2016

An Efficient Solution to the Mixed Shop Scheduling Problem Using a Modified Genetic Algorithm

V. Nguyen
Old Dominion University

H. P. Bao
Old Dominion University, hbao@odu.edu

Original Publication Citation

Nguyen, V., & Bao, H. P. (2016). An efficient solution to the mixed shop scheduling problem using a modified genetic algorithm. *Procedia Computer Science, 95*, 475-482. doi:https://doi.org/10.1016/j.procs.2016.09.324

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 95 (2016) 475 – 482

Procedia Computer Science

Complex Adaptive Systems, Publication 6
Cihan H. Dagli, Editor in Chief
Conference Organized by Missouri University of Science and Technology
2016 - Los Angeles, CA

An Efficient Solution to the Mixed Shop Scheduling Problem Using a Modified Genetic Algorithm

V. Nguyen[a]* and H. P. Bao[b]

[a] *Ph.D. Student, Mechanical & Aerospace Engineering Department, Old Dominion University, Kaufman Hall, Norfolk, VA 23529*
[b] *Professor, Mechanical & Aerospace Engineering Department, Old Dominion University, Kaufman Hall, Norfolk, VA 23529*

Abstract

The mixed job shop scheduling problem is one in which some jobs have fixed machine orders and other jobs may be processed in arbitrary orders. In past literature, optimal solutions have been proposed based on adaptations of classical solutions such as by Johnson, Thompson and Giffler among many others, by pseudopolynomial algorithms, by simulation, and by Genetic Algorithms (GA). GA based solutions have been proposed for flexible Job shops. This paper proposes a GA algorithm for the mixed job shop scheduling problem. The paper starts with an analysis of the characteristics of the so-called mixed shop problem. Based on those properties, a modified GA is proposed to minimize the makespan of the mixed shop schedule. In this approach, sample instances used as test data are generated under the constraints of shop scheduling problems. A comparison of our results based on benchmark data indicate that our modified GA provides an efficient solution for the mixed shop scheduling problem.

Peer-review under responsibility of scientific committee of Missouri University of Science and Technology

Keywords: Mixed shop scheduling; Shop scheduling; Genetic Algorithm; Makespan optimization

* Corresponding author. Tel.: +757 355 3389.
E-mail address: vnguy025@odu.edu

Peer-review under responsibility of scientific committee of Missouri University of Science and Technology
doi:10.1016/j.procs.2016.09.324

1. Introduction

Scheduling is an important activity in any manufacturing environment. It is a key factor for enhancing manufacturing productivity and meeting customer demand, which plays a significant role in customer satisfaction. Shop scheduling is one of the most interesting areas of research in the field of scheduling due to the demands of industry. In a shop scheduling problem, there are a set of n jobs, each of which consists of a set of operations processed on a set of m machines. According to the restrictions on the technological routes of the jobs, a general shop is indicated by three well-known models: a flow shop (each job is characterized by the same technological route), a job shop (each job has a specific route) and an open shop (no technological route is imposed on the jobs) [1]. With the purpose of making the model more realistic, mixed shop is introduced in the literature of shop scheduling problems. Mixed shop is a combination of the job shop problem and the open shop problem. It is an NP-hard problem. In past research, several researchers had worked on mixed-shop scheduling problem. Strusevich (1991) studied polynomial-time algorithm to find both pre-emptive and non-preemptive optimal mixed-shop schedules. Shakhlevich, Sotskov, and Werner (1999) discussed the complexity of mixed-shop problem under various criteria by presenting different polynomial and pseudopolynomial algorithms. Liu and. Ong (2013) proposed the Bacterial Foraging Optimization algorithm which is featured with an Ant Colony Optimization algorithm and proposed a nature inspired computing approach to solve the Mixed Shop Scheduling problem.

Genetic algorithm has been applied to "pure" shop scheduling problems (flow shop, job shop and open shop) and combinatorial problems such as flexible job-shop and stage shop scheduling. The results have proved that genetic algorithm is an effective method to solve shop scheduling problems with less computational effort and better results [18].In this paper, we propose an improved genetic optimization process based on the general genetic algorithm to solve the mixed-shop scheduling optimization problem. The objective of this model is to minimize makespan. In comparison with results obtained through classic solutions by Giffler and Thompson (1960) our solution proves itself to be a superior one. The algorithm will then be tested with the benchmark problems from 3x3 to 20x20 jobs x machines which are taken from the OR-Library website [11].

