Design and Implementation of a Multi-Agent System for the Job Shop Scheduling Problem

Leila Asadzadeh *leila_asadzadeh_cs@yahoo.com*
Information Technology Department
Payame Noor University
Tehran, 19395-4697, I. R. of IRAN

Kamran Zamanifar *zamanifar@eng.ui.ac.ir*
Computer Engineering Department
University of Isfahan
Isfahan, I. R. of IRAN

Abstract

Job shop scheduling is one of the strongly NP-complete combinatorial optimization problems. Developing effective search methods is always an important and valuable work. Meta-heuristic methods such as genetic algorithms are widely applied to find optimal or near-optimal solutions for the job shop scheduling problem. Parallelizing genetic algorithms is one of the best approaches that can be used to enhance the performance of these algorithms. In this paper, we propose an agent-based parallel genetic algorithm for the job shop scheduling problem. In our approach, initial population is created in an agent-based parallel way then an agent-based method is used to parallelize the genetic algorithm. Experimental results showed that the proposed approach enhances the performance.

Keywords: Job Shop Scheduling Problem, Genetic Algorithms, Parallel Genetic Algorithms, Agents and Multi Agent Systems.

1. INTRODUCTION

Job shop scheduling problem is one of the most important problems in machine scheduling. This problem is considered to be a member of a large class of intractable numerical problems known as NP-hard [1]. High complexity of problem makes it hard and in some cases impossible to find the optimal solution within reasonable time. Hence, searching for approximate solutions in polynomial time instead of exact solutions at high cost is preferred for difficult instances of problem.

Historically job shop scheduling problem has been primarily treated using the branch and bound [2-4], heuristic rules [5-7] and shifting bottleneck procedure [8]. In recent years, meta-heuristic methods are widely applied to this problem. These methods, such as taboo search [9-11], simulated annealing [12-15], genetic algorithms [16-20], neural networks [21] and ant colony optimization [22-25] are well suited to solving complex problems with high costs. A survey on job shop scheduling techniques can be found in [1].

Comparing with other meta-heuristic methods, genetic algorithms are widely used to solve various optimization problems. Many genetic algorithm based approaches are proposed for the job shop scheduling problem. In [26,27] authors introduce an approach that uses load balancing of machines as an important parameter in job assignment. An advantage of these approaches is that maximize machine utilization while minimizing makespan. Ombuki and Ventresca [28] proposed a local search genetic algorithm that uses an efficient solution representation strategy in which both checking of the constraints and repair mechanism can be avoided. In their approach at local search phase a new mutation-like operator is used to improve the solution quality. They also developed a hybrid strategy using the genetic algorithm reinforced with a taboo search for problem. Lin et al. [29] introduced a hybrid model consisting of coarse-grain genetic algorithms connected in a fine- grain style topology. Their method can avoid premature convergence, and it produced excellent results on standard

benchmark job shop scheduling problems. Chen et al. [30] gave a tutorial survey of recent works on various hybrid approaches of the genetic algorithms proposed so far for the job shop scheduling problem. Wang and zheng [20] by combining simulated annealing and genetic algorithms developed a general, parallel and easily implemented hybrid optimization framework, and applied it to job shop scheduling problem. Based on effective encoding scheme and some specific optimization operators, some benchmark job shop scheduling problems are well solved by the hybrid optimization strategy. In [19] a hybrid method is proposed to obtain a near-optimal solution within a reasonable amount of time. This method uses a neural network approach to generate initial feasible solutions and then a simulated annealing algorithm to improve the quality and performance of the initial solutions in order to produce the optimal/near-optimal solution. Chen et al [31] proposed an agent-based genetic algorithm that accelerates the creation of initial population. In this approach, the processing of selection, crossover and mutation can be controlled in an intelligent way.

In this paper, we propose an agent-based parallel genetic algorithm for the job shop scheduling problem. In our approach, initial population is created in an agent-based parallel way then an agent-based method is used to parallelize genetic algorithm.

The reminder of this paper is organized as follow. In section 2, we describe job shop scheduling problem. Details of our proposed agent-based architecture and the parallel genetic algorithm are represented in section 3. In section 4, we discuss implementation and experimental results of proposed approach. Conclusion is represented in section 5.

