximum lateness, some example questions are randomly created, and the related results are reported in terms of the RDI in Table 1. The relative deviation index (RDI) is used for the given problem as a common performance measure to compare the instances. Then the results obtained from the proposed parallel genetic algorithm are compared with the calculation of the question by GA and are analyzed.

Table 1: Comparison between GA and PPGA

Problem	**n×m**	**GA**		**PPGA**	
		RDI	**CPU time**	**RDI**	**CPU time**
1	10×5	0.16	1	0.04	0.13
2	10×10	0.18	1	0.04	0.28
3	10×20	0.19	1	0.05	0.29
4	20×5	0.36	2	0.18	0.46
5	20×10	0.38	2	0.21	0.49
6	20×20	0.44	3	0.23	0.55
7	50×5	0.33	4.23	0.28	0.68
8	50×10	0.36	5.58	0.32	0.69
9	50×20	0.37	6.12	0.38	0.75
10	100×5	0.38	8.25	0.08	0.79
11	100×10	0.48	8.85	0.04	0.86
12	100×20	0.62	9.23	0.12	1

In Table 1 the results obtained from the GA and PPGA calculation with various sizes that are determined by n and m, where n =10, 20, 50, 100and m =5, 10, 20. For each combination of n and m, 10

instances are randomly generated and then the relative deviation index (RDI) is used for the makespan of the given problem as a common performance measure to compare the instances. The result shows that PPGA has better performance compared to the GA

Conclusions
The experiments prove that the PPGA can be used to solve minimizing the maximum lateness on uniform parallel machines effectively and the efficiency of PPGA has been perfectly shown by the expansion. In this paper we propose a parallel genetic algorithm to solve the minimizing the maximum lateness on uniform parallel machines effectively Problem with regard to being NP-hard, the method of parallel genetic algorithm with the use of MATLAB 7.0 software has been developed and then the quality of the results with their time of calculation is compared with the results obtained from GA and For other state of production such as parallel machine series machine more researchers could done for future works. Other Meta heuristic methods like Memetic algorithm, SA algorithm PSO algorithm could be used as well.

ACKNOWLEDGEMENTS
The author gratefully acknowledges the support provided by the Islamic Azad University Tonekabon Branch for their kind support and cooperation during researched plan that this paper resulted from it.

REFERENCES
Carlier J (1987). Scheduling jobs with release dates and tails on identical parallel machines to minimize the makespan, European Journal of Operational Research **29** 298–306.
Carlier J, Pinson E (1998). Jackson's pseudo preemptive schedule for the P|ri, qi|Cmax, Annals of Operations Research **83** 41–58.
Dessouky MM (1998). Scheduling identical jobs with unequal ready times on uniform parallel machines to minimize the maximum lateness, Computers & Industrial Engineering **34** (4) 793–806.
Eren T (2009). A note on minimizing maximum lateness in an m-machine scheduling problem with a learning effect, Applied Mathematics and Computation **209** 186–190.
Gharbi A and Haouari M (2007). An approximate decomposition algorithm for scheduling on parallel machines with heads and tails, Computers & Operations Research **34** 868–883.
Graham RL, Lawler EL, Lenstra JK and Rinnooy Kan AHG (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey, Annals of Discrete Mathematics **5** 287–326.
Koulamas C and Kyparisis GJ (2000). Scheduling on uniform parallel machines to minimize maximum lateness, Operations Research Letters **26** 175–179.
Lenstra JK (1977). Sequencing by enumerative methods, in: Mathematical Centre Tracts, vol. 69, Centre for Mathematics and Computer Science, Amsterdam,
Lenstra JK, Rinnooy Kan and AHG Brucker P (1977). Complexity of machine scheduling problems, Annals of Discrete Mathematics **1** 343–362.
Mastrolilli M., (2003). Efficient approximation schemes for scheduling problems with release dates and delivery times, Journal of Scheduling **6** 521–531.
Néron E, Tercinet F and Sourd F (2008). Search tree based approaches for parallel machine scheduling, Computers & Operations Research **35** (4) 1127–1137.
Uzsoy R and Velasquez JD (2008). Heuristics for minimizing maximum lateness on a single machine with family-dependent set-up times, Computers & Operations Research **35** 2018–2033.
Vakhania N (2004). Single-machine scheduling with release times and tails. Annals of Operations Research **129** 253–271.
Wu CC, Lee WC and Chen T (2007). Heuristic algorithms for solving the maximum lateness scheduling problem with learning considerations, Computers & Industrial Engineering **52** 124–132.