Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Technology 1 (2012) 351 – 355

Procedia Technology

Solving flow shop scheduling problem using a parallel genetic algorithm

Mostafa Akhshabi [a] *, Javad Haddadnia[b] , Mohammad Akhshabi [a]

[a] *Electrical Engineering Department, SabzevarTarbiat Moallem University, Sabzevar, Iran*
[b] *Young Researchers Club, Neyshabur Branch, Islamic Azad University, Neyshabur,Iran, mostafa.akhshabi@jau-neyshabur.ac.ir*
m.akhshabi@toniau.ac.ir

Abstract

The effort of searching an optimal solution for scheduling problems is important for real-world industrial applications especially for mission-time critical systems. In this paper, a parallel GA is employed to solve flow shop scheduling problems to minimize the makespan.According to our experimental results, the proposed parallel genetic algorithm (PPGA) considerably decreases the CPU time without adversely affecting the makespan.

Keywords: flow shop scheduling problems, Parallel, genetic algorithm

1. Introduction

Scheduling and assignment problems are almost always parallelized by multiple-threads methods, such as parallel tabu search of Taillard for the QAP [1] and for the job shop problem [2], parallel scatter-search of James et al. [3] for the QAP and parallel genetic algorithm of Bozejko and Wodecki [4] for the single machine total weighted tardiness scheduling problem.

These implementations do not parallelize the cost function _ the most expensive element of computations _ but execute multiple working, cooperative or independent, metaheuristic threads

(maybe except a parallel genetic algorithm where a process of population evaluation can be also parallelized). The method of cost function computing in parallel is strongly connected with

this problem; Steinhöfel et al. [5] propose a method, based on a parallel Floyd_Warshall algorithm, of parallel cost function computing for the job shop problem.

Bozejko[6] propose a methods of parallelizing the criterion function calculations for a single solution and a group of concentrated solutions (local neighborhood) dedicated to being used in metaheuristic approaches. Also a parallel scatter-search metaheuristic is proposed as a multiple-thread approach for the flow shop problem. Borgulya [7] present an evolutionary algorithm (EA) for the no-wait flow shop scheduling problem. This is a new island model with special master-slave structure.

Genuine methods of parallel cost function computing for a single solution and a group of local solutions (neighborhood) are presented in a paper for the flow shop problem with C_{max} and C_{sum} criteria. These methods concern dissimilar techniques of a parallel algorithm's projecting process as well as different necessities of modern algorithms of discreet optimization (analysis of one solution, analysis of a local neighborhood). Efficiency, cost and computation speedup depending on type of the problem, its size and the environment of the parallel system used are considered especially in this part of the paper. Theoretical estimations of properties are derived for particular algorithms, and comparative analysis of the advantages resulting from applications of different approaches has been conducted.

doi:10.1016/j.protcy.2012.02.073

2. Flow shop scheduling problem

The flow shop scheduling problem consists of scheduling of given n jobs with same order and given processing time on m machines. The flow shop scheduling problem has a main assumption, i.e. n jobs process on m machines in the same order [8].We use the following notations:

t(i, j)	processing time for job i on machine j (i = 1, 2,. . . ,n), (j = 1, 2,. . . ,m).
n	total number of jobs to be scheduled.
m	total number of machines in the process.
$\{\pi_1, \pi_2, \ldots, \pi_n\}$	permutation job set.

The makespan, criteria considered in this paper can be formulated as follows:

Completion times $C(\pi_i, j)$:

$C(\pi_1, 1) = t(\pi_1, 1)$

$C(\pi_i, 1) = C(\pi_{i-1}, 1) + t(\pi_i, 1)$ $\quad i = 2,. . . ,n$

$C(\pi_1, j) = C(\pi_1, j\text{-}1) + t(\pi_1, j)$ $\quad j = 2,. . . ,m$

$C(\pi_i, j) = \max\{C(\pi_{i-1}, j), C(\pi_i, j\text{-}1)\} + t(\pi_i, j)\}$ $\quad i = 2,. . . ,n;\ j = 2,. . . ,m$

And Makespan, calculated as $C(\pi_n, m)$

3. 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 Figure 1.

```
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
```

Figure 1. Pseudo code for genetic algorithm

3.1. 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.

5	1	-1	2	-1	3	4

Figure 2: A sample chromosome

The sample chromosome in Figure 2 has a batch size of 5 jobs with 3 processors. This chromosome represents the following job allocation.

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

Table 1: Job Allocation

3.2. 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.

3.3. 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.

Parents

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

3.4. 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.

3.5. 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.

4. Experimental 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 makespan in flow shop problem, some example questions are randomly created, and the related results are reported in terms of the RDI in Table 2. 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.

problem	n×m	GA		PPGA	
		RDI	CPU time	RDI	CPU time
1	10×5	0.15	1	0.04	0.33
2	10×10	0.18	1	0.04	0.28
3	10×20	0.18	1	0.04	0.26
4	20×5	0.35	2	0.16	0.46
5	20×10	0.38	2	0.19	0.48
6	20×20	0.42	3	0.19	0.52
7	50×5	0.32	4.20	0.21	0.62
8	50×10	0.36	5.55	0.18	0.64
9	50×20	0.32	6.12	0.26	0.69
10	100×5	0.31	8.13	0.08	0.79
11	100×10	0.33	8.76	0.04	0.82
12	100×20	0.32	9.14	0.12	1

Table 2: comparison between GA and PPGA

In table (2) 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

5. Conclusions

The experiments prove that the PPGA can be used to solve JSSP 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 flow shop Scheduling 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.

Reference

[1] E. Taillard, (1991). Robust taboo search for the quadratic assignment problem, Parallel Computing 17 443_455.

[2] E. Taillard, (1994). Parallel taboo search techniques for the job shop scheduling problem, ORSA Journal on Computing 6 108-117.

[3] T. James, C. Rego, F. Glover,(2005). Sequential and parallel path-relinking algorithms for the quadratic assignment problem, IEEE Intelligent Systems 20 (4) 58_65.

[4] W. Bo»ejko, M. Wodecki,(2004). Parallel genetic algorithm for minimizing total weighted completion time, in: Lecture Notes in Computer Science, vol. 3070, Springer, 400_405.

[5] K. Steinhöfel, A. Albrecht, C.K.(2002). Wong, Fast parallel heuristics for the job shop scheduling problem, Computers and Operations Research 29 151_169.
[6] w. Bozejko,(2009). Solving the flow shop problem by parallel programming, Journal of Parallel and Distributed Computing 69 470_481
[7] I. Borgulya,(2010). An Island Model for the No-Wait Flow Shop Scheduling Problem, Springer-Verlag Berlin Heidelberg PPSN XI, Part II, LNCS 6239 280–289.
[8] Baker, K. R. (1974). Introduction to sequencing and scheduling. New York: Wiley.