ISSN:1991-8178

Australian Journal of Basic and Applied Sciences

Journal home page: www.ajbasweb.com

Impact of Genetic Algorithm Parameters on its performance for Solving Flow Shop Scheduling Problem

[1]Hani A. Al-Rawashdeh and [2]Hisham O. Al-Rawashdeh

[1]*AL-Hussein Bin TalalUniversity / Faculty of Engineering / Mechanical Engineering Department*
[2]*AL-Hussein Bin Talal University/ Faculty of Engineering / Electrical Engineering Department*

ARTICLE INFO

Article history:
Received 12 October 2014
Received in revised form 26 December 2014
Accepted 1 January 2015
Available online 27 February 2015

Keywords:
Flow Shop Scheduling; Genetic Algorithm; Makespan; Population Size; Crossover Probability and Mutation Probability

ABSTRACT

The primary objective of flow shop scheduling is to obtain the best sequence which optimizes various objectives such as makespan, total flow time, total tardiness, or number of tardy jobs, etc. Due to the combinatorial nature of the flow shop problem (***FSP***) there is a lot of artificial intelligence methods proposed to solve it. The Genetic Algorithm (***GA***), one of these methods, is considered a valuable search algorithm capable of finding a reasonable solution in a short computational time. ***GA***parameters, (population size, crossover probability and mutation probability) give different values that can be combined to give various ***GAs.*** In this paper we investigate the impact of the ***GA*** parameters (population size, crossover probability and mutation probability)on the quality of the ***GA*** solution in solving the flow shop scheduling problems. In this paper*four*population size (***Ps***), *five*crossover probability (***Pc***) and*ten* mutation probability (***Pm***) are investigated. The computational results show that there are significant differences among the investigated parameters on the performance of the proposed ***GA***.

To Cite This Article: Hani A. Al-Rawashdeh and Hisham O. Al-Rawashdeh., Impact of Genetic Algorithm Parameters on its performance for Solving Flow Shop Scheduling Problem. ***Aust. J. Basic & Appl. Sci.,*** *9(5): 510-519, 2015*

INTRODUCTION

FSP is solvable to optimality in polynomial time when number of machines are limited to two, $m = 2$,**(Johnson 1954).** When the ***FSP*** enlarges as including more jobs and machines ($m > 2$), it becomes a combinatorial optimization problem. It is clear that combinatorial optimization problems are NP-hard problem class, and near optimum solution techniques are preferred for such problems**(Betul and Mehmet 2010).**

This paper is organized as follows: the next two subsections are introduction about flow shop scheduling problem, basic genetic algorithm and ***GA*** for ***FSP***. Section 2 presents the literature review of genetic algorithm parameters (population size, crossover probability and mutation probability) in solving ***FSP***. Section 3, present the summary of the most ***GA***parametersused in solving ***FSP***. Section 4 present theexperimental set up, experimental design and test problem. Section 5 presents the computational results.Conclusion is presented in section 6.

Flow Shop Scheduling Problem (FSP):

FSP is one of the most widely studied scheduling problems in literature. It is commonly used as a benchmark for testing new heuristic algorithms, although it can hardly be applied to shop floor sequencing problems found in real production systems**(Duda 2006)**.

In classical ***FSP*** there is a set of n jobs, $(1, 2, \dots, n)$, to be processed in a set of m machines,$(1, 2, \dots, m)$, in the same order, i.e. first in machine 1 then on machine 2 and so on, until machine m. The objective is to find a sequence for the processing of all the jobs in the machines to optimize a given criterion. In the literature, the most common criterion is the minimization of the total completion time or **makespan** of the schedule (C_{max}); this is sometimes referred to as maximum flow time (F_{max}). The processing times needed for the jobs on the machines are noted asp_{ij}. Where$i = 1, 2, \dots, n$and $j = 1, 2, \dots, m$. For instance, Figure 1 shows a sequence of ***7*** jobs through ***3*** machines.

Jobs Machines:

Corresponding Author: Hani A. Al-Rawashdeh, AL-Hussein Bin TalalUniversity / Faculty of Engineering / Mechanical Engineering Department
E-mail: hani_raw@yahoo.com.

Australian Journal of Basic and Applied Sciences, 9(5) March 2015, Pages: 510-519

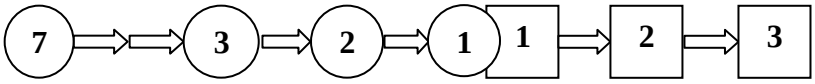

Fig. 1: an example of 7 jobs sequence and 3 machines.

The FSP entails a number of assumptions:

- All jobs are independent and available for processing at time 0.
- Machines are continuously available (no breakdowns).
- Each machine can only process one job at a time.
- Each job can be processed only on one machine at a time.
- Once the processing of a given job has started on a given machine, it cannot be interrupted and processing continues until completion (no preemption).
- Setup time and transportation time of the jobs are sequence independent and are included in the processing times, or ignored.
- In-process inventory is allowed. If the next machine on the sequence needed by a job is not available, the job can wait and joins the queue at that machine.
- The processing times of the jobs at the machines are known in advance.