2. Definition of mixed –shop scheduling problem and proposed modification to the general GA algorithm

Let $J = \{J_i\}_{1\le i\le N}$ be a set of *N* jobs to be scheduled where each job J_i consists of n_i operations. Let $O_{i,j}$ be the j^{th} operation of J_i. Let $M = \{M_k\}_{1\le k\le m}$ be a set of *m* machines and let p_{ij} be the processing time of $O_{i,j}$ on machine $M_{i,j} \in M$. The set N is split into two subsets: $N = N_J \cup N_O$. The jobs of the set N_J have to be processed as in the job-shop: for any job $J_i \in N_J$, there is a given machine order $l_i = (M_{ik1},...,M_{ik_{ni}})$ which determines a sequence of operations of that job: $(O_{i,1},...,O_{i.n_i})$, where operation $O_{i,j}$ has to be processed after operation $O_{i,j-1}$ with j=2,…, n_i. The jobs of the set N_O have to be processed as in the open-shop: each job $J_i \in N_O$ has to be processed exactly once on each machine and the machine order of this job is not fixed before scheduling. Given a schedule, we denote by st_{ij} and C_{ij} the starting time and completion time of operation $O_{i,j}$ ($1\le i\le N, 1\le j\le n_i$), respectively. The objective is to find a schedule having a minimum completion time (or, *makespan*), denoted by $C_{\max} = \max_{i=1..N}(C_i)$, where $C_i = \max_{1\le j\le n_i}(C_{i,j})$ is the completion time of job J_i. [3]

An example of mixed-shop scheduling with 3 jobs and 3 machines is shown in table 1 below:

Table 1. A 3 jobs by 3 machines scheduling problem

	Sequence of operations		
	1	2	3
J1 (Job Shop)	(1,3)	(2,1)	(3,2)
J3 (Job Shop)	(2,3)	(3,2)	(1,3)
J2 (Open Shop)	Any order (3,1)	Any order (1,5)	Any order (2,3)

Notation: (m,p) or (machine, processing time). For example (2, 3) means machine 2 processing time 3

The following assumptions apply for the mixed-shop problem:

- Each machine can only execute one operation at a time and, once started, the operation cannot be interrupted
- All jobs are released at time t = 0
- There is no transportation time between machines

In this paper**,** we propose to modify the general genetic algorithm for mixed –shop scheduling problems as follows.

Let us give each operation a task ID. Table 2 lists the jobs and operations in each job with the corresponding task ID

Table 2. Task IDs

Job	Op.	Task ID	Job	Op.	Task ID	Job	Op.	Task ID
1	1	1	2	1	4	3	1	7
	2	2		2	5		2	8
	3	3		3	6		3	9

In mixed shop scheduling problem, there are two kinds of constraints: precedence constraints and non-simultaneous constraints. Precedence constraints (call set ϕ) applied for jobs that belong to the job-shop set. Non-simultaneous constraints – called set ψ - are for the set of pairs of tasks that cannot be performed simultaneously because of sequence requirement, belonging to the same job or requiring the same machine. One of the key features of this paper is the swapping technique as explained below.

2.1. Chromosome representation

A chromosome of the GA represents a sequence of tasks in which a gene is a task ID.

Chromosome	1	5	3	4	9	8	6	2	7
Sequence consideration	1	2	3	4	5	6	7	8	9

Figure 1. A sample chromosome

In this chromosome, the 5th gene has a value is 9; this means that the 5th task (indicated in the sequence consideration) is task ID 9, which is operation 3 of job 3.

2.2. Precedence requirement

Check the chromosome to see if it satisfies the precedence constraint o. If it does not satisfy the constraint, then swap the position to make it satisfy.

In the example, $\phi = \{(1,2,3),(7,8,9)\}$ by observation, and there are clear violations between task IDs 3 and 2, and task ID 9 and 7. Therefore, the sample chromosome is modified as follows:

Modified chromosome	1	5	2	4	7	8	6	3	9
Sequence handling	1	2	3	4	5	6	7	8	9

Figure 2. Modified sample chromosome in figure 2

Base on the task IDs, We have the information on job, processing time and required machine as following:

Modified chromosome	1	5	2	4	7	8	6	3	9
Sequence considering	1	2	3	4	5	6	7	8	9
Job	1	2	1	2	3	3	2	1	3
Processing time	3	5	1	1	3	2	3	2	3
Required machine	1	1	2	3	2	3	2	3	1
Operation _of job_	1 of 1	Open	2 of 1	Open	1 of 3	2 of 3	Open	3 of 1	3 of 3

Figure 3. Job, processing time required machine and operation number information corresponding to the modified chromosome in figure 3.