2. JOB SHOP SCHEDULING PROBLEM

Job Shop Scheduling Problem can be described as follow. A set of *n* jobs and a set of *m* machines are given. Each job consists of a sequence of operations that must be processed on a specified order. Each job consists of a chain of operations, each of which needs to be processed during an uninterrupted time period of a given length on a given machine. Each machine can process only one job and each job can be processed by only one machine at a time. Usually we denote the general job shop scheduling problem as *nxm,* where *n* is the number of jobs and *m* is the number of machines. TABLE 1 shows an example 5×4 job shop scheduling problem. The duration in which all operations for all jobs are completed is referred to as the makespan. A schedule determines the execution sequence of all operations for all jobs on machines. The objective is to find optimal schedule. Optimal schedule is the schedule that minimizes makespan. Due to factorial explosion of possible solutions, job shop scheduling problems are considered to be a member of a large class of intractable numerical problems known as NP-hard [1]. It is hard and in some cases impossible to find the optimal solution within reasonable time due to High complexity of problem. Hence, searching for approximate solutions in polynomial time instead of exact solutions at high cost is preferred for difficult instances of problem.

Job	***Machine, Processing time***			
P_1	2,4	1,3	3,11	4,10
P_2	4,10	1,8	2,5	3,4
P_3	1,5	3,6	2,4	4,3
P_4	1,7	2,3	3,2	4,12
P_5	3,5	4,8	1,9	2,5

TABLE 1: An Example *5×4* Job Shop Scheduling Problem

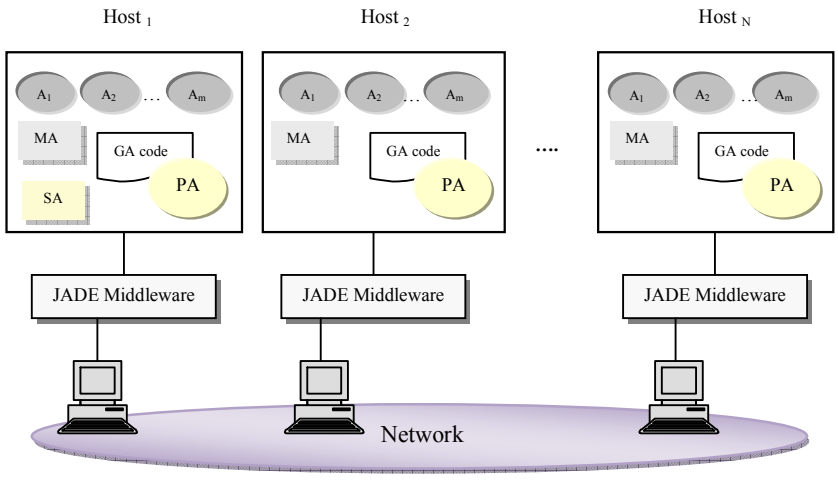

FIGURE 1: Improved Agent-based Architecture for the Job Shop Scheduling Problem

3. AGENT-BASED PARALLEL MODEL

In [32] we proposed an agent-based parallel approach for the job shop scheduling problem. In that model, we developed a multi-agent system containing some agents with special actions that are used to parallelize the genetic algorithm and create its population. We used *JADE* middleware [33] to implement our multi-agent system. Agents distributed over various hosts in network and *JADE* provides a secure communication channel for them to communicate. In this model, each agent has been developed for a special purpose. We can describe them as follow [32]:

- ***MA (Management Agent):*** *MA* and *Ai (i=1,2,...,m)* agents have the responsibility of creating the initial population for the genetic algorithm. This agent controls the behaviors of *Ais* and coordinates them in creation step.
- ***Ai (Execute Agent):*** Each machine has an *Ai* agent to schedule the operations on it.
- ***PA (Processor Agent):*** Each *PA* locates on a distinct host and executes genetic algorithm on its sub-population.
- ***SA (Synchronization Agent):*** This agent locates on main host and coordinates migration between sub-populations of *PA* agents.

In that model, the genetic population is created serially by *MA* and *Ai (i=1,2,...,m)* agents. The sub-populations of *PA* agents are determined and sent to them by *MA*. One disadvantage of this model is the lack of load balancing on the network hosts. On the other hand, the main host that locates the *MA*, *Ai* and *SA* agents is the bottleneck of system and if it crash, the whole multi-agent system will be stopped working.

To solve this problem and improve the performance of creating the initial population, we can extend the model to create sub-populations in a parallel manner. An overall architecture of improved agent-based model has represented in FIGURE 1. In this model, each host has one *MA* and m *Ai (i=1,2,…,m)* agents. These agents have the responsibility of creating the subpopulation for their host's *PA*.

