Journal of Theoretical and Applied Information Technology
20th April 2014. Vol. 62 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

SOLVING THE JOB SHOP SCHEDULING PROBLEM WITH A PARALLEL AND AGENT-BASED LOCAL SEARCH GENETIC ALGORITHM

LEILA ASADZADEH
Department of Computer Engineering and Information Technology, Payame Noor University, I. R. of IRAN
E-mail: leila_asadzadeh_cs@yahoo.com , l_asadzadeh@pnu.ac.ir

ABSTRACT

Job shop scheduling problems play an important role in both manufacturing systems and industrial process for improving the utilization of resources, and therefore it is crucial to develop efficient scheduling technologies. The Job shop scheduling problem, one of the best known production scheduling problems, has been proved to be NP-hard. In this paper, we present a parallel and agent-based local search genetic algorithm for solving the job shop scheduling problem. A multi agent system containing various agents each with special behaviors is developed to implement the parallel local search genetic algorithm. Benchmark instances are used to investigate the performance of the proposed approach. The results show that the proposed agent-based parallel local search genetic algorithm improves the efficiency.

Keywords: *Job Shop Scheduling Problem, Parallel Genetic Algorithms, Local Search, Multi Agent System*

1. INTRODUCTION

The Job Shop Scheduling Problem (JSSP) is a real-world problem in a field of production management. To survive in the modern competitive marketplace, which requires lower cost and shorter product life cycles, a corporation must respond quickly and precisely to the customer's demands. Effective scheduling plays an important role in this adaptation. JSSP is an optimization problem that can be described in terms of a set of jobs, each with one or more operations. The operations of a job have to be processed in a specified sequence on a specific set of machines. The time required for all operations to complete their processes is called the makespan. The objective of JSSP aims to minimize the makespan value. Job shop scheduling problem is a difficult NP-hard combinatorial optimization problem.

Meta-heuristics are one of many approximation methods widely used to solve practical optimization problems. In recent years, several algorithms employing a meta-heuristic approach such as Genetic Algorithm (GA), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), Bee Colony Optimization (BCO), and Artificial Bee Colony (ABC) have been applied to solve JSSP.

Genetic algorithms have been implemented successfully in many scheduling problems, in particular job shop scheduling. Parallelization and Hybridization are an extremely effective ways of improving the performance and effectiveness of genetic algorithms.

Parallel GA (PGA) is explored by the researchers in 1980s and 1990s to make the technique faster by executing GAs on parallel computers. PGA can be classified into four types: master-slaver model, coarse-grained, fine-grained and hybrid algorithm [1]. Many researchers employed these types of PGA in solving JSSP with varying degrees of success. Zhang and Chen [2] 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. Kirley [3] had divided JSSP into sub-problems of lower complexity and used parallel evolution of partial solutions. Asadzadeh and Zamanifar [4] proposed an agent-based parallel genetic algorithm approach. This parallel approach is based on a coarse-grained model. The initial population is divided into sub-populations, and each sub-

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

population is evolved separately. Communication between sub-populations is restricted to the migration of chromosomes. Lin et al. [5] 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.

The most common form of hybridization is to couple genetic algorithms with problem-specific methods in order to make the approach really effective. Traditional heuristic methods are incorporated to enhance the performance of genetic search. Ombuki and Ventresca [6] 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. Wang and zheng [7] 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 this paper, we use both parallelization and hybridization to enhance the efficiency of GA and propose an agent-based parallel local search genetic algorithm for solving the job shop scheduling problem. A multi agent system containing various agents each with special behaviors is developed to implement the parallel local search genetic algorithm.

The reminder of this paper is organized as follow. In Section 2, we describe our proposed parallel local search genetic algorithm and its details. Experimental results are described in section 3. Our conclusion is given in Section 4.