2.3. Objective function and fitness evaluation

Based on the sequence of operations represented by the chromosome, the objective function (makespan) can be calculated through the following procedure:

Step 1: Operation is considered based on the order in the sequence. Determine the ready time for each machine. Check type of operation.

Step 2: If operation is open shop type, then its start time is the maximum value of its machine ready time and the complete time of its non-simultaneous operations.

If operation is job shop type, its starting time is the maximum value of its machine ready time and the complete time of its precedence operation.

The complete time of the operation is the sum of start time and processing time.

Update the ready time of the machine by the completion time.

Step 3: Steps 1–2 are repeated for the next operation in sequence until the last is considered. The makespan is the maximum value of the complete time of the operation in the sequence.

For fitness evaluation, we apply a simple definition for the fitness, i.e. Fitness=1/makespan

2.4. Tournament selection

Tournament selection provides selection pressure by holding a tournament among S competitors, with S being the tournament size. The winner of the tournament is the individual with the highest fitness of the S tournament competitors. The winner is then inserted into the mating pool.

The mating pool, being comprised of tournament winners, has a higher average fitness than the average population fitness. This fitness difference provides the selection pressure, which drives the GA to improve the fitness of each succeeding generation.

Increased selection pressure can be provided by simply increasing the tournament sizes, as the winner from a larger tournament will, on average, have a higher fitness than the winner of a smaller tournament.

2.5. Order Crossover (OX), Inverse Mutation, and Selection of new generations

For order crossover and inverse mutation, we apply the following steps:

1. Given two parent chromosomes, two random crossover points are selected from one parent, partitioning it into a left, middle and right substring.
2. Produce the child chromosome by copying the middle substring of the first parent into the corresponding positions
3. Delete the tasks from the same middle substring from the 2nd parent. The remaining tasks in the 2nd parent are transferred in sequence from left to right into the empty slots of the child's chromosome.
4. Inverse mutation is performed on the child's chromosome by selecting 2 inverse points then reverse the task order in between these two points.

For the selection of new generations, we apply the Steady-State Method which consists of deleting *n* worst old members and replacing them with *n* best new members. *n* is a parameter to be experimented with.

3. Implementation and results

To illustrate the effectiveness of the proposal approach, we used bench mark problems with 3 x 3, 6 x 6, 10 x 10,15 x 15 and 20 x 20 jobs x machines. [7,11] Note that not all bench mark problems for all three categories of job scheduling, i.e. job shop, open shop, and mixed shop, and their "optimum" solutions are available in the literature. For the cases where no optimum solution exists, we arbitrarily modify the data so that we can proceed toward solutions for all three categories of job shop, open shop and mixed shop. Table 3 summarizes the results based on the available bench mark problems.

Table 3. Results of proposed GA algorithm

Population Size	Job Shop			Open Shop			Mixed Shop		
	Bench Mark?	Optimum result (#)	Our result (%)	Bench Mark ?	Optimum result (#)	Our result (%)	Bench Mark ?	Optimum result (#)	Our result (%)
3 x 3	Yes	11(14)	11 (0%)	No	NA	11	No	NA	11
6 x 6	Yes	55 (15)	55 (0%)	No	NA	47	No	NA	47
10 x 10	Yes	930 (15)	960 (3.2%)	No	NA	739	No	NA	848
15 x 15				Yes	937 (16)	972 (3.7%)			
20 x 20				Yes	1155 (16)	1200 (3.8%)			

[#] Reference where optimum result was obtained
[%] Percentage difference between optimum result and our result

Looking at table 3, it is clear that our modified GA algorithm provides equivalent results with the optimum results if the population size is 6 x 6 or below. For larger populations, i.e. 10 x 10 and above, our solutions appear to be not as good as the optimum results. Nevertheless in all cases our results are off by no more than 4%. Considering that the optimum results were obtained by analytical techniques such as Branch and Bound, and Simulated Annealing which require substantial and complicated formulation as well as sophisticated programming skills, our straight forward modification of the GA algorithm provide a relatively easy approach to yielding almost similar results. This feature can be considered as beneficial for practicing engineers who may lack the time or the programming knowledge to write sophisticated programming codes for their scheduling problems.