To synchronize the various processor agents in migration phase, synchronization agent *(SA)* locates on main host and synchronizes them.

Parallel creation of sub-populations improves the speed and performance. On the other hand, the division of the population into several sub-populations and sending them to *PAs* can be avoided.

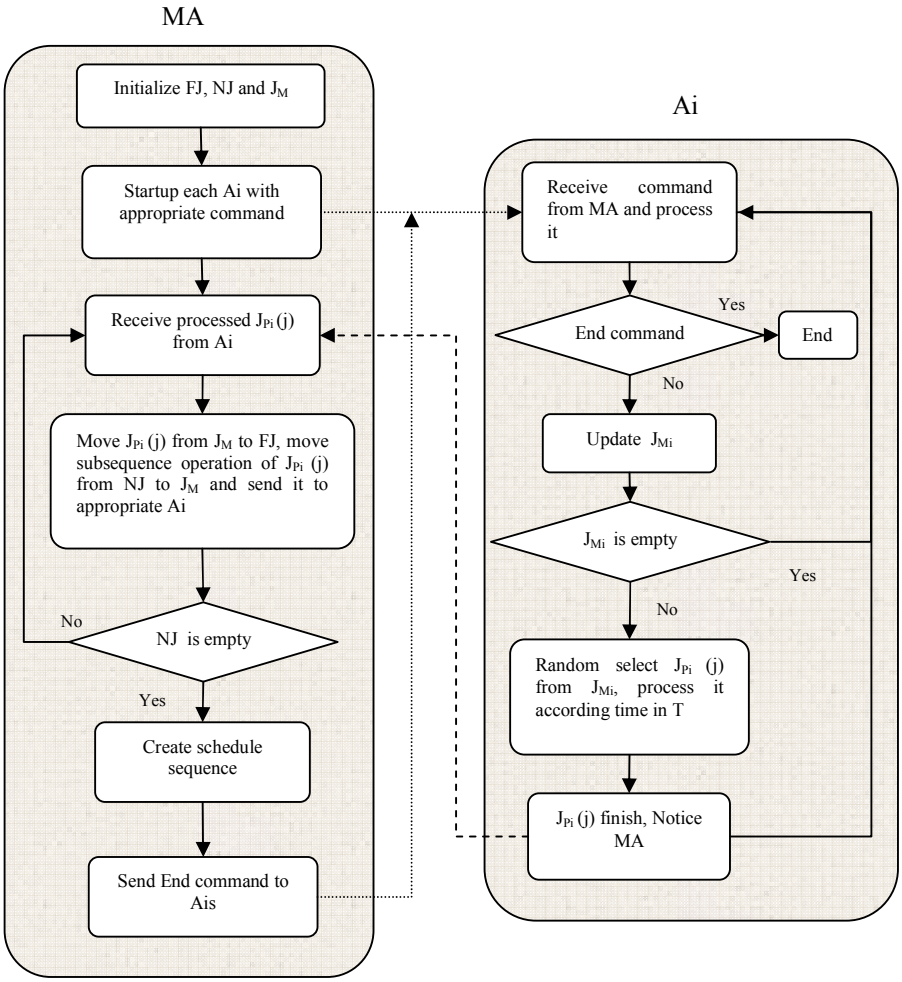

FIGURE 2: Schedule Process of *MA* and *Ai*

3.1 Initial Population of Genetic Algorithm

Before we describe how initial population can be created by an agent-based method we define the job shop scheduling problem formally with the following definitions [31].

1. $P = \{P_1, P_2, \ldots, P_n\}$ is the set of n jobs.
2. $M = \{M_1, M_2, \ldots, M_m\}$ is the set of m machines.
3. Job P_i has k_i operations. $J_{Pi} = \{J_{Pi}(1), J_{Pi}(2), J_{Pi}(k_i)\}$ is the set of operations of job P_i. $J_P = \{J_{P1}, J_{P2}, \ldots, J_{Pn}\}$ is the matrix of operations of n jobs. The value of $J_{Pi}(j)$ is the machine that operation j of job P_i must be processed on.
4. Let T is the $n \times m$ matrix of processing times of each job in m machines. $T[i, j]$ is the processing time of job P_i on machine j.

Status of each operation is determined by using the following definitions.

