J. Basic. Appl. Sci. Res., 1(10)1484-1489, 2011

ISSN 2090-424X
Journal of Basic and Applied Scientific Research
www.textroad.com

Parallel Genetic Algorithm to Solving Job Shop Scheduling Problem

Mohammad Akhshabi [1]*, Mostafa Akhshabi [2], Javad Khalatbari [3]

[1]Department of Industrial Engineering, Islamic Azad University Tonekabon Branch, Tonekabon, Iran
[2]Young Researchers Club, Neyshabur Branch, Islamic Azad University, Neyshabur,Iran
[3] Department of Management, Islamic Azad University Ramsar Branch, Ramsar, Iran

ABSTRACT

A parallel genetic algorithm has been proposed to job shop scheduling problem (JSSP). The JSSP is known to be NP complete. Genetic algorithms, a meta-heuristic search technique, have been applied successfully in this field. The suggested algorithm uses multiple processors with centralized control for scheduling. Jobs are taken as batches and are scheduled 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.

KEY WORDS: Genetic algorithms, Parallel Processing, Scheduling, makespan.

INTRODUCTION

Genetic algorithm (GA) is a powerful search technique based on natural biological assessment which is applied for finding an optimal or near-optimal solution. The idea of GA was first suggested by Holland [1] in the early 1970s and since then has been widely used in solving optimization problem. In contrast to other optimization methods, GA functions by generating a large set of possible solutions to a given problem instead of working on a single solution. The technique such as the process of selection, crossover, mutation and evaluation.Has been implemented successfully in many scheduling problems, in particular job shop scheduling. Job shop scheduling problem (JSSP) is a difficult NP-hard combinatorial optimization problem. Earlier work on solving JSSP centered on exact algorithms such as branch-and-bound approach [2, 3]. However, the work focused on small sized instances which can be solved in a reasonable computation time. As the problems become more complex, the research focused various other techniques such as simulated annealing [4–6], and genetic algorithms [7–11], ant colony optimization (ACO) [12,13]. To express the characteristics of the real world problems, Oduguwa et al. [14] did a research on the applications of evolutionary computing which includes GA in manufacturing industry. Generally, GA is embed to minimize the makespan of the scheduling jobs with quick convergence to the optimal solution, hence reducing the computational cost. The complexity of some of the JSSP instances provides the impetus for the search of a faster convergence algorithm and one which can avoid local minima. Parallel GA (PGA) is expressed by the researchers in 1980s and 1990s [15–18] to make the technique faster by executing GAs on parallel computer. PGA can be classified into four types: global parallelization (master-slaver model), coarse-grained, fine-grained and hybrid algorithm. Many researchers employed these types of PGA in solving JSSP with varying degrees of success. For example, Kirley [19] had divided JSSP into sub-problems of lower complexity and used parallel evolution of partial solutions. Later, Zhang and Chen [20] did a research on coarse-grained PGA based grid job scheduling and able to minimize the execution time of jobs and makespan of resources compared to the serial process, hence improve the utilization of resources. This proposed algorithm proved to produce minimal makespan and near-optimal solution. Another paper on JSSP that employed PGA is by Park et al. [21], which proposed an island-model PGA to prevent premature convergence, minimize the makespan and generate better solution than serial GA. A coarse grain PGA had also been embedded by Defersha and Chen [22] to solve a lot streaming problem in a flexible job-shops environment. The proposed algorithm was implemented based on island-model parallelization technique with different connection topologies such as ring, mesh and fully connected. Zhao et al. [23] suggested a combination of solving optimization problem using coarse grain size PGA (CGS-PGA) and master-slaver model PGA (MSM-PGA) with Multi-Agent theory. The algorithm is made up of many M-Agents (master course) and many A-Agent (slaver course). Through the developed algorithm, the speed and precision of calculation had improved significantly besides suppressing the emergence of the early-maturing phenomenon. The advantages of both CGS-PGA and MSM-PGA had been utilized while overcoming their deficiencies. In other related area, Skinner et al. [24] had proposed the