In the case of the 6 x 6 and 10 x 10 mixed shop problems, it is interesting to explore further the differences between our results and the ones available in [17]. Figure 4 is a plot of the makespan for 10 trials and figure 5 a plot of the processing times for 10 trials. From these two figures, it is clear that our GA solution yields not only the better and more steady results, i.e. smaller and steady make span, but also the shorter processing times too.

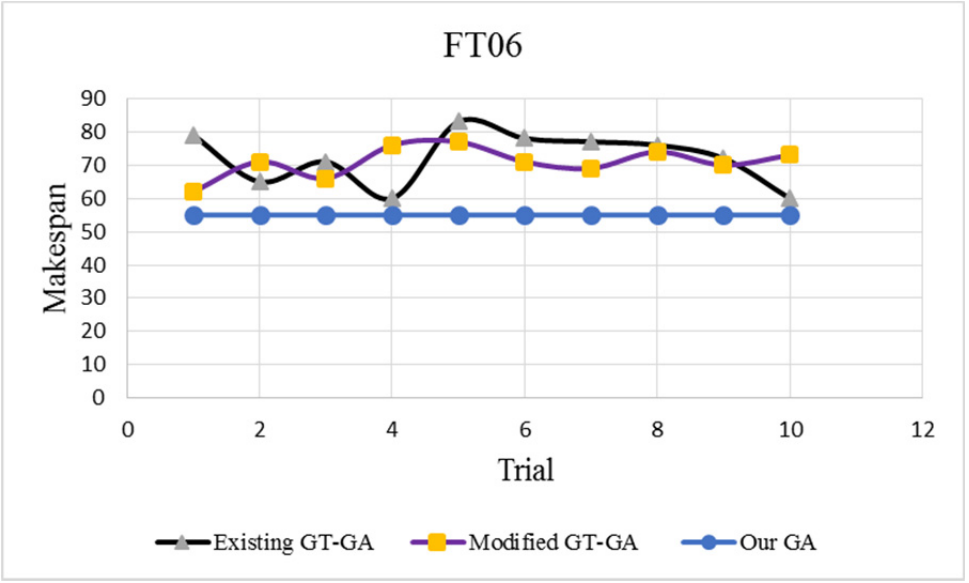

Figure 4.1. Comparison between the existing, modified GT-GA in [17] and our GA in makespan objective (FT 06)

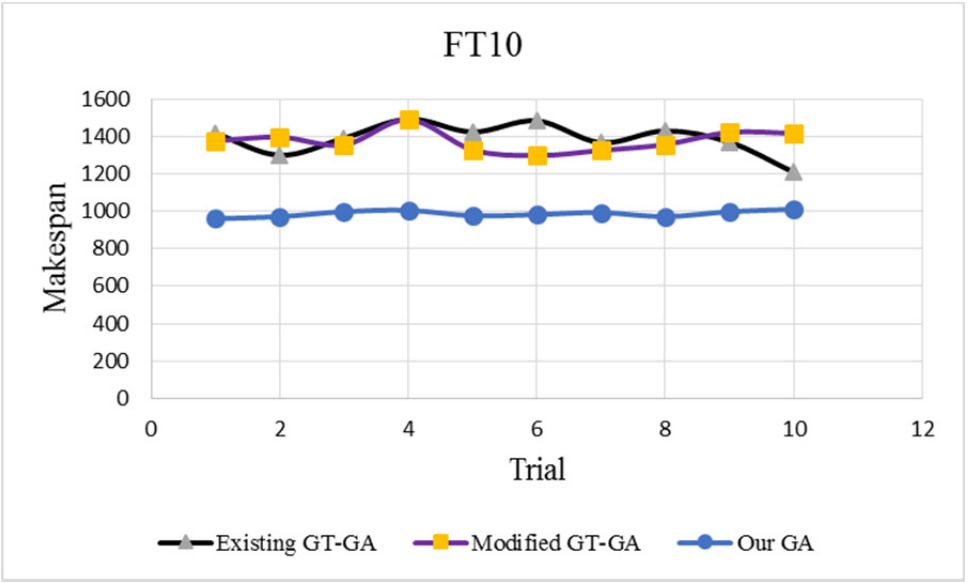

Figure 4.2. Comparison between the existing, modified GT-GA in [17] and our GA in makespan objective (FT10)

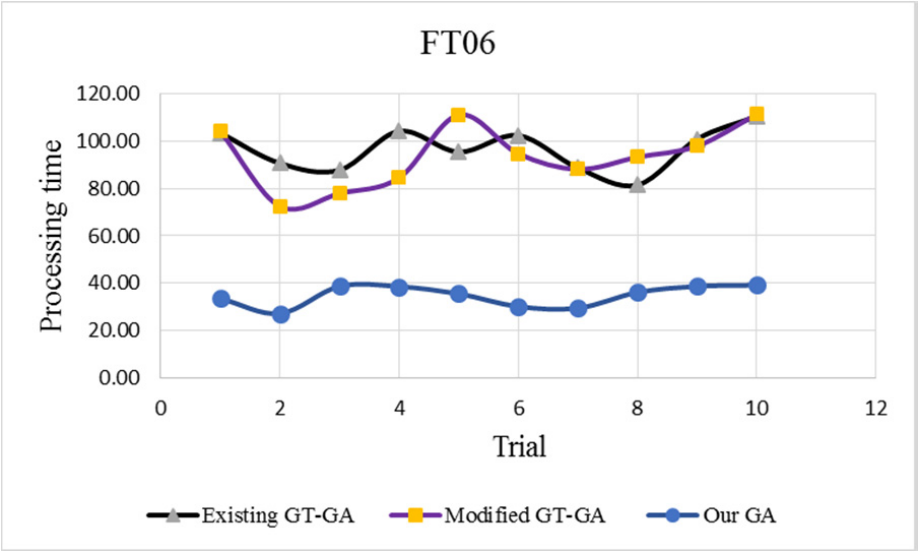

Figure 5.1. Comparison between the existing, modified GT-GA in [17] and our GA in processing time (FT 06)

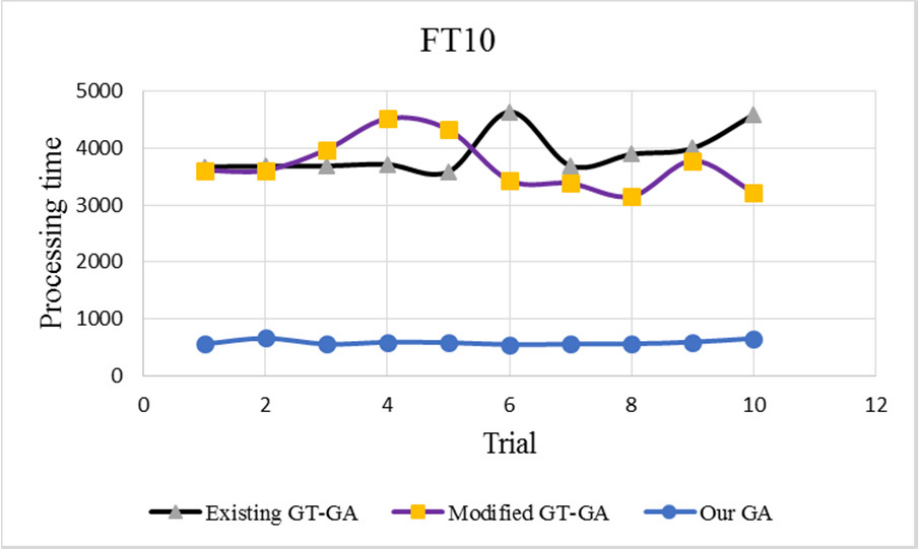

Figure 5.2. Comparison between the existing, modified GT-GA in [17] and our GA in processing time (FT 10)