5. J_{Mi} is the set of all schedulable operations on machine M_i. A schedulable operation is an operation for which all the foregoing operations have finished. $J_M = \{J_{M1}, J_{M2}, \ldots, J_{Mm}\}$ is the set of schedulable operations on all machines.
6. NJ is the set of un-schedulable operations, i.e. operations for which at least one of the foregoing operations has not finished.
7. FJ is the set of finished operations.

Chromosomes of the genetic population are created by using method proposed in [31]. In this method, two kinds of agents are used to create chromosomes of the initial population: the management agent (*MA*) and the execute agent *Ai (i=1,2,…,m)*. Each machine in a specified problem instance has an execute agent. Each *Ai* schedules the operations of its machine. *MA*

manages operations of all jobs and controls the execution of *Ais*. In the first step, *MA* initializes J_M , *FJ* and *NJ* as follow.

$FJ=\phi$

$NJ=\{J_{P1}, J_{P2}, \ldots, J_{Pn}\}$

$J_M =\{ J_{P1}(1), J_{P2}(1)\ldots, J_{Pn}(1) \}$

Parallel genetic algorithm in our approach has multiple populations. The whole population is divided into several subpopulations which are called islands and each island is evolved independently. To improve the speed we produce various sub-populations concurrently. Each host in the system has one *MA* agent and *m Ai (i=1,2,…,m)* agents and these agents produce chromosomes for one of sub-populations. The schedule process in FIGURE 2 is used to create sub-populations. Each execution of this process creates a chromosome indicating a feasible schedule for a specified problem instance. To create a sub-population with size *N* we execute the schedule process *N* times.

3.2 Parallel Genetic Algorithm

To parallelize our genetic algorithm we use a coarse-grained model. This model has multiple and smaller populations and exchanges information among the sub-populations. This exchange is performed by moving some individuals from one population to another and is known as migration. Communication between sub-populations restricted to migration of chromosomes.

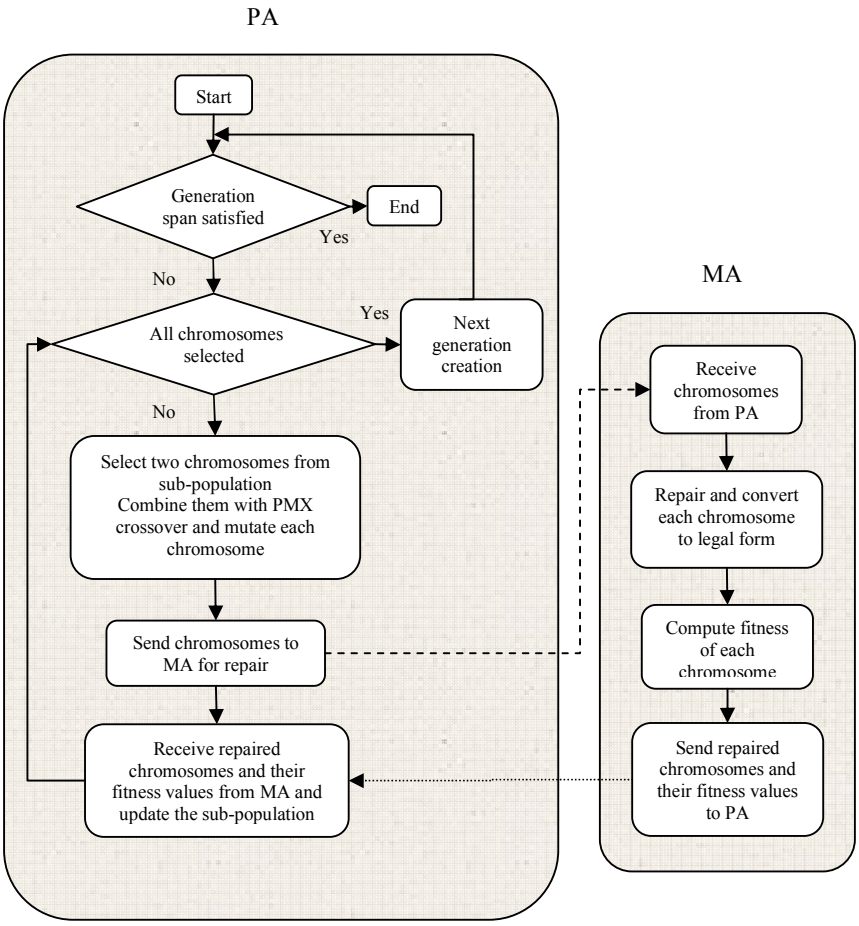

FIGURE 3: *PA* and *MA* Communication in the Execution Phase

In our method, various sub-populations are created by *MA* and *Ai (i=1,2,…,m)* agents in a parallel way. Each processor agent *(PA)* locates on a distinct host and executes genetic algorithm on its sub-population independently. Different sub-populations communicate with exchanging of migrants. Parallel genetic algorithm consists of two phases: The execution phase and the migration phase. In the execution phase, sub-populations are evolved independently by processor agents and in the migration phase, *PAs* exchange migrants. These two phases run repeatedly for predefined times [32]. Detailed communication between *PA* and *MA* in execution phase is showed in FIGURE 3.

3.2.1 Migration Policy

Communication between various sub-populations is carried out by exchanging of migrants. In our approach we use synchronous migration policy. Each *PA* executes genetic algorithm on its sub-population for a predefined number of generations then it sends a message to *SA* informing end of its execution. The *SA* is a synchronization agent, which coordinates migration between sub-populations of *PA* agents. After receiving message from all the *PAs*, *SA* broadcasts a message to them notifying start of the migration phase. In the migration phase, each *PA* exchanges some of its best chromosomes with its neighbors (see FIGURE 4). Chromosomes with low fitness value in sub-population are replaced with the best chromosomes of neighbors [32].

4. IMPLEMENTATION AND EXPERIMENTAL RESULTS

We showed that the parallel agent-based genetic algorithm for the job shop scheduling problem enhances the speed and performance [32]. In this paper we evaluate our improved approach. Firstly, we explain detailed implementation of the genetic algorithm as follow.

a. ***Chromosome representation***: Operation-based method that each job has a distinct number for indicating its operations.
b. ***Selection:*** Roulette wheel selection containing the elite retaining model [34].
c. ***The crossover operator:*** Partially matched crossover *(PMX)* [35], two crossover points is chosen from the chromosomes randomly and equably. Then the genes of two parents that are in the area between crossover points are exchanged.
d. ***The mutation operator:*** Shift mutation, a point from the chromosome is chosen randomly and the gene that is in this point is exchanged with its subsequent gene.
e. ***Fitness function***: The fitness function is defined as follow [34]:

$$Fitness(C) = P\_Time_{max} - P\_Time\ (C) \qquad (1)$$

Where *P_Time (C)* is the maximal processing time of chromosome *C* and P_Time_{max} is the maximum value of *P_Time (C)*. In our approach, *MA* agents compute fitness value of chromosomes.

Creating new chromosomes by using crossover operator may be lead to illegal schedules. Repair mechanism is used by *MAs* to convert these chromosomes to legal form. When a new chromosome is created, *PA* sends it to *MA* agent of its host. *MA* replaces repeat operations of the new chromosome with absent operations to ensure the appearance times of each job P_i is equal to k_i. Repaired chromosome is sent to *PA*.
We used some benchmark instances for the job shop scheduling problem. These problem instances are available from the OR library web site [36]. We set parameter values for genetic algorithm as follow:

- population size =1000
- generation span =1000
- crossover rate = 0.95
- mutation rate = 0.01

The number of *PA* agents was fixed at eight in our experiments and these agents form a virtual cube among them. Each *PA* has three neighbors. Parameters of migration were set as follow:

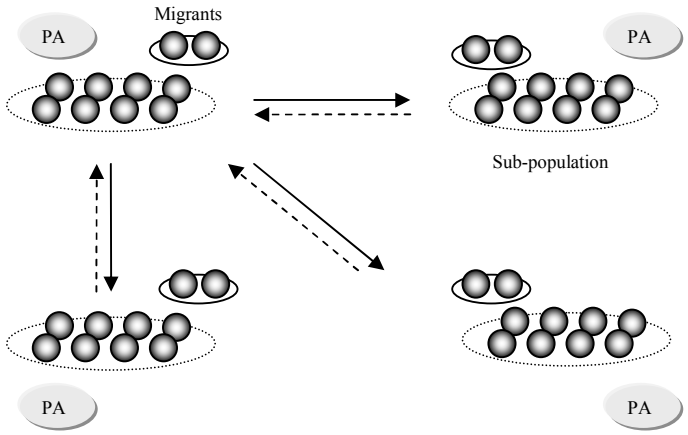

FIGURE 4: Migration Between *PA* Agents

- Migration Frequency : 100 generations
- Migration Rate : 10 chromosomes
- Migration Topology : cube