***Corresponding Author:** Mohammad Akhshabi, Department of Industrial Engineering, Islamic Azad University Tonekabon Branch , Tonekabon, Iran, Email: m.akhshabi@toniau.ac.ir

hybridization of two different optimization methods, PGA and Sequential Quadratic Programming (SQP). The simulation experiments show that this method had combined the robust search property of PGA with high convergence velocity of SQP. A set of multimodal, non-linear, smooth objective functions was used to compare PGA with the proposed hybrid algorithm. The experiments prove that the overall computation time was reduced, increase solution quality or accuracy and maintain the robustness. Tang and Lau [25] had investigated on PGA for floor plan area optimization in very large-scale integrated-circuit (VLSI) design automation. This was presented by adopting an island model with an asynchronous migration mechanism, and using different chromosome representations and genetic operators. With asynchronous migration, the efficiency had been improved in contrast to synchronized migration. When compared to the corresponding sequential GA, the PGA had produced better floor planning results. The performance of the PGA was also empirically investigated for different number of islands and migration interval where the PGA is sensitive to both parameters with some correlations between them. Another recent research on PGA is done by Yussof et al. [26] where they suggested a coarse-grained PGA for solving the shortest path routing problem in computer networks. Although the shortest path routing algorithms are already well-established such as Dijkstra's algorithm and Bellman–Ford algorithm, Yussof et al. had attempted to implement PGA as an alternative method to solve this problem which was developed on MPI cluster. The performance of the algorithm in terms of computation time was studied as well as the effect of migration on the proposed algorithm. The results show that the computation time was greatly reduced compared to the serial version of the algorithm. an example is the distributed evolutionary simulated annealing (dESA) [27]. In this work multiple agent (island) systems is applied, where algorithm is evolved in each island and enables each individual to get better chance of being selected across the whole search. The results obtained indicated that the dESA can solve problems far faster compared to non distributed simulated annealing method.

Problem formulation

In general, scheduling problem consists of a set of concurrent conflicting goals that are to be satisfied using a limited number of resources. The JSSP is one of the hardest combinatorial optimization problems, which is a process to allocate limited resources such as machines to competing job. This problem can be described [27] as follows: given a set of n jobs (J) given by J = $\{J_1, J_2, \ldots, J_n\}$, ach composed of more than one operation (O) that must be processed on a set of m machines (M) given by M = $\{M_1, M_2, \ldots, M_m\}$.Each operation occupies one of the machines for a fixed duration. The constraints for the scheduling problems are:

• Each machine can only process one operation at a time.

• Once an operation in iterates processing on a given machine, it must complete the processing on that machine without interruption.

• All operations must be in technological sequence.

The problem's main aim is to schedule the activities or operations of the jobs on the machines such that the total time to finish all jobs,that is the makespan, C_{max} is minimized. The term makespan refers to the cumulative time to complete all the operations of all jobs on all machines. It is a measure of the time period from the starting time of the first operation to the ending time of the last operation. Sometimes there may be multiple solutions that have the minimum makespan, but the goal is to find out any one of them : it is not necessary to find all possible optimum solutions. Let each job j consists of p operations $O_j = \{o_{1j}, o_{2j}, \ldots, o_{pj}\}$,which are interrelated by two types of constraints . First is the precedence constraint, in which the operations of each particular job has to be processed after all predecessor operations, P_i , are completed and before its successor Si commences. Second constraint is the operation i can only be scheduled if the machine it requires is idle. The completion time of the operation, c_{ji}considers the starting time or the release time of the operation r_{ji}, the duration of the operation or processing time needed by job j of operation i one ach machine, denoted by d_{ji}, as well as the waiting time in between operations (if required) w_{ji}. Hence,the completion time of operation o_{ji} is $c_{ji} = r_{ji} + d_{ji} + w_{ji}$.

The completion time of the whole schedule or the makespan is also the maximum of the machines' completion time or the jobs' completion time. Therefore , the makespan is C_{max} = max ($C_1, C_2, \ldots, C_m$) , where $C_1, C_2, \ldots, C_m$ are the completion time of machines 1,2, . . ., m respectively. One of the most important characteristic to differentiate job shop scheduling from other type of scheduling is the order in which a job is processed is taken into account. In JSSP, an order must be specified, as well as defining the amount of time that each job must spend at each machine. This has significantly lowers the size of the search space and makes the representations scheme used in a program potentially harder normal scheduling problem. The problem itself is trying to have every job finished in as little time as possible. This is known as minimizing the makespan.