2. AGENT-BASED PARALLEL LOCAL SEARCH GENETIC ALGORITHM FOR JSSP

Recent years have seen more active research on parallel genetic algorithms (PGAs) being applied in solving difficult problems. Hard problems normally require a bigger population, and this implies the requirement of higher computational power. One of the most popular PGAs is the coarse grained PGAs where the population is subdivided into a few subpopulations keeping them relatively isolated from each other. This model of parallelization introduces an extra operator from normal GAs, which is the migration operator, used to send individuals from one subpopulation to another. In this research, we use the island model where the population is partitioned into small subpopulations by geographical isolation and migration can happen between neighbor islands (subpopulation). The island model usually has several isolated subpopulations of individuals evolve in parallel where each island does its own genetic operation and periodically sharing its best individuals through migration.

For more complicated problems a genetic algorithm needs to be integrated with problem-specific methods in order to make the approach really effective. Coupling genetic algorithms with local search techniques can be an extremely effective way of improving the performance and effectiveness of these algorithms. A local search operator can be incorporated into the genetic algorithm by applying the operator to each member of the population after each generation. This hybrid method often carried out in order to produce stronger results than the individual approaches can achieve on their own.

Agents and multi-agent systems have wide application in parallel and distributed systems. One of the important features of agents is their capability in parallel implementing of genetic algorithms. We used this feature in our approach and proposed a parallel and distributed model for job shop scheduling problem. To implement our parallel local search genetic algorithm, a multi agent system containing some intelligent agents is developed. Agents of multi agent system have special actions that are used to implement the parallel genetic algorithm.

In this section the proposed agent-based model for JSSP is introduced and its structure and details are described. The architecture of agent-based model is introduced in section 2.1. In section 2.2, parallel genetic algorithm, local search procedure and the migration mechanism are illustrated.

2.1 Architecture of Proposed Agent-Based Model

To implement the agent-based model, we used JADE [8] as a platform and built our agents on it. The model contains various agents that communicate over the context that provided by JADE middleware. Agents developed on JADE can interoperate with other agents built with the same standard. JADE allows each agent to dynamically

Journal of Theoretical and Applied Information Technology
20th April 2014. Vol. 62 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

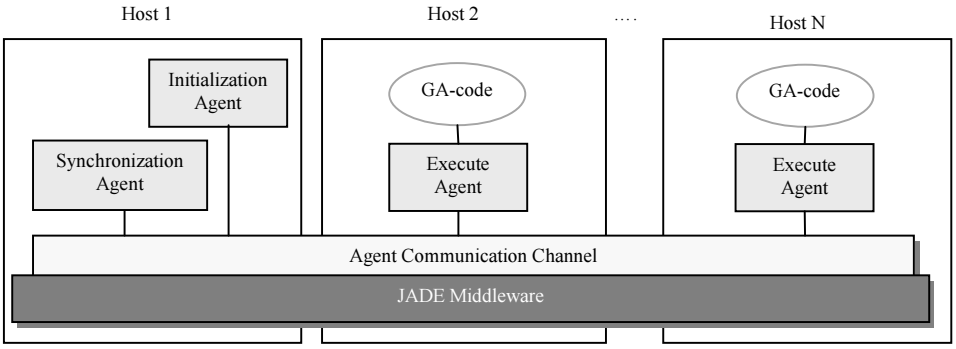

Figure 1: Proposed Agent-based Architecture for Job Shop Scheduling Problem

discover other agents and to communicate with them in a peer-to-peer manner.

An overall schematic of this architecture has represented in *Figure 1*. Agents distributed over various hosts in the network and JADE provides a secure communication channel for them to communicate. We have some containers in our platform that agents are executed on them.

Each agent of the proposed model has been developed for a special purpose. *Initialization Agent (IA)* has the responsibility of creating the initial population for the genetic algorithm and computing the fitness value of chromosomes. This agent locates on main host and controls the behaviors of other agents in the platform. Each *Execute Agent (EA)* locates on a distinct host and executes genetic algorithm on its sub-population. *Synchronization Agent (SA)* locates on main host and coordinates migration between sub-populations of EA agents.

2.2 Parallel Local Search Genetic Algorithm

In this paper, we use the island model to implement our parallel local search genetic algorithm where the population is partitioned into small subpopulations and migration can happen between neighbor islands. The island model usually has several isolated subpopulations of individuals evolve in parallel where each island does its own genetic operation and periodically sharing its best individuals through migration.

To improve the performance and effectiveness of the parallel genetic algorithm, we couple the genetic algorithm with local search technique. A local search procedure is incorporated into the genetic algorithm by applying the operator to each member of the population at each generation. This hybrid method produces stronger results than the individual approaches.

The agent-based parallel local search genetic algorithm is shown in *Figure 2*. As shown in this figure, after loading the genetic algorithm and JSSP parameters, IA creates genetic population and computes fitness value of chromosomes then it divides it to some sub-populations with the same size and sends each of them to a execute agent (EA). Each EA locates on a distinct host and executes genetic algorithm on its sub-population independently. Different sub-populations communicate with exchanging of migrants. Parallel local search genetic algorithm consists of two phases: evolution phase and migration phase. In evolution phase, sub-populations are evolved independently by execute agents and in migration phase, exchanging of migrants is done. These two phases run repeatedly for predefined times.

2.2.1 Local search procedure

After creating a new chromosome using the crossover operator, EA applies Local search procedure on it to improve its quality. A local search based on the Variable Neighboring Search method (VNS) [9] is performed on the chromosomes. This procedure is shown in *Figure 3*. Local search procedure consists of two mutation operator: *Exchange* and *Insert* that are applied on the selected chromosome randomly. The new created chromosome is accepted if the fitness of it be better than previous one. *Figure 4* illustrates an example of *Exchange* and *Insert*.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

```
Begin
1    IA Initializes the JSSP and GA parameters;
2    IA creates the genetic population and calculates the fitness value of chromosomes and sends chromosomes to EAs;
3      For each EA do in parallel
4        Receive sub-population P from IA;
5        While generation-span not satisfied do
           /* Evolution phase */
6          i=0;
7          Create a new sub-population P′ ;
8          While (i < migration-frequency) do
9            While ( all chromosomes not selected) do
10             Select two chromosomes from P;
11             Combine chromosomes by crossover operator and create new chromosomes;
12             Execute local search procedure on new chromosomes;
13             Add new chromosomes to P′ ;
14           End while
15           P = P′;
16           Remove all chromosomes of P′;
17           i=i+1;
18         End while
           /* Migration phase */
19         Send message to SA;
20         Receive message from SA and start migration;
21         Select some of best solutions from sub-population and send them to neighbors;
22         Receive bests from neighbors and update the sub-population;
23       End while
24     End for
End
```

Figure 2: Agent-based Parallel Local Search Genetic Algorithm

```
Procedure Local Search
  Get initial solution xS
  x = xS
  p=1
  n = number of jobs
  m = number of machines
  α = random_integer_number [1, nm]
  β = random_integer_number [1, nm], β ≠ α
  x = Exchange(x, α, β)
  α = random_integer_number [1, nm]
  β = random_integer_number [1, nm], β ≠ α
  x = Insert(x, α, β)
  α = random_integer_number [1, nm]
  β = random_integer_number [1, nm], β ≠ α
  x = Exchange(x, α, β)
  For i=1 to nm do
    α = random_integer_number [1, nm]
    β = random_integer_number [1, nm], β ≠ α
    If (p=1) then x' = Exchange (x, α, β)
    Else if (p=0) then x' = Insert(x, α, β)
    If (fitness (x' ) ≥ fitness (x)) then  x = x'
    Else p = | p - 1|
  End for
 If (fitness (x) ≥ fitness (xS)) then xS = x
End Procedure
```

Figure 3: Local Search Procedure

Journal of Theoretical and Applied Information Technology
20th April 2014. Vol. 62 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

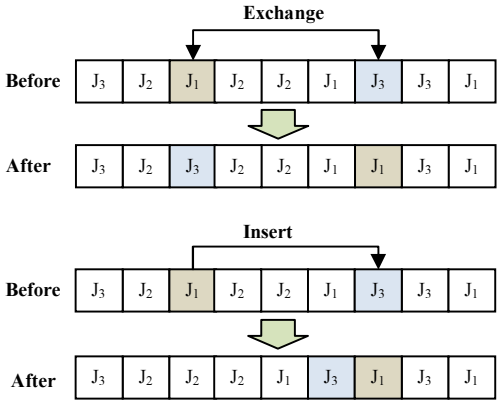

Figure 4: Exchange and Insert Mechanisms

2.2.2 Migration mechanism

Communication between various sub-populations is carried out by exchanging of migrants. In our approach we use synchronous migration policy. Each EA executes local search genetic algorithm on its sub-population for a predefined number of generations then it sends a message to SA informing end of its evolution phase.

The SA is a synchronization agent, which coordinates migration between sub-populations of EA agents. After receiving message from all the EAs, SA broadcasts a message to them notifying start of the migration phase. In the migration phase, each EA exchanges some of its best chromosomes with its neighbors. Chromosomes with low fitness value in sub-population are replaced with the best chromosomes of neighbors.

3. EXPERIMENTAL RESULTS

Before evaluating our parallel local search genetic algorithm we explain detailed implementation of the genetic algorithm such as chromosome representation, fitness function, selection mechanism and crossover operator as follow.

a) Chromosome representation: The feasible solutions of job shop scheduling problem are encoded by chromosomes. We use a method that was introduced by Gen and Tsujimura [10] to encode chromosomes of the problem. In this method, each job has a distinct number that is used to indicate its operations. In the n jobs and m machines problem, the maximum number of genes in each chromosome is n×m. Job J_i appears in a chromosome m times. An example chromosome of the problem in *Table 1* is (3 2 1 2 3 1 1 3 2). In this chromosome, 1 means job J_1, 2 means job J_2, etc. Job J_1 has three operations; thus the number of times 1 appears in the chromosome is three.

Table 1: An Example of a 3×3 JSSP

Job	Machine, Processing time		
J_1	1,3	2,3	3,3
J_2	1,2	3,3	2,4
J_3	2,3	1,2	3,1

b) Fitness function: To determine the survival probability of a chromosome at the next generation, we need to define a fitness function. In this paper, we use a well-known fitness function proposed by Goldberg [11] to evaluate the chromosomes. Suppose that X_c is a chromosome with makespan value $M(X_c)$ and the M_{max} is the maximum value of makespan in the genetic population. The Fitness of chromosome X_c is calculated as follows:

$$Fit\ (X_c) = M_{max} - M(X_c) \quad (1)$$

c) Selection mechanism: Roulette wheel selection containing the elite retaining model [11] is used to select chromosomes. In the elite retaining model, the best chromosome from the previous generation is copied to the next generation. Hence the best produced solution can never become worse from one generation to the next.

d) The crossover operator: After the selection of chromosomes, in order to create new chromosomes for the next generation a crossover operator is used. We use the partially matched crossover (PMX) proposed by Goldberg and Lingle [12]. Two crossover points are chosen from the chromosomes randomly and equably. Then the genes of two parents that are in the area between the crossover points are exchanged. This operator can produce illegal schedules. The IA repairs each illegal schedule and converts it to a legal one.

We evaluate the agent-based local search genetic algorithm using LA problem instances contributed by Lawrence [13]. These problem instances are available from the OR library web site [14].

The proposed parallel local search genetic algorithm is implemented on a network with five computers. One of computers is indicated as main host and executes IA and SA. The number of EA agents was fixed at four in our experiments and these agents form a virtual cube among them. Each EA has two neighbors.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

The parameter values for parallel local search genetic algorithm are defined as follows: population consists of 100 chromosomes (so each ub-population has 25 chromosomes), the generation span is 100, the crossover rate is 0.95, the migration frequency is 10 generations, the migration rate is 5 chromosomes and the migration topology is cube.

In order to determine the performance of the parallel local search genetic algorithm, results are compared with other algorithms (Ombuki and Ventresca [6], Asadzadeh and Zamanifar [4], Dorndorf and Pesh [15], Goncalves et al. [16] and Binato et al. [17]). A summary of experimental results is given in Table 2. This table lists problem name, problem size (number of jobs × number of machines), the best known solution (BKS), the solution obtained by parallel local search genetic algorithm (PLSGA) and the solution obtained by each of the other algorithms.

As shown in *Table 2*, the PLSGA obtained the best known solution for almost all problem instances and the quality of solutions that have been found is better than almost all other algorithms except the Goncalves et al. [16] algorithm that had better solution quality in LA22 and LA27 instances.

For the purpose of showing the behavior of the convergence point, the parallel local search genetic algorithm applied on LA21 and LA15 instances during various generations and the average makespan of the best schedules obtained are shown in *Figures 5-6*. As shown by these figures, convergence speed in the PLSGA is very high. Thus coupling genetic algorithms with local search heuristic and parallelization accelerates the convergence speed and improves the performance of these algorithms.

Table 2: Experimental Results on Benchmark Instance

Problem	Size (n×m)	BKS	PLSGA	Ombuki and Ventresca	Asadzadeh and Zamanifar	Dorndorf and Pesh	Goncalves et al.	Binato et al.
LA01	10×5	666	666	666	666	666	666	666
LA02	10×5	655	655	655	655	681	655	655
LA03	10×5	597	597	597	617	620	597	604
LA08	15×5	863	863	863	863	863	863	863
LA09	15×5	951	951	951	951	951	951	951
LA10	15×5	958	958	958	958	958	958	958
LA12	20×5	1039	1039	1039	1039	1039	1039	1039
LA14	20×5	1292	1292	1292	1292	1292	1292	1292
LA15	20×5	1207	1207	1207	1273	1207	1207	1207
LA16	10×10	945	945	959	994	1008	945	946
LA17	10×10	784	784	792	793	809	784	784
LA18	10×10	848	848	857	860	916	848	848
LA19	10×10	842	842	860	873	880	842	842
LA20	10×10	902	907	907	912	928	907	907
LA21	15×10	1046	1046	1114	1146	1139	1046	1091
LA22	15×10	927	960	989	1007	998	935	960
LA23	15×10	1032	1032	1035	1033	1072	1032	1032
LA26	20×10	1218	1218	1307	1323	1278	1218	1271
LA27	20×10	1235	1281	1350	1359	1378	1256	1320
LA30	20×10	1355	1355	1451	1437	1411	1355	1368
LA31	30×10	1784	1784	1784	1844	-	1784	1784
LA32	30×10	1850	1850	1850	1907	-	1850	1850
LA33	30×10	1719	1719	1745	-	-	1719	1719

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

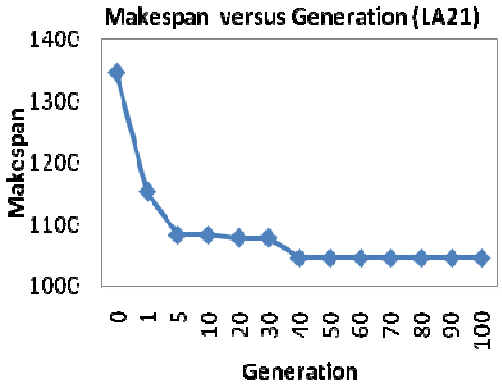

Figure 5: Convergence Curves for *PLSGA for LA21 Instance.*

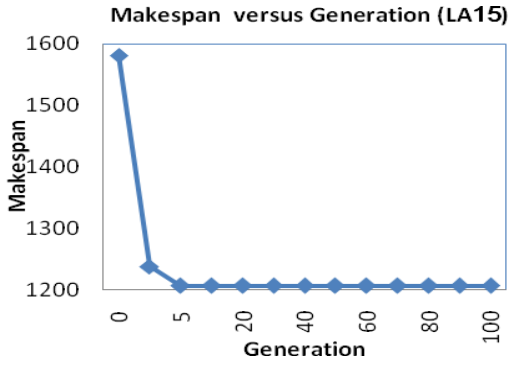

Figure 6: Convergence Curves for *PLSGA for LA15 Instance.*

4. CONCLUSION

In this paper, we solved the job shop scheduling problem with an agent-based parallel local search genetic algorithm. A multi agent system containing various agents each with special behaviors was developed to implement the parallel local search genetic algorithm. We used the island model to parallelize the genetic algorithm where the population is partitioned into small subpopulations and migration can happen between neighbor subpopulation. To enhance the efficiency of the genetic algorithm, a local search procedure was applied.

The results obtained from our proposed method showed that the agent-based parallel local search genetic algorithm is effective in finding optimal and near optimal solutions for various instances of the job shop scheduling problem. The results also showed that convergence speed of our algorithm is high and optimal or near optimal solutions are found rapidly. Parallelization and local search heuristic accelerate the convergence speed and improve the performance of genetic algorithms. In the future, we will concentrate the proposed method to apply it on other optimization problems.

ACKNOWLEDGMENTS

This work is supported by the Payame Noor University of I. R. of Iran through the research program under Grant no. D/72096/7.

REFERENCES:

[1] R. Yusof, M. Khalid, G.T. Hui, S.M. Yusof, and M.F. Othman, "Solving job shop scheduling problem using a hybrid parallel micro genetic algorithm", *Applied Soft Computing*, Vol. 11, 2011, pp.5782–5792.

[2] H. Zhang and 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.

[3] M. Kirley, "A co-evolutionary 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.

[4] L. Asadzadeh and K. Zamanifar, "An agent-based parallel approach for the job shop scheduling problem with genetic algorithms", *Mathematical and Computer Modeling*, Vol. 52, No. 11-12, 2010, pp.1957–1965.

[5] 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.

[6] B.M. Ombuki, and M. Ventresca, "Local search genetic algorithms for the job shop scheduling problem", *Applied Intelligence*, Vol. 21, 2004, pp. 99–109.

[7] L. Wang, and D.Z. Zheng, "An effective hybrid optimization strategy for job shop scheduling problems", *Computers & Operations Research*, Vol. 28, 2001, pp. 585–596.

[8] F. Bellifemine, A. Poggi, and G. Rimassa, "Developing multi-agent systems with a FIPA-compliant agent framework", *Software: Practice and Experience*, Vol. 31, 2001, pp. 103–128.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[9] P. Hansen, N. Mladenovic´, and J.A.M. Pérez, "Variable neighborhood search: methods and applications. 4OR: Quarterly journal of the Belgian", *French and Italian operations research societies*, Vol. 6, 2008, pp. 319–360.

[10] M. Gen, and Y. Tsujimura, "Genetic algorithms for solving multiprocessor scheduling problems", *In Proceeding of 1st Asia-pacific conference on simulated evolution and learning*, 1997, pp. 106–115.

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

[12] D.E. Goldberg, and R. Lingle, "Alleles, loci, and the TSP", *In Proceeding of 1st International Conference on Genetic Algorithms*, 1985, pp. 154–159.

[13] S. Lawrence, "Resource Constrained Project Scheduling: An Experimental Investigation of Heuristic Scheduling Techniques", Graduate School of Industrial Administration, Carnegie Mellon University, Pittsburgh, 1984.

[14] D.C. Mattfeld, and R.J.M. Vaessens, "Job shop scheduling benchmarks", Internet: www mscmga.ms.ic.ac.uk, 2008.

[15] U. Dorndorf, and E. Pesch, "Evolution based learning in a job shop environment", *Computers and Operations Research*, Vol.22, 1995, pp. 25-40.

[16] 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, 2005, pp. 77–95.

[17] S. Binato, W.J. Hery, D.M. Loewenstern, and M.G.C. Resende, "A GRASP for job shop scheduling", *In C.C. Ribeiro, P. Hansen, (Eds.), Essays and Surveys in Metaheuristics*, Kluwer Academic Publishers, 2002.