Basic Genetic Algorithm (GA):

GA was developed by Holland in 1975 (Holland 1975). It is a search technique based on the mechanics of natural genetics and survival of the fittest. The ***GA*** object determines which individuals should survive, which should reproduce, and which should die. Since ***GAs*** are adaptive and flexible, they were shown to be successfully applied to several optimization problems (references).The basic steps of GA are as follows(Tyagi and Varshney 2012):

1. *Initialize a population of binary or non-binary chromosomes.*

2. *Evaluate each chromosome in the population using the fitness function.*

3. *Select chromosome to mate (reproduction).*

4. *Apply genetic operators (crossover and mutation) on chromosome selected.*

5. *Put chromosomes produced in a temporary population.*

6. *If the temporary population is full, then go to step 7. Otherwise;go to step 3.*

7. *Replace the current population with the temporary population.*

8. *If termination criterion is satisfied, then quit with the best chromosome as the solution for the problem. Otherwise, go to step 2.*

A GA for the FSP:

In ***GA*** method, the encoding, crossover and mutation process will give different forms. The different forms of crossover and mutation process in ***GA*** method can be combined to give various ***GAs*** that can be used to get an optimal or near optimal solution of the ***FSP***in reasonable CPU time. The following are generic steps for FSP GA:

*Step 1. Based on the later review, the permutation encoding is adapted for all **FSP** genetic algorithms. That is, [3421 5],a chromosome, represents a job sequence where job 3 is processed first, and then job 4 is processed, and so on.*

*Step 2. In order to find the optimal solution of the problem, standard **GA** starts from a set of assumed or randomly generated chromosomes called initial population with size **Ps**, a set of solutions (chromosome) over sequence of generation.*

Step 3. Each chromosome in the population is evaluated based on fitness criterion.

*Step 4. Check termination criterion, **Tc**, if happening, stop and get the best solution. Else, reproduce a new population as follows:*

*Step 5. Two parent strings are drawn from the population according to a selection method, **Sm**, for reproduction. The number of copies reproduced by an individual parent is expected to be directly proportional to its fitness value.*

*Step 6. Crossover method, **Cm**, is used with probability **Pc** to recombine the two selected parents to get better offspring.*

*Step 7. Mutation method, **Mm**, is applied with probability **Pm** on the offspring generated by the **Cm**. It helps to preserve a reasonable level of population diversity.*

Step 8. *If the new population generated completed, go to Step#3. Else, go to Step#5.*

Literature Review:

Chen *et al*. 1995 generated a ***GA*** based heuristic for ***FSP*** with makespan criterion, in which the initial population was generated by ***CDS*** (Campbell *et al*.1970) and ***RA*** (Dannenbring's 1977). 200 problems were generated for 20 different combinations of job size and number of machines, $\boldsymbol{n} \in \{7, 10, 15, 25\}$, and $\boldsymbol{m} \in \{4, 5, 8, 10, 15\}$. According to generated results of trial examples, the ***GA*** default parameters are proposed as ***Ps=60***,***Pm=0***, ***Pc=1***, and ***20*** generation as a termination criteria. The computational results of the heuristic on a SUN workstation compared with the results of two existing heuristics, SPIRIT (Widmer and Hertz 1989) and Ho and chang's heuristic (Ho and Chang 1991).

Reeves 1995proposeda ***GA*** for finding the minimum makespan of ***n***-job, ***m***-machine permutation ***FSP***. The initial population is

randomly generated. The default ***GA*** parameters used were ***Ps=30***, ***Pc=1***, ***Pm=0.8***.On Taillard benchmarks problems (Taillard 1993), the performance of the algorithm is compared with that of a native neighborhood search technique and with a simulated annealing(***SA***) algorithm.

Murata *et al*. 1996 applied a***GA*** with an objective of minimizing the makespan, and examined two hybridizations of the ***GA*** with other search algorithms. As test problems, they randomly generated 100 ***FSP*** with 20 jobs and 10 machines and 50 jobs and 10 machines. The initial population is randomly generated.Using simulations on the test problems, they found that the following specifications worked best (***Ps=10***, ***Pc=1.0,***and***Pm=1.0***). Based on the default ***GA*** specification, the authors compared the ***GA*** with three search algorithms, local search (***LS***), tabu search (***TS***) and simulated annealing(***SA***).

Reeves and Yamada 1998 proposed a***GA*** to minimize the makespan. The algorithm characteristics are as follow: randomly generated initial population,***Ps=15*** and ***Pc=0.5.*** Thecomputational are done on Taillard benchmark problems (Taillard 1993).

Tang and Liu 2002proposeda ***GA***for ***FSP*** with the objective to minimizing mean flow time. Two new operations are introduced into the algorithm to improve the general ***GA*** procedure. One replaces the worst solutions in each generation with the best solutions found in previous generation. The other improves the most promising solution through local search. Their ***GA*** use the following parameters, ***Ps=80, Pc=0.99, Pm=0.12***, and maximum generation (***Gmax***) ***=1000***. To evaluate the performance of the proposed ***GA***, Computational experiments were carried out on anumber of randomly generated problem instances, $\boldsymbol{n} \in \{50, 75,100,125,150\}$ and $\boldsymbol{m} \in \{5, 10, 15, 20\}$. Two heuristic algorithms, Ho and Chang (Ho and Chang 1991) and Rajendran (Rajendran 1993) and a basic ***GA*** were used for comparison.