Parallel Genetic Algorithms (PGAs)

Sequential Genetic algorithms have been used successfully in many different domains. some GAs need to have very large populations which make them impossible to run efficiently on a single machine. Some GAs get trapped in a suboptimal region of search space. The most common problem faced by a sequential GA is the CPU time. In most parallel algorithms, the basic idea behind the algorithm is to divide the task into subtasks and use different processors to execute each subtask. This divide-and-conquer approach can be applied to GAs in many different ways, and the literature contains many examples of successful parallel implementations. A complete classification of PGAs is given in [28]. In master slave parallelization, there is a single panmictic population, but the evaluation of the fitness function is distributed among several processors. Since, selection and crossover consider the entire population they are also called Global parallel GAs. The master always waits for the slowest slave processor before starting the next generation. Asynchronous master slave parallelization is an extension of synchronous master slave parallelization in which the master does not wait for the slowest processor. The selection operator gets affected because of the change and the resulting GA dynamics are difficult to analyze [29].One of the advantages of synchronous master slave parallelization is that the underlying GA characteristics are not affected. This model does not assume anything about the underlying architecture and therefore is most suited in distributed environment. Fine-grained parallel GAs are most suited for massively parallel computers and consists of one spatially-structured population. Selection and mating are restricted to a small neighborhood, but neighborhoods overlap permitting some interaction among all the individuals. The selection and mating operations in these PGAs are different from those of a sequential GA. PGAs with multipopulation are also possible. The important characteristics of multideme PGAs, also called as coarse grained GAs, are a few relatively large subpopulations and migration.

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

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

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.

Figure 2: A sample chromosome

5	1	-1	2	-1	3	4

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

Table 1: 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.

4.5 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.

4.6 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.

4.7 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.

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 PPGA, 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 makespan of 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 2: comparison between GA and PPGA

problem	n×m	GA		PPGA	
		RDI	CPU time	RDI	CPU time
1	10×5	0.10	1	0.02	0.23
2	10×10	0.13	1	0.02	0.23
3	10×20	0.17	1	0.03	0.23
4	20×5	0.41	2	0.12	0.30
5	20×10	0.43	2	0.19	0.30
6	20×20	0.52	2	0.21	0.30
7	50×5	0.32	3.20	0.20	0.38
8	50×10	0.38	3.45	0.18	0.39
9	50×20	0.33	4.09	0.27	0.39
10	100×5	0.25	5.13	0.09	0.89
11	100×10	0.23	5.76	0.08	0.92
12	100×20	0.22	6.08	0.14	1

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.

Conclusions

In this paper we present a parallel genetic algorithm to solve the Job-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 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. 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 field visits and data collection.

REFERENCES

[1] J.H. Holland, Adaptation in Natural and Artificial Systems, the University of Michigan Press, 1975.

[2] D. Applegate, W. Cook, A computational study of job-shop scheduling, ORSA J. Comput. 3 (1991) 149–156.

[3] J. Carlier, E. Pison, An algorithm for solving the job-shop problem,Manage. Sci. 35 (1989) 164–176.

[4] M.E. Aydin, T.C. Fogarthy, A distributed evolutionary simulated annealing algorithm for combinatorial optimisation problems, J. Heuristics 10 (2004) 269–292.

[5] M. Kolonko, Some new results on simulated annealing applied to the job shop scheduling problem, Eur. J. Oper. Res. 113 (1999) 123–136.

[6] T. Satake, K. Morikawa, K. Takahashi, N. Nakamura, Simulated annealing approach for minimizing the makespan of the general job-shop, Int. J. Prod. Econ. 60 (1999) 515–522.

[7] J.F. Goncalves, J.J.M. Mendes, M.G.C. Resende, A hybrid genetic algorithm for the job shop scheduling problem, Eur. J. Oper. Res. 167 (2005) 77–95.

[8] M.F. Hussain, S.B. Joshi, A genetic algorithm for job shop scheduling problems with alternate routing, in: Proceedings of the 1998 IEEE Inter- national Conference on Systems, Man, and Cybernetics, vol. 3, 1998, pp. 2225–2230.

[9] L. Liu, Y. Xi, A hybrid genetic algorithm for job shop scheduling problem to minimize makespan, in: Proceedings of the Sixth World Congress on Intelligent Control and Automation, 2006, pp. 3709–3713.

