Study on genetic algorithm (GA) approaches for solving Flow Shop Scheduling Problem (FSSP)

IOP Conf. Series: Materials Science and Engineering **857** (2020) 012009 doi:10.1088/1757-899X/857/1/012009

Study on genetic algorithm (GA) approaches for solving Flow Shop Scheduling Problem (FSSP)

A Syarif [1,*], W Wamiliana [2], P Lumbanraja [1] and M Gen[3,]**

[1] Department of Computer Science, Faculty of Mathematics and Sciences, University of Lampung, Bandar Lampung, Indonesia
[2] Department of Mathematics, Faculty of Mathematics and Sciences, University of Lampung, Bandar Lampung, Indonesia
[3] Research Institute for Science and Technology, Tokyo University of Science (TUS), Tokyo, Japan

* Email: admi.syarif@fmipa.unila.ac.id

** Email: mitsuogen@gmail.com

Abstract. The scheduling problem is known as one of the well-known optimization problems. It occurs in many situations of our daily-life applications, especially in industrial fields. One type of scheduling problem is called Flow Shop Scheduling Problem (FSSP). It belongs to the class of NP-complete problem. During the last decades, researches on exploring more accurate and efficient heuristic methods to solve hard optimization problems have taken considerable attention from researchers. Among them, GA has been one of the powerful and widely used algorithms. In this paper, we present two GA approaches to solve FSSP. The main objective is to investigate the effectiveness and the efficiency of GA based on different variations of the chromosome representation, referred to as the job-based GA (jb-GA) and machine-based GA (mb-GA). We conducted numerical experiments using standard test problems (Benchmark test problems). We also present the comparison of results with those given by another heuristic algorithm (NBH Algorithm) and the optimal solutions reported in the literature. Those demonstrate the jb-GA is more effective and efficient almost all of the time. The current limitation of this approach, like many other heuristic methods, is that it still sometimes gives the near-optimal solutions.

Keyword: genetic algorithm, scheduling, benchmark, flow shop

1. Introduction

The scheduling problem concerns the allocation of limited sources over time to perform the task to satisfy specific criteria. This problem exists everywhere in our daily life, especially in industrial applications. However, it is known as one of the hard combinatorial optimization problems. It has highly complex constraints and belongs to the class of NP-hard problems. Despite its NP-Hardness and its importance, during the last decades, many solution methods have also been proposed to solve it [1]. There have been many variations of scheduling problems for different applications [2,3]. In general, there are two types of scheduling problems discussed in the literature. Those are Flowshop Scheduling Problem (FSSP) and Jobshop Scheduling Problem (JSSP).

FSSP occurs when *m* machine process *n* jobs in the same sequence. A different series usually will differ in terms of processing time. An example of FSSP occurs in manufacturing facilities where jobs moved from machine-to-machine. It is widely known as one of the NP-complete optimization problems with n! Possible schedule. Recently, there have been many variations of FSSPs intensively studied in the literature for various applications. There have many variants of solution methods to solve FSSP; most of them are heuristic methods [4].

Nowadays, as computers rapidly increased, researchers have more attention on applying heuristic methods such as Genetic Algorithm (GA), Tabu Search (TS), and Simulated Annealing (SA) for solving various NP-hard/NP-complete optimization problems including Scheduling Problem. Most of the objective is to develop both accurate and efficient heuristic methods. Among them, GA is the most powerful and widely used [5]. Several researchers reported the successful implementation of GA to solve a wide variety of real-world applications, including engineering, economics, finance, manufacturing, agriculture, business, etcetera. In our previous works, we also have reported the success of GA for various combinatorial optimization problems [6], [7], [8], and [9]. Though GA has been a versatile approach for searching the global optimality, it also has a disheartening weakness in gaining too many time to reach optimal solutions. The success of GA depends on several factors, including an efficient design of the chromosome representation, method of crossover and mutation, selection methods, and the value of GA parameters. Thus, research on determining an efficient design of the GA approach for a specific problem becomes very crucial.

In this research work, we present two GA approaches called job-based GA (jb-GA) and machine-based GA (mb-GA) to solve FSSP. These approaches differ in the way to represent the chromosome. Our primary intention is to investigate the effectiveness and efficiency of GAs to solve FSSP. We carried out some numerical experiments using Benchmark test problems to see the performances of the algorithms [10]. The results are comparable with those given by another algorithm called the NBH algorithm [11].

We organize the remainder of this paper as follows: In Section 2, we give a brief overview of the FSSP. Section 3 describes the design strategies of the proposed GA approaches, including the design of chromosome representations, genetic operations, and selection strategy. We present the numerical experiment results and the comparison with other heuristic methods in Section 4. Finally, Section 5 describes the conclusion showing the remarkable effectiveness of the approaches.

2. Flow Shop Scheduling Problem

Flow shop scheduling is one of the problems that s follows: There is a set of *m* number machines and *n* number of jobs. Each job consists of *m* operation(s) that must be processed with a different device. The sequence for processing all jobs in the *m* machine(s) is the same. t_{ij} (i =1, …, n; j =1, …, m) denote the processing time of job *i* by using machine *j*. For FSSP, we have the following assumption:

- Every job has to be processed on all machines in the order j = 1, 2, …., m.
- Each machine processes just one job at a time.
- Operations are not preemptive.
- The processing times include Set-up times for the operations.
- Operation sequences of the jobs are the same on every machine.

The usual objective function is to determine the schedule (the processing job sequence on the machine or machine sequence to process jobs) with minimum Makespan. There are also some different objective functions used, i.e., total tardiness, mean flowtime, and so on. We can find the mathematical formulation of FSSP in [12].

3. Design of The GAs

In this section, we describe the GA, which is one of the accurate and efficient heuristic methods to solve hard optimization problems. It was first introduced by Holland [13] and popularised by several researchers, including [14], [15], and [4].

3.1. Initialization

When implementing GA for a specific application, the first step is to find a way to represent the possible problem solution. Here, we applied the permutation-based representation called job-based GA (jb-GA) and machine-based representation (mb-GA). For jb-GA, a list of *n* jobs represents the chromosome. Each job appears once in the list. Thus, this representation will always yield a feasible schedule. The order represents the sequence of the job processed in each machine. We construct the schedule following the order in the list.

Similarly, for mb-GA, the chromosome represents the order of the machine to process each job. It will be a list of *m* machines. We generate the value of each gene in the chromosome randomly. As an example, Figure 1 illustrates a chromosome for the problem ta001 (20 jobs and five machines).

1	5	3	2	4

Figure 1. An example of chromosome representation

3.2. Crossover and Mutation

The purpose of crossover operation is to make replication of the chromosome. It has a vital role in the success of GA. For permutation-based representation, we cannot use simple two-point crossover operations. There are many variants of crossover operations usually used for permutation representation, such as Partially Matched Crossover (PMX), Position-based crossover (PX), and Weight mapping crossover (WMX) [4]. Here, we adopt the PMX crossover as follows:

Procedure: PMX
Step 1: Select a section of chromosome randomly
Step 2: Exchanged each substring
Step 3: Determine the mapping of genes in each substring
Step 4: Update the chromosome with information on Step. 3

Another essential feature of GA is the mutation operation. It is usually done by exchanging the data within a chromosome to prevent premature loss of information. In this paper, we adopt the inversion mutation that selects two positions within a chromosome at random and then inverts the sub-string between these two positions. We illustrate the inversion mutation operation as follows:

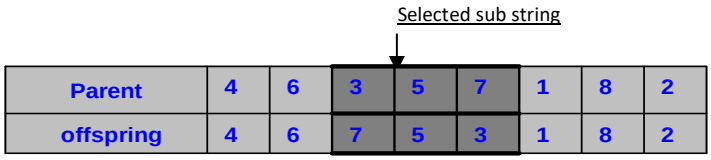

Figure 2. Example of inversion mutation

3.3. Evaluation and Selection

When using GA, we have to assets each chromosome on how well it fits with the problem requirements. Here, we use the makespan as the fitness value. The selection process is also known as an essential step in applying GA. The main objective is to guide in determining the chromosome for the next population. The selection process is done based on the fitness value. There have been many selection strategies reported for various GA applications [5]. We adopt the elitist selection strategy by selecting the best *pop_size* chromosome to the next generation.

4. Experimental Design and Results

There are two purposes of this section: First is to explain the design of the numerical experiments, including the design of the test problems and parameter setting. The second is to evaluate the effectiveness and efficiency of GAs to solve FSSP.

4.1. Design of Test Problems

Several numerical experiments has been done to show the effectiveness and efficiency of the proposed approaches. In our experiments, we used a total of 18 different size Benchmark instances provided by literature [10]. Those problems have the number of jobs 20-100 and the number of machines 5-20. Those approaches were implemented in C++ and run on PC with processor Intel-Core i5. We set the crossover and mutation probabilities as 0.4 and 0.2, respectively.

The population size is varied based on the size of the problems. Table 1 summarizes the overall results of the experiments.

Table 1: Design of numerical experiment and results

No.	Test problems	Number of jobs	Number of machines	max_gen	Optimal	mb-GA	jb-GA	NEH*	time**
1	ta001	20	5	600	1278	1376	1297	1286	25.18
2	ta006	20	5	600	1195	1373	1195	1228	25.18
3	ta011	20	10	600	1582	1716	1592	1680	33.88
4	ta016	20	10	600	1397	1515	1412	1453	33.88
5	ta021	20	20	600	2297	2346	2316	2410	36.83
6	ta026	20	20	600	2226	2302	2230	2349	36.83
7	ta031	50	5	600	2724	2899	2729	2733	34.36
8	ta036	50	5	600	2829	3068	2832	2850	34.36
9	ta041	50	10	700	3025	3459	3098	3146	40.91
10	ta046	50	10	700	3006	3470	3116	3178	40.91
11	ta051	50	20	700	3875	4192	3995	4038	49.56
12	ta056	50	20	700	3698	4080	3829	3918	49.56
13	ta061	100	5	700	5493	5646	5495	5567	47.82
14	ta066	100	5	700	5135	5553	5144	5139	48.8
15	ta071	100	10	800	5770	6366	5842	5848	63.23
16	ta076	100	10	800	5303	5965	5344	5373	63.23
17	ta081	100	20	2000	6286	6974	6456	6661	187
18	ta086	100	20	2000	6437	7074	6661	6761	187

*Nawaz, Encore and Ham (NEH) [11]

**Computational Time of jb-GA (in the second)

The above results that jb-GA outperforms mb-GA on the solution quality, all of the time. It can reach the optimal/near-optimal solutions to the problem. The comparison with the results of the NEH algorithm shows an impressive performance of the jb-GA on the quality of solutions (95 percent). The above results also indicate the reasonable computational time of jb-GA.

In these experiments, we define the error as the percentage of (Obtained Solution – Optimal Solution)/Optimal Solution. The next Figure 3 illustrates the comparison of the errors for each instance.

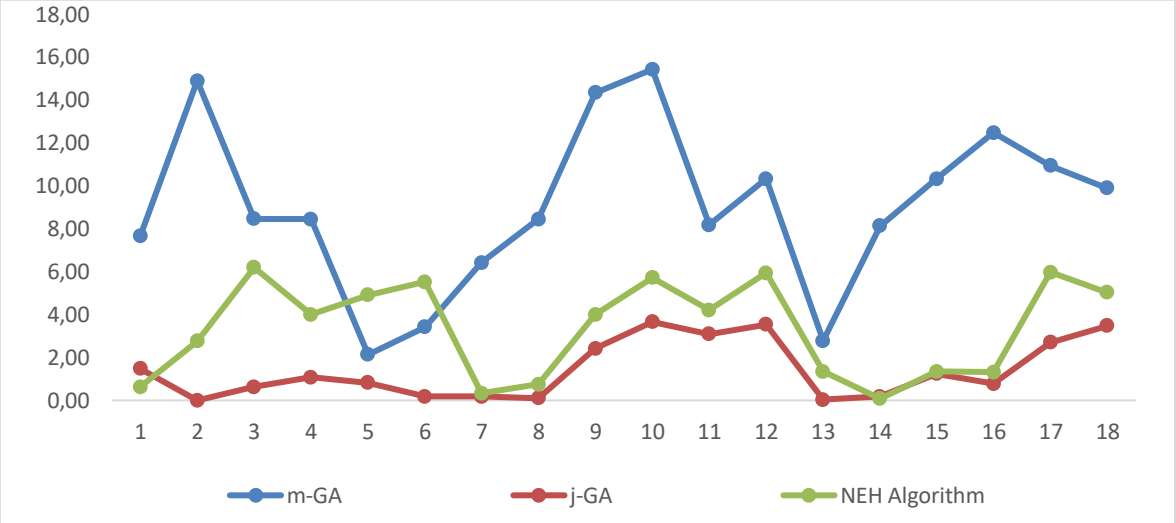

Figure 3. The comparative error of the methods

Like many other heuristic methods, the above results show that jb-GA still has a limitation on the number of instances solved (NIS) optimally. For some significant size problems, it even often reaches the near-optimal solutions. So there is always a place for improvement in the quality of solution or computational cost. Finally, in the next Figure 4, we illustrate the obtained schedule and the convergence of objective value in the generation for test problem ta061.

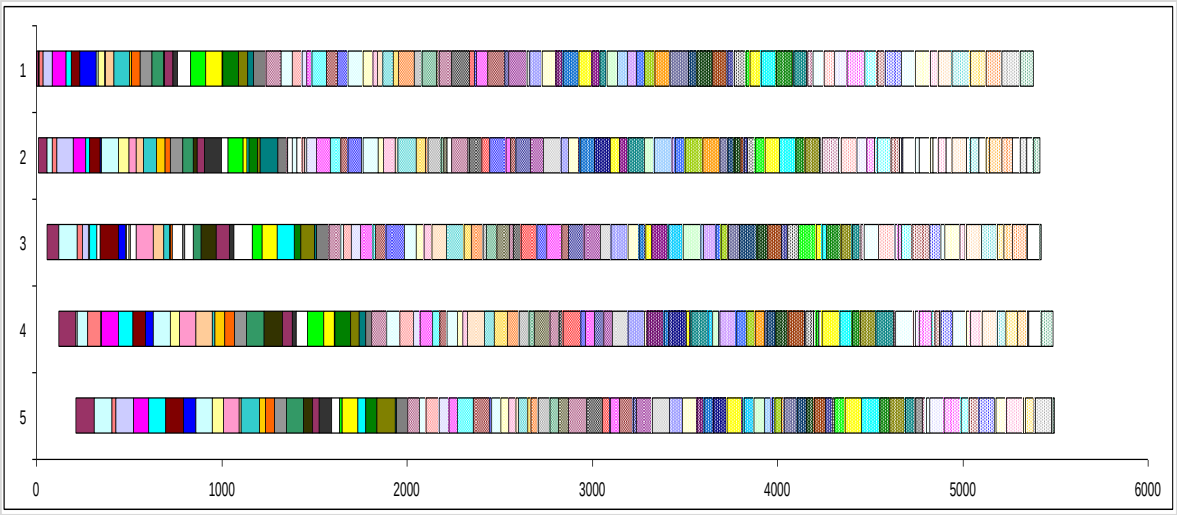

Figure 4. Makespan of the schedule for test problem ta061

5. Conclusions

We have presented two GA approaches called job-based GA (jb-GA) and machine-based GA (mb-GA) to solve FSSP. To demonstrate the effectiveness of the methods, we conducted several numerical experiments. We use the Benchmark scheduling test problems given in the literature. We compare the results with those provided by another heuristic algorithm (NEH algorithm). The results demonstrate that jb-GA has higher accuracy and achieves the optimal solution with reasonable computational time. This finding confirms the usefulness of GA to solve FSSP. The current limitation of this approach, like many other heuristic methods, is that jb-GA still sometimes gives the near-optimal solution.

Acknowledgement
This research has been supported by the Scientific Research Unggulan, Grant-in-Aid for Scientific Research by the Ministry of Research and Higher Education, Lampung University, No. : 2226/UN26.21/PN/2019, Indonesia, 2019.

References

[1] C. Rajendran and D. Chaudhuri, 1993 "An efficient heuristic approach to the scheduling of jobs in a Flow Shop. European," *J. Oper. Res.*, **61**, no. 3, pp. 318-325.,
[2] M. Basseur, F. Seynhaeve, and E. Talbi, 2002 "Design of multi-objective evolutionary algorithms: application to the flow-shop scheduling problem," in *Proc. of the 2002 Congress on Evolutionary Computation*, pp. 2 1151-1156.
[3] P. Jin and S. Kaoping, 2003 "Fuzzy flow-shop scheduling models based on credibility measure," in *Proc. of the 12th IEEE International Conference on Fuzzy Systems*, pp. 139-144,
[4] M. Gen and R. Cheng, 2000 *Genetic Algorithms and Engineering Optimization*, New York: John Wiley & Sons.
[5] M. Gen and R. Cheng, 1997 *Genetic Algorithms and Engineering Design.*
[6] A. Syarif, Y. S. Yun, and M. Gen, 2002 "Study on multi-stage logistic chain network: A spanning tree-based genetic algorithm approach," *Int. J. Comput. Ind. Eng.*, **43**, no. 1-2, pp. 299–314,
[7] A. Syarif and M. Gen, 2003 "Solving exclusionary side constrained transportation problem by using a hybrid spanning tree-based genetic algorithm," *J. Intell. Manuf.*, **14**, no. 3-4, pp. 389–399
[8] M. Gen and A. Syarif, 2005 "Hybrid genetic algorithm for multi-time period production/distribution planning," *Int. J. Comput. Ind. Eng.*, **48**, pp. 799-809,
[9] M. Gen and A. Syarif, 2003 "Double Spanning Tree-Based Genetic Algorithm for Two-Stage Transportation Problem," *Int. J. Knowledge-based, Eng. Syst.*, 7, no. 4, pp. 214-221
[10] E. Taillard, "Benchmarks for the basic scheduling problems., 1993 " *Eur. J. Oper. Res.*, **64**, pp. 278–285
[11] P. Chang, C. Liu, and C. Fan, 2006 "A Depth-First Mutation-Based Genetic Algorithm for Flow Shop Scheduling Problems," in *Proc. of International Conference on Hybrid Information Technology*, pp. 25–32.
[12] M. Seda, 2007 "Mathematical Models of Flow Shop and Job Shop Scheduling Problems," *Int. J. Appl. Math. Comput. Sci.*, **4**, no. 41, pp. 241–246.
[13] J. H. Holland, 1975 *Adaptation in Natural and Artificial Systems*.
[14] D. Goldberg, 1989 *Genetic Algorithm in Search, Optimization, and Machine Learning*, Reading, MA: Addison-Wesley.
[15] Z. Michalewicz, 1989 *Genetic Algorithms + Data Structure = Evolution Program*. New York: Springer-Verlag.