To evaluate our proposed parallel agent-based genetic algorithm, we compare it with the serial case that we have only one genetic population that evolves by a *PA* agent. We computed average makespan of best schedules obtained by applying algorithms on some problem instances during various generations. Results are shown in FIGUREs 5-7. These figures demonstrate the effect of applying parallel and serial approaches on *LA30*, *ORB09* and *FT10* instances. As shown by these figures, convergence to near optimal solution in parallel method is happen rapidly and the solutions that it finds in various generation numbers have shorter lengths than those that are found by serial method.

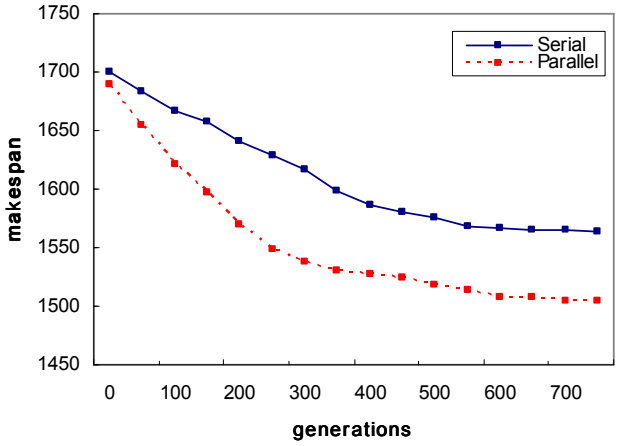

FIGURE 5: Parallel Approach Finds Better Solutions Comparing with Serial Method. Test Problem: *LA30*. Results Are Averages of Best Solutions Over 10 Runs.

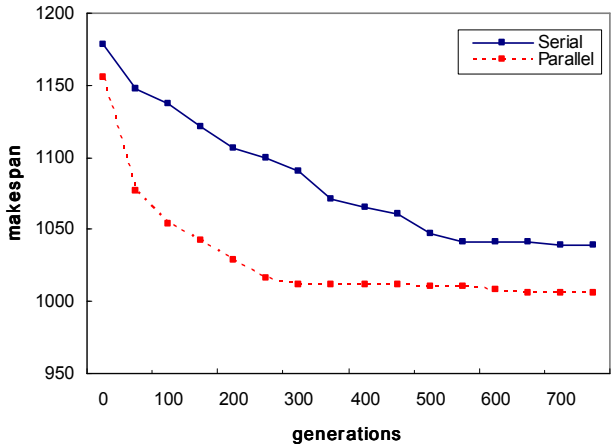

FIGURE 6: Parallel Approach Finds Better Solutions Comparing With Serial Method. Test Problem: *ORB09*. Results Are Averages of Best Solutions Over 10 Runs.

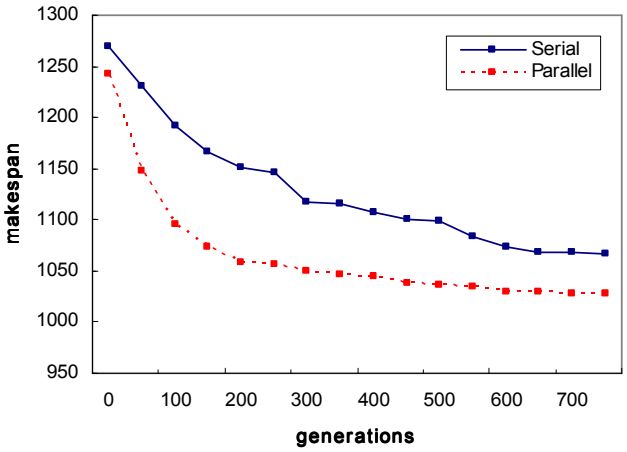

FIGURE 7: Parallel Approach Finds Better Solutions Comparing With Serial Method. Test Problem: *FT10*. Results Are Averages of Best Solutions Over 10 Runs.

To compare the parallel method proposed to create genetic population with the serial approach, we carried out some experiments. The evaluation parameter is the required time for creation of sub-populations for various problem instances. In serial method we have one population with 1000 chromosomes that is divided into 8 sub-populations with 125 chromosomes after that it has been created by *MA* and *Ai (i=1,2,...,m)* agents. In parallel method, 8 sub-populations each with 125 chromosomes are created in parallel manner. Results are shown in TABLE 2. Times are in second. According to experimental results, parallel creation of sub-populations takes less time comparing with serial method.

5. CONCLUSION

Job shop scheduling problem is one of the most important problems in machine scheduling. This problem is considered to be a member of a large class of intractable numerical problems known as NP-hard. In this paper, we proposed an agent-based parallel genetic algorithm for this problem. To enhance the performance of the creation of the initial population for the genetic algorithm, we parallelized it using agent-based method. We compared performance of the parallel approach with the serial method. The results showed that the parallel method improves the speed of genetic population creation. Future work will concentrate on improving the performance of our method and applying it to similar problems.

Problem	Jobs	Machines	Time/s Serial	Time/s Parallel
LA04	10	5	20	6
LA06	15	5	25	6
LA11	20	5	36	5
LA16	10	10	130	9
LA21	15	10	125	15
LA26	20	10	156	17
LA31	30	10	328	25

TABLE 2: Comparisons Between Serial and Parallel Methods for Creating the Genetic Population on *LA* Instances

6. REFERENCES

[1] A. S. Jain and S. Meeran. “Deterministic job-shop scheduling: past, present and future,” Department of Applied Physics and Electronic and Mechanical Engineering, University of Dundee, Dundee, Scotland, UK, 1998.

[2] J. Carlier and E. Pinson. “An algorithm for solving the job shop problem.” *Management Science*, vol. 35, no. 29, pp. 164-176, 1989.

[3] B. J. Lageweg, J. K. Lenstra, and A. H. G. Rinnooy Kan. “Job shop scheduling by implicit enumeration.” *Management Science*, vol. 24, pp. 441-450, 1977.

[4] P. Brucker, B. Jurisch, and B. Sievers. “A branch and bound algorithm for job-shop scheduling problem.” *Discrete Applied Mathematics*, vol. 49, pp. 105-127, 1994.

[5] V. R. Kannan and S. Ghosh. “Evaluation of the interaction between dispatching rules and truncation procedures in job-shop scheduling.” *International journal of production research*, vol. 31, pp. 1637-1654, 1993.

[6] R. Vancheeswaran and M. A. Townsend. “A two-stage heuristic procedure for scheduling job shops.” *Journal of Manufacturing Systems*, vol. 12, pp. 315-325, 1993.

[7] Z. He, T. Yang, and D. E. Deal. “Multiple-pass heuristic rule for job scheduling with due dates.” *International journal of production research*, vol. 31, pp. 2677-2692, 1993.

[8] J. adams, E. Balas, and D. Zawack. “The shifting bottleneck procedure for job shop scheduling.” *Management Science*, vol. 34, pp. 391–401, 1988.

[9] E. Nowicki and C. Smutnicki. “A fast taboo search algorithm for the job-shop problem.” *Management Science*, vol. 42, no. 6, pp. 797-813, June 1996.

[10] S. G. Ponnambalam, P. Aravindan, and S. V. Rajesh. “A tabu search algorithm for job shop scheduling.” *International Journal of Advanced Manufacturing Technology*, vol. 16, pp. 765-771, 2000.

[11] P. V. Laarhoven, E. Aarts, and J. K. Lenstra. “Job shop scheduling by simulated annealing.” *Operations Research*, vol. 40, pp. 113-125, 1992.

[12] J. B. Chambers. “Classical and flexible job shop scheduling by tabu search,” Ph.D. dissertation, University of Texas at Austin, Austin, TX, 1996.

[13] M. E. Aydin and T. C. Fogarty. “Simulated annealing with evolutionary processes in job shop scheduling,” in Evolutionary Methods for Design, Optimization and Control, (Proceeding Of EUROGEN 2001), Barcelona, 2002.

[14] M. Kolonko. “Some new results on simulated annealing applied to job shop scheduling problem.” *European Journal of Operational Research*, vol. 113, pp. 123-136, 1999.

[15] T. Satake, K. Morikawa, K. Takahashi, and N. Nakamura. “Simulated annealing approach for minimizing the makespan of the general job-shop.” *International Journal of Production Economics*, vol. 60-61, pp. 515-522, 1999.

[16] F. D. Croce, R. Tadei, and G. Volta. “A genetic algorithm for the job shop problem.” *Computers and Operations Research*, vol. 22, pp. 15-24, 1995.

[17] J. F. Goncalves, J. J. d. M. Mendes, and M. G. C. Resende. “A hybrid genetic algorithm for the job shop scheduling problem.” *European Journal of Operational Research,* vol. 167, pp. 77-95, 2005.

[18] L. Wang and D. Z. Zheng. “A Modified Genetic Algorithm for Job Shop Scheduling.” *International journal of advanced manufacturing technology*, pp. 72-76, 2002.

[19] R. T. Mogaddam, F. Jolai, F. Vaziri, P. K. Ahmed, and A. Azaron. “A hybrid method for solving stochastic job shop scheduling problems.” *Applied Mathematics and Computation*, vol. 170, pp. 185-206, 2005.

[20] L. Wang and D. Z. Zheng. “An effective hybrid optimization strategy for job-shop scheduling problems.” *Computers & Operations Research*, vol. 28, pp. 585-596, 2001.

[21] S. Y. Foo, Y. Takefuji, and H. Szu. “Scaling properties of neural networks for job shop scheduling.” *Neurocomputing*, vol. 8, no.1, pp. 79-91, 1995.

[22] J. Zhang, X. Hu, X. Tan, J. H. Zhong, and Q. Huang. “Implementation of an Ant Colony Optimization technique for job shop scheduling problem.” *Transactions of the Institute of Measurement and Control*, vol. 28, pp. 93-108, 2006.

[23] K. L. Huang and C. J. Liao. “Ant colony optimization combined with taboo search for the job shop scheduling problem.” *Computers & Operations Research*, vol. 35, pp. 1030-1046, 2008.

[24] J. Montgomery, C. Fayad, and S. Petrovic. “Solution representation for job shop scheduling problems in ant colony optimization,” Faculty of Information & Communication Technologies, Swinburne University of Technology, 2006.

[25] M. Ventresca and B. Ombuki. “Ant Colony Optimization for Job Shop Scheduling Problem,” in Proceedings of 8th IASTED International Conference On Artificial Intelligence and Soft Computing, 2004, pp. 451-152.

[26] S. Petrovic, and C. Fayad. “A genetic algorithm for job shop scheduling with load balancing,” School of Computer Science and Information Technology, University of Nottingham, Nottingham, 2005.

[27] S. Rajakumar, V. P. Arunachalam, and V. Selladurai. “Workflow balancing in parallel machine scheduling with precedence constraints using genetic algorithm.” *Journal of Manufacturing Technology Management*, vol. 17, pp. 239-254, 2006.

[28] B. M. Ombuki, and M. Ventresca. “Local search genetic algorithms for the job shop scheduling problem.” *Applied Intelligence,* vol. 21, pp. 99-109, 2004.

[29] S. C. Lin, E. D. Goodman, and W. F. Punch. “Investigating parallel genetic algorithms on job shop scheduling problems,” Genetic algorithm research and applications group, State university of Michigan, Michigan, 1995.

[30] R. Cheng, M. Gen, and Y. Tsujimura. “A tutorial survey of job-shop scheduling problems using genetic algorithms, part II: Hybrid genetic search strategies.” *Computers & Industrial Engineering*, vol. 36, pp. 343–364, 1999.

[31] Y. Chen, Z. Z. Li, and Z. W. Wang. “Multi-agent-based genetic algorithm for JSSP,” in Proceedings of the third international conference on Machine Learning and Cybernetics, 2004, pp. 267-270.

[32] L. Asadzadeh, K. Zamanifar. *“*An Agent-based Parallel Approach for the Job Shop Scheduling Problem with Genetic Algorithms*.” Mathematical and Computer Modeling*, Vol. 52, pp. 1957-1965, 2010.

[33] F. Bellifemine, A. Poggi, and G. Rimassa. “Developing multi-agent systems with a FIPA-compliant agent framework.” *Software: Practice and Experience*, vol. 31, pp. 103-128, 2001.

[34] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. MA: Addison-Wesley, 1989.

[35] D. E. Goldberg and R. Lingle. “Alleles, loci, and the TSP,” in Proceeding of 1st International Conference on Genetic Algorithms*,* 1985, pp. 154-159.

[36] D. C. Mattfeld and R. J. M. Vaessens. “Job shop scheduling benchmarks.” Internet: www mscmga.ms.ic.ac.uk, [Jul. 10, 2008].