[10] J.C.H. Pan, H.C. Huang, A hybrid genetic algorithm for no-wait job shop schedul- ing problems, Expert Syst. Appl. 36 (2009) 5800–5806.

[11] L.J. Park, C.H. Park, Genetic algorithm for job shop scheduling problems based on two representational schemes, Electron. Lett. (1995) 2051–2053.

[12] C. Blum,M. Samples, An ant colony optimization algorithm for shop scheduling problems, J. Math. Model. Algorithms 3 (2004) 285–308.

[13] A. Colorni, M. Dorigo, V. Maniezzo, M. Trubian, Ant system for job-shop schedul- ing, Belg. J. Oper. Res. Stat. Comput. Sci. (JORBEL) 34 (1994) 39–53.

[14] V. Oduguwa, A. Tiwari, R. Roy, Evolutionary computing in manufacturing industry: an overview of recent applications, Appl. Soft Comput. 5 (2005) 281–299.

[15] P.B. Grosso, Computer simulations of genetic adaptation: parallel subcompo- nent interaction in amultilocus model, Unpublished Doctoral Dissertation, The University of Michigan, 1985.

[16] R. Tanese, Parallel genetic algorithm for a hypercube, in: Proceedings of the Second International Conference on Genetic Algorithms, 1987, pp. 177–183. [17] R Bianchini, C.M. Brown, Parallel genetic algorithms on distributed-memory architectures Transputer Research and Applications, 6, IOS Press, 1993, 67–82.

[18] J.R. Koza, D. Andre, Parallel genetic programming on a network of transputers, Tech. Rep. No. STAN-CS-TR-95-1542, Stanford University, Stanford, CA, 1995.

[19] M. Kirley, A coevolutionary genetic algorithm for job scheduling problems, in: Proceedings of the 1999 Third International Conference on Knowledge-based Intelligent Information Engineering Systems, 1999, pp. 84–87.

[20] H. Zhang, R. Chen, Research on coarse-grained parallel genetic algorithm based grid job scheduling, in: Proceedings of the Fourth International Conference on Semantics, Knowledge and Grid, 2008, pp. 505–506.

[21] B.J. Park, H.R. Choi, H.S. Kim, A hybrid genetic algorithm for the job shop scheduling problems, Comput. Ind. Eng. 45 (2003) 597–613.

[22] F.M. Defersha, M. Chen, A coarse-grain parallel genetic algorithm for flexible job-shop scheduling with lot streaming, in: Proceedings of 2009 International Conference on Computational Science and Engineering, 2009, pp. 201–208.

[23] T. Zhao, Z. Man, Z. Wan, G. Bi, A CGS-MSM parallel genetic algorithm based on multi-agent, in: Proceedings of the 2nd International Conference on Genetic and Evolutionary Computing, 2008, pp. 10–13.

[24] B.T. Skinner, H.T. Nguyen, D.K. Liu, Hybrid optimisation using PGA and SQP algorithm, in: Proceedings of the 2007 IEEE Symposium on Foundations of Computational Intelligence (FOCI 2007), 2007, pp. 73–80.

[25] M. Tang, R.Y.K. Lau, A parallel genetic algorithm for floorplan area optimiza- tion, in: Proceedings of the 7th International Conference on Intelligent Systems Design and Applications, 2007, pp. 801–806.

[26] S. Yussof, R.A. Razali, O.H. See, A. Abdul Ghapar, M. Md Din, A coarse- grained parallel genetic algorithm with migration for shortest path routing problem, in: Proceedings of the 11th IEEE International Con- ference on High Performance Computing and Communications, 2009, pp. 615–621.

[27]M. Emin Aydin, C. Terence, Fogarty,A distributed evolutionary simulated annealing algorithm for combinatorial optimisation problems, J. Heuristics 10 (2004) 269–292.

[28] E Cantu-Paz ,A survey of parallel genetic algorithms, Calculateurs Paralleles, Reseaux et Systems Repartis, 1998

[29] M Nowostawski, R Poli, Parallel Genetic Algorithm Taxonomy, - 3rd International Conference on Knowledge-Based Intelligent Information Engineering Systems 1999.