4. Conclusion

In this paper, we have proposed a G.A. algorithm that is based on a simple reformulation of the problem and the efficient manipulation of the general G.A. algorithm to solve the scheduling of either standard job shop operations or mixed shop operations. We have shown that converting each operation for each job into individual task ID in combination with the classic Giffler Thompson algorithm in a general G.A. framework works very well as demonstrated in the five bench mark examples shown in table 3 above. We demonstrated that, for problem size of 6 x 6 or less, our algorithm matches the optimum solutions available in the literature, but for larger problem sizes, our results are off by no more than 4%. While we did not achieve the optimum results, we want to highlight the benefit of our straight forward GA modification for practicing engineers who may lack the time or the programming

knowledge to write sophisticated programming codes for their scheduling problems. Much remains to be investigated. For example in the above case studies the only resources are the machines and their availability. Quite often the operators are also resources not to be ignored. Bumb has addressed this problem in (8). But he has not applied the task ID concept there, and it should be interesting to see the results had this concept been applied. Multi-mode resource constraints is another fertile area of research in G.A. as discussed by Ghouddousi et al (9). Multi-objective besides minimization of the makespan is also a rich field for research as indicated in (10). In summary in this paper we have found a way of reformulating the scheduling problem. This reformulation approach together with conventional G.A. has allowed us to tackle the problem of machine scheduling for job shop, open shop and mixed shop requirements in an efficient manner (computer time) and scope (type of job scheduling).

References:

1. P. Brucker, *Scheduling Algorithms*, fifth edition, Springer 2007

2. V. A.Strusevich. “Two-machine Super-shop Scheduling Problem”. *The Journal of the Operational Research Society*, vol. 42, No. 6 (1991), pp.479-492.

3. N.V. Shakhlevich, Yu.N. Sotskov, and F.Werer. “Shop scheduling problem with fixed and non-fixed machine orders of jobs”. *Annals of Operations Research*, 92:281-304, 1999.

4. N.V. Shakhlevich, Yu.N. Sotskov, and F.Werer. “Complexity of mixed shop scheduling problem: a survey. “ *European Journal of Operational Research*, 120,343-351, 2000.

5. S. Q. Liu and H. L. Ong, “Metaheuristics for the Mixed Shop Scheduling Problem,” *Asia-Pacific Journal of Operational Research*, Vol. 21, No. 4, 2004, pp. 97-115.

6. M. Pinedo, *Scheduling theory, algorithms, and systems*, second edition, Prentice-Hall, Inc. 2002

7. E. Taillard, “BenchMarks For Basic Scheduling Problems,” *European Journal of Operations Research*, 64, 1993, pp. 278-285

8. N. Bumb, “Job shop Scheduling using G.A. and the Giffler Thompson Approach”, *Master thesis*, Department of Mechanical & Aerospace Engineering, December 2009. Supervisor: Han P. Bao

9. P. Ghouddousi, et al, (2013) “Multi-mode resource-constrained discrete time-cost resource optimization in project scheduling using non-dominated sorting genetic algorithm”. *Automation in Construction* 30(2013), 216-227

10. P. Brucker, et al (1999) “Resource-constrained project scheduling: Notation, classification, models, and methods”, *European Journal of Operations Research* 112 (1999), 3-41.

11. J. E. Beasley 1990 OR-Library http://people.brunel.ac.uk/~mastjjb/jeb/info.html (accessed January 1, 2015)

12. A. Klemmt et al (2009) “Simulation-based optimization vs. mathematical programming: a hybrid approach for optimizing scheduling problems.” *Computer Integrated Manuf*acturing 25 917–25

13. J. Carlier and E. Pinson (1989) “An algorithm for solving the job-shop problem.” *Management Science* 35 164–76

14. C. Zhang, P. Li, Y. Rao, and S. LiA (2005) “New Hybrid GA/SA Algorithm for the Job Shop Scheduling Problem.” Evolutionary Computation in Combinatorial Optimization Volume 3448 of the series Lecture Notes in Computer Science pp 246-259

15. J. E. Beasley, OR-Library http://people.brunel.ac.uk/~mastjjb/jeb/orlib/files/jobshop1.txt

16. J. E. Beasley, OR-Library http://people.brunel.ac.uk/~mastjjb/jeb/orlib/openshopinfo.html

17. L. H. Peng & S. Salim “A Modiffled Giffler and Thompson Genetic Algorithm on the Job Shop Scheduling Problem” MATEMATIKA, 2006, Volume 22, Number 2, pp. 91-107

18. J. Blazewicz, W. Domschke, and E. Pesch “The job shop scheduling problem: conventional and new solution techniques.” European Journal of Operational Research 93 (1996) 1-33

19.P. Brucker**, B. Jurisch, and B. Sievers “A branch and bound algorithm for the job-shop scheduling problem*.” Discrete Applied Mathematics 49 (1994) 107-127