Zdansky and Pozivil 2002 proposed a combination of genetic/tabu search algorithm (***GATS***) for ***HFSP*** to minimize the makespan, and compared its performance with pure ***TS*** and pure ***GA***. The test problems are sets of randomly generated with $\boldsymbol{n}/\boldsymbol{m} \in \{10/5, 12/8, 15/10\}$. They used genetic components similar to ***GA*** described in (Stluka 1999).The algorithm was tested against pure ***TS*** and pure ***GA***.

Eliter *et al*. 2004 proposed a***GA*** -based heuristic for the ***FSP*** with makespan criterion. They used $p_s - 1$ schedules produced by ***CDS***(Campbell *et al*.1970) method and one schedule produced by Dannenbring's method (Dannenbring 1977) to generate the initial population. Based on (Chen *et al*. 1995), the authors used ***Ps=60,Pc=1.0*** and***Pm=0.05***. Based on trial examples, they used ***20***generations (***Gmax***) as the termination criterion. In order to examine the effectiveness of the proposed***GA***, the performance of the algorithm is compared over 230 generated problems forming 23 different combination of jobs and machines (***n/m***= 8/5, 8/10, …,35/35, 40/40) with the ***NEH***algorithm (Nawaz *et al*. 1983).

Iyer and Saxena 2004 proposed a***GA*** for the permutation ***FSP*** with the objective of minimizing the makespan. They redesigned the standard ***GA*** implementation by using structural information from the problem. They considered five different problem dimensions, $\frac{\boldsymbol{n}}{\boldsymbol{m}} \in \{\frac{10}{10}, \frac{20}{10}, \frac{49}{15}, \frac{60}{25}, \frac{100}{40}\}$. They used two methods to simulate the matrices of processing times, one using uniform distributions and the other using normal distributions. The initial population is randomly generated. They used the following parameters based on (Bagchi and Deb 1996), ($\boldsymbol{Ps} \in \{20, 100\}$, $\boldsymbol{Pc} \in \{0.7, 0.9\}$ and $\boldsymbol{Pm} \in \{0.005, 0.05\}$. The termination of their algorithm is after 300 iterations.

Ruiz *et al*. 2006 proposed a hybrid genetic algorithm (***HGA***) that uses a simple form of local search based on the ***NEH*** heuristic(reference). The objective is to minimize the makespan. In the algorithm, the initial population is generated as follows: first individual based on the standard ***NEH***, up to B% based ona modified ***NEH,*** and the remaining 100-B% randomly. The proposed experimental parameters are as follow: $\boldsymbol{Ps} \in 20, 30, 40, 50\}$, $\boldsymbol{Pc} \in \{0.0, 0.1, 0.2, 0.3, 0.4\}$ and $\boldsymbol{Pm} \in \{0.0, 0.005, 0.01, 0.015\}$. For the evaluation they used Taillard's standard benchmark (Taillard 1993) and compared the results against 11 published methods.

Sadegheih 2006 proposed a ***GA***. The algorithm is tested on 8-jobs and 7-machines example for each of the combinations of the following parameter level: $\boldsymbol{Ps} \in \{20, 60, 100\}$,$\boldsymbol{Pm} \in \{0.001, 0.005, 0.01, 0.02\}$, and $\boldsymbol{Pc} \in \{0.1, 0.2, 0.6, 0.8, 0.9\}$. The author found that the proposed GA performance is sensitive mutation probability within a good range of values [0.005:0.01], however, outside the good range the ***GA*** performance deteriorates in crossover probability, the performance is insensitive to crossover rate in range [0.1:0.9] but operator is necessary.The performance is sensitive to population size in range [20:100].

Wang *et al*. 2006 proposed a Hybrid Genetic Algorithm (***HGA***) for permutation ***FSP*** with limited buffers with the objective to minimize the makespan. In the ***HGA***, the local search based on graph model is employed. The initial population is generated as follows: one chromosome using the standard ***NEH*** heuristic and $\boldsymbol{Ps} -$

1 chromosomes generated randomly. They used ***Ps=50,*** $\boldsymbol{Gmax = 20}n * m$ **.**To show the effectiveness of the ***HGA***, they compared the ***HGA*** with the simple ***GA*** and ***TS*** and ***SA*** with benchmarks of different scales of (Carlier 1978, Reeves 1995 and Taillard 1993).

Octavia *et al*. 2007 discussed the application of ***HGA*** to solve practical***FSP***. The ***HGA*** was run on the following sets of parameters: ***Ps=20***, (***Gmax***) ***=2000***, and ***Pm=0.01***. The initial population is generated as 1 chromosome by NEH, 1 chromosome by ascending order of job time, 1 chromosome by descending order of job time, and the rest randomly. Practical case was used as a test case to evaluate the performance of the proposed algorithm. Each test problem consisted of ***120*** jobs including type of product and number of unit would be produced. The proposed algorithm was compared to ant colony, genetic-tabu.

Adusumilli *et al*. 2008 proposeda ***GA*** for two machines ***FSP***to minimize some of finishing time of arbitrary number of jobs.The default parameters are set as ***Ps=50***, (***Gmax***) ***=500***, ***Pc=0.85,Pm=0.01***, and the initial population is randomly. The proposed ***GA*** is tested on 20-jobs, 2-machines problemand compared with branch and bound technique and Johnson algorithm.

Kahraman *et al*. 2008 proposed a ***GA*** for ***HFSP*** with the objective of minimizing makespan. They gave a complete evaluation of the different parameters and operator of the algorithms using the following experiment: ***Ps=25***, $\boldsymbol{Pc} \in \{(0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$, and $(\boldsymbol{Pm}) \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$. The proposed algorithm is tested on Carlier and Neron's benchmark problems (Carlier and Neron's 2000). The authors found that the best operators set are: $(\boldsymbol{Pc}) = \{0.1, 0.2, 0.3\}$, and $(\boldsymbol{Pm}) \in \{0.1, 0.2\}$. The results are compared with other methods that test the same problems.

Kim and Jeong 2009 proposed a flow shop scheduling with no-wait flexible lot streaming (FSS-nwFLS) using adaptive ***GA***to minimize the makespan.The proposed ***GA*** use randomly generated initial population, ***Ps =50, Pc=0.5, Pm=0.5*** and ***Gmax=1500.*** They run ***14*** type of problems considering the number of products (5, 10, 15, 20, 25, 30, 50), the number of sub-lots (15, 25, 30, 45, 50, 60, 65, 75, 90, 100, 125, 150, 250). The results of the proposed GA are compared with other two traditional ***GAs***.

Engin *et al*. 2011 proposed a***GA*** based on a permutation representation of the n jobs of ***HFSP*** with multiprocessor task problems to minimize makespan. They proposed the following experiment: randomly generated initial population , $\boldsymbol{Ps} \in \{15, 25, 40, 50\}$, $(\boldsymbol{Pc}) \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$, $(\boldsymbol{Pm}) \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$, stopping criterion: (***Gmax***) = 10000, 25 min.C***PU*** time).The proposed approach was tested on a set of 240 problems (Oguz 2006, Oguz and Ercan 2005 and Kahraman *et al*. 2010).

Navala 2011proposed a review of using a ***GA*** based approaches in scheduling of ***FMS*** (flexible manufacturing system). They reviewed the most of the research papers that contributed to scheduling of ***FMS*** using ***GAs***. They found that, in some contributions considered only one single criterion and in some multi criterion. They also observed that use of ***GA*** in integration with other meta heuristics like ***TS***, ***SA***, neural networks to determine the optimized schedule in an ***FMS***.

Verma and Dhingra 2011described multiprocessor task scheduling in the form of permutation ***FSP***, which has an objective function for minimizing the makespan. They proposed the following GA: randomlygenerated initial population, ***Ps=100***,***RWS***,***Pc=0.6, Pm=0.0*** and***Gmax=100*** as a stopping criterion. They solved an example of 15-jobs and 4-processors.

Chen *et al*. 2012 proposed a self-guided ***GA*** for permutation to minimize FSP's makespan. In the proposed algorithm they used ***TTS***,***Ps=100***, ***Pc=0.6***and***Pm=0.3***. They conducted extensive computational to compare the self-guided ***GA*** with several other algorithms using the 110 Taillard instances (Taillard 1993).

Summary:

Table 1 shows the summary of different ***GAs*** specifications used in the reviewed literaturesuch as population size, crossover and mutation probabilities. Also, we can notice that the following ranges: $\mathbf{Ps} \in [\mathbf{5 - 100}]$ **,** $\boldsymbol{Pc} \in [\mathbf{0 - 1.0}]$, $\boldsymbol{Pm} \in [\mathbf{0 - 1.0}]$,and differentproblem sizes ranging from $\boldsymbol{n \in [8 - 500]}$ and $\boldsymbol{m \in [2 - 40]}$.We used***6***selection methods, ***17*** crossover operators and ***8***mutation operators used in designing different genetic algorithms for solving ***FSP*** as shown in Table 2, 3 and 4 respectively.

Table 1: summary of the genetic algorithms used for solving FSP

Author/s	*Year*	*Criterion*	*Ps*	*Pc*	*Pm*	*Max. no. of Gen. (Gmax)*	*Maximum size of problems (n, m*	*System Specification*
Chen *et al*.	1995	makespan	60	1.0	0.0	20	(25, 15)	Programmed using Fortran 77 & run on a SUN 4/490 Workstation.

Author/s	*Year*	*Criterion*	*Ps*	*Pc*	*Pm*	*Max. no. of Gen. (Gmax)*	*Maximum size of problems (n, m*	*System Specification*
Reeves	1995	makespan	30	1.0	0.8	---	(75,20)	Programmed in Pascal, & run on a Sequent S82 Computer
Murata *et al*.	1996	makespan	5 10* 20 30 40 50	0.5 0.6 0.7 0.8 0.9 1.0*	0.5 0.6 0.7 0.8 0.9 1.0*	5000	(50.10)	---
Reeves and Yamada	1998	makespan	15	0.5	---	700	(200,20)	Programmed in C, and implemented on DEC Alpha 600 5/226 Computer.
Tang and Liu	2002	mean flow time	80	0.99	0.12	1000	(150,20)	Pentium PC Computer ---
Zdansky and Pozivil	2002	makespan	---	---	---	100	(15,10)	Programmed in Pascal, and run on a Pentium IV(256 MB) Computer
Eliter *et al*.	2004	makespan	60	1.0	0.05	20	(40,40)	Programmed in C, and run on PCs Pentium Processors
Iyer and Saxena	2004	makespan	20 100*	0.9* 0.7	0.05* 0.005	300	(100,40)	Programmed in (Delphi 7.0), and tested on Intel Pentium IV Processor running at 2.8 GHz with 512 MB
Ruiz *et al*.	2006	makespan	20* 30 40 50	0.0 0.1 0.2 0.3 0.4*	0.0 0.005 0.01* 0.015	25* 50 75	(500,20)	Implemented in Turbo C++
Sadegheih	2006	makespan	20 60 100	0.1 0.2 0.6 0.8 0.9	0.001 0.005* 0.01* 0.02	900	(8,7)	Coded in C++ and run on a PC with AMD Athlon 1.0 GHz CPU Implemented in Microsoft Visual Basic 6.0
Wang *et al*.	2006	makespan	50	---	0.2	2000	(100,20)	Coded in a C++ & developed at MIT & run under GNU g++ Complier
Octavia *et al*	2007	makespan	20	---	0.01	2000	n=120 factory	Implemented in Borland Delphi and run on a PC P4 Processor with 3 GHz, 512 MB
Adusumilli *et al*.	2008	makespan	50	0.85	0.01	500	(20,2)	Implemented in the Java using an IBM P 1.4 GHz Computer with 512 MB
Kahraman *et al*. Kim and Jeong	2008	makespan	25	0.1* 0.2 0.3 ,..., 0.9 1.0	0.1* 0.2 0.3 ,…, 0.9 1.0	72000	(15,10)	Implemented in Borland Delphi and run on a PC P4 Processor with 3 GHz, 512 MB

Author/s	*Year*	*Criterion*	*Ps*	*Pc*	*Pm*	*Max. no. of Gen. (Gmax)*	*Maximum size of problems (n, m*	*System Specification*
Engin *et al*.	2009	makespan	50	0.5	0.5	1500	(50,5)	Implemented using MATLAB at command line
Verma and Dhingra	2011	makespan	15 25 40 50*	0.1 0.2 0.3 ,…, 0.9* 1.0	0.1* 0.2 0.3 ,…, 0.9 1.0	10000	(100,8)	Pentium 4 with 3GHz Processor and 512 MB.
Chen *et al*.	2011	makespan	100	0.6	---	100	(15,4)	---
Tyagi and Varshney	2012	makespan	100	0.6	0.3	300	(200,20)	
	2012	makespan	60	10	0.05	20	(40,20)	

*Shows the best GA parameters (Ps, Pc and Pm).

Table 2: selection methods used in FSP.

#	Selection Methods (Sm)	Codes
1	Random Selection	RMS
2	Roulette Wheel Selection	RWS
3	Rank Selection	RKS
4	Tournament Selection(tour size 2)	TTS2
5	Tournament Selection(tour size 3)	TTS3
6	Tournament Selection(tour size 4)	TTS4

Table 3: Crossover operators methods used in FSP.

#	Crossover OperatorMethods(Cm)	Codes
1	One-Point Crossover	1PX
2	Two-Point Crossover Version1	2PXV1
3	Two-point Crossover Version2	2PXV2
4	Two-Point Crossover Version3	2PXV3
5	Position Based Crossover Version1	PBXV1
6	Position Based Crossover Version2	PBXV2
7	Linear Order Crossover	LOX
8	Partially Mapped Crossover	PMX
9	Longest Common Subsequence Crossover	LCSX
10	Order Crossover	OX
11	Order Based Crossover	OBX
12	Cycle Crossover	CX
13	Similar Block Order Crossover	SBOX
14	Similar Job Order Crossover	SJOX
15	Order-Based Crossover	OBX
16	Maximal Preservative Crossover	MPX
17	Alternating Position Crossover	APX

Table 4: Mutation operator methods used in FSP

#	Mutation Operator Methods (Mm)	Codes
1	Adjacent Two-Job Change Mutation	AD2JM
2	Arbitrary Two-Job Change Mutation	AR2JM
3	Arbitrary Three-Job Change	AR3JM
4	Shift Mutation	SHM
5	Inversion Mutation Version1	INMV1
6	Inversion Mutation Version2	INMV2
7	Displacement Mutation	DM
8	Scramble mutation	SCM

Table 5: Genetic algorithm Parameters

Genetic algorithm Parameters	Values			
Crossover Probabilities (*Pc*)	0.3, 0.4, 0.5, 0.6, 0.7			
Mutation Probabilities (*Pm*)	0.001, 0.005, 0.01, 0.02, 0.05, 0.07, 0.1, 0.15, 0.2, 0.3			
Population Size (*Ps*)	10	20	30	50
Max. number of generation	1000	500	333	200

Experimental Set up:

We built a genetic algorithm based on the steps mentioned in the subsection (1.3) using ***C*** sharp language (Microsoft Visual Studio 2010). The six selection methods mention in Table 2, the seventeen crossover method mention in Table 3, and the eight mutation method mentioned in Table 4 are coded in the proposed ***GA***.

Experimental design:

We design full factorial experiment as follows:

1. Based on Tables 2, 3, and 4 we have the following: 6 selection methods, 17 crossover methods and 8 mutation methods.
2. The ***GA*** depends also on the following other parameters: population size (***Ps***), crossover probability (***Pc***) and mutation probability (***Pm***). We propose values for these parameters as shown in Table 5. Based on Table 5 we have the following: 4 *Ps*, 10 *Pm*, and 5 *Pc*.

Therefore, the total number of combination = 6 Sm * 17 Cm * 8 Mm* 4 Ps * 5 Pc * 10 Pm = 163200.

Test problem:

We test every combination of the experiment on the first Taillard problem, ta001. The problem consists of 20 jobs and 5 machines. The processing time matrix is drown from uniform distribution [1, 99]. The upper bound of this problem is 1278. The problem data and information can be downloaded from the OR Library [(http://people.brunel.ac.uk/~mastjjb/jeb/orlib/files/flowshop1.txt)]

Computational results:

All computational results have been obtained on a Core 2 Duo 2.0 GHz personal computer. Each combination is run 10 times. Then, the best, average and standard deviation of makespan for the 10 are obtained.

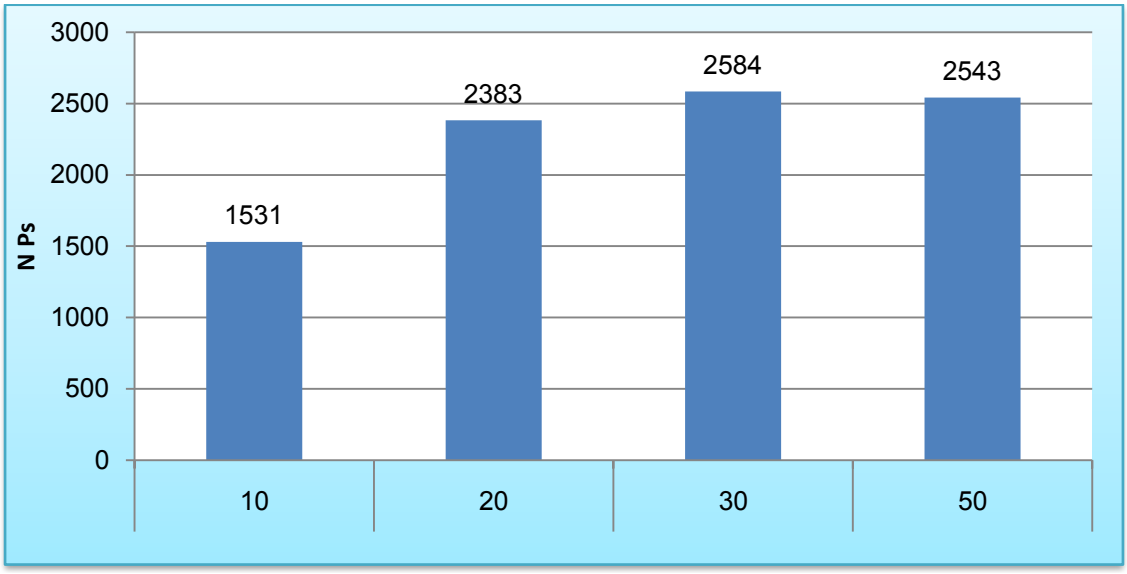

Fig. 2: Population size.

Impact of the population size (Ps):

The contribution of the four population sizes $\boldsymbol{Ps} \in \{10, 20, 30 \text{ and } 50\}$ is determined in the figure below.

Figure 2 shows the distribution of the 9041 combinations on the population size. Applying the chi square test give (P-value equals 0.0000) which means significant differences among the population size. From the figure 2 it shows that the population size ($\boldsymbol{Ps = 30}$) gave best results than other population sizes, then the ($Ps = 50$ and 20. respectively). and the ($Ps = 10$) gave worst results.

5.2 Impact of crossover probability (Pc):

The contribution of the five crossover probabilities $\boldsymbol{Pc} \in \{0.3, 0.4, 0.5, 0.6 \text{ and } 0.7\}$ is determined in the figure below.

Figure 3 shows the distribution of the 9041 combinations on the crossover probability. Applying the chi square test give (P-value equals 0.0000) which means significant differences among the crossover probability. From the figure 3 it shows that

the crossover probability ($\boldsymbol{Pc = 0.6}$) gave best results than other crossover probabilities, then the crossover probabilities ($Pc = 0.4, 0.7$ and 0.5 respectively) . And the crossover probability ($Pc = 0.3$) gave the worst results.

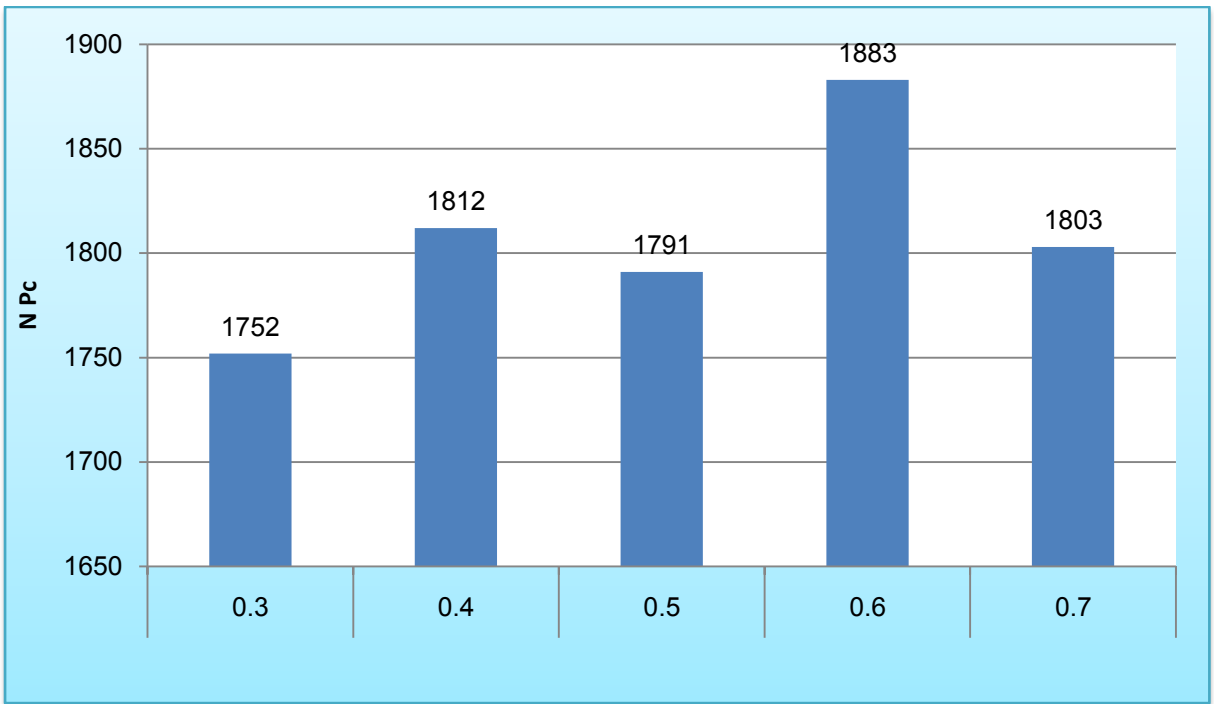

Fig. 3: Crossover probabilities.

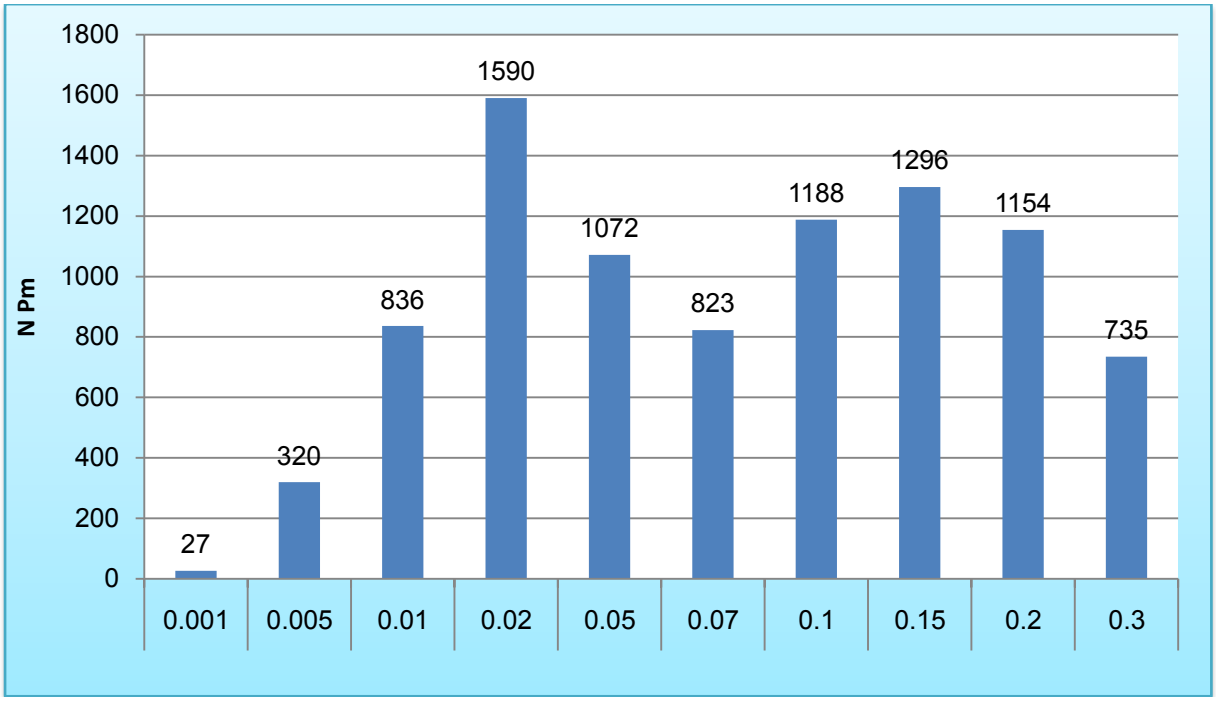

Fig. 4: Mutationprobabilities.

5.3 Impact of the mutation probability (Pm):

The contribution of the ten mutation probabilities

$\boldsymbol{Pm} \in \{0.001,\ 0.005,\ 0.01,\ 0.02, 0.05, 0.7, 0.1, 0.15, 0.2$ and $0.3\}$ is determined in the figure 4.

Figure 4 shows the distribution of the 9041 combinations on the mutation probability. Applying the chi square test give (P-value equals 0.0000) which means significant differences among the

mutation probability. From figure 4 below it shows that the mutation probability ($\boldsymbol{Pm = 0.02}$) gave best results than other mutation probabilities, then the mutation probabilities ($P\ m = 0.15, 0.1, 0.2$ and 0.05 respectively). And the mutation probabilities gave worst results when ($Pm =$ $0.001, 0.005, 0.3, 0.07,$ and 0.01 respectively).

Table 6: Default values of GA parameters.

Default Values	*Type / Value*
Population size (*Ps*)	*30*
Crossover Probability (*Pc*)	*0.6*
Mutation Probability (*Pm*)	*0.02*

Conclusions:

From figures 2, 3 and 4 above, we find the default values (best ***GA***parameters) (populationsize, crossover probability and mutation probability). (See *Table 6*)

It is clear that ***FSP*** problem still needs a further work such as a comparative study for all the above mentioned ***GA*** operators and parameters to see their impact on the quality of the problem solution. Moreover, the impact of problem size on the ***GA*** specification is needed.

REFERENCES

Adusumilli, K., D. Bein and W. Bein, 2008. *"A genetic algorithm for two machine flow shop problem"*. Proceeding of the Hawaii International Conference on System Science, 41: 1-8.

Bagchi, T.P. and K. Deb, 1996. *"Calibration of GA parameters: the design of experiment approach"*.Computer Science and Informatics, 26(3): 46-56.

Betul Yagmahan and Mehmet Mutlu, 2010. *"A multi-objective ant colony system algorithm for flow shop scheduling problem"*. Expert Systems with Applications, 37: 1361-1368.

Campbell, H.G., R.A. Dudek and M.L. Smith, 1970. *"A heuristic algorithm for the n-job m-machine sequencing problem"*. Management Science, 16(10): 169-174.

Carlier, J., 1978. *"Ordonnancements a contraintesdisjonctives"*. R.A.I.R.O Recherecheoperationelle /Operations Research, 12: 333-51.

Carlier and E. Neron, 2000. *"An exact method for solving the multiprocessor flow shop"*. R.A.I.R.O-R.O., 34: 1-25.

Chen, C.L., V.S. Vempati and N. Aljaber, 1995. *"An application of genetic algorithms for flow shop problems"*. European Journal of Operational Research, 80: 389-396.

Chen, S.H., P.C. Chang, T.C.E. Cheng and Q. Zhang, 2012. *"A self-guided genetic algorithm for permutation flowshop scheduling problems"*. Computer and Operations Research, 39: 1450-1457.

Dannenbring, D.G., 1977. *"An evaluation of flow shop sequencing heuristics"*. Management Science 23(11): 1174-1182.

Duda, J., 2006. *"Local search and nature based metaheuristic. A case of flow shop scheduling problem"*. Proceedings of ISSN International Multiconference on Computer Science and Information Technology, pp: 17-24.

Engin, O., G. Ceran and K. Mustafa, 2011. *"An efficient genetic algorithm of hybrid flow shop scheduling with multiprocessor task problems"*. Applied Soft Computing, 11: 3056-3065.

Etiler, O., B. Toklu, M. Atak and J. Wilson, 2004. *"A genetic algorithm for flow shop scheduling problems"*. Journal of the Operational Research Society, 55: 830-835.

Goldberg, D. and R. Lingle, 1985. *"Alleles, loci and the traveling salesman problem"*. In Proceedings of the International Conference on Genetic Algorithms and Their Applications, Pittsburg, USA, pp: 154-159.

Holland, J.H., 1975. "*Adaptation in natural and artificial systems"*. University of Michigan Press, Ann Arbor.

Ho, J.C. and Y.L. Chang, 1991. *"A new heuristic for the n-job, m-machine flow shop problem"*. European Journal of Operational Research, 52: 194-202.

Iyer, K. and B. Saxena, 2004. *"Improved genetic algorithm for permutation flowshop scheduling problem"*. Computers and Operations Research, 31: 593-606.

Johnson, S.M., 1954. *"Optimal two-and three-stage production schedules with setup times included"*. Naval Research Logistics Quarterly, 1: 61-68.

Kahraman, C., O. Engin, I. Kaya and M.K. Yilmaz, 2008. *"An application of effective genetic algorithms for solving flow shop scheduling problems"*. International Journal of Computational Intelligence System, 1(2): 134-147.

Kahraman, C., O. Engin, I. Kaya and R.E. Ozturk, 2010. *"Multiprocessor task scheduling in multistage hybrid flow shop: a parallel greedy algorithm approach"*. Applied Soft Computing, 10: 1293-1300.

Kim, K. and J. Joeng, 2009. *"Flow shop scheduling with no-wait flexible lot streaming using adaptive genetic algorithm"*. International Journal Advance Manufacturing Technology, 44:

1181-1190.

Murata, T., H. Ishibuchi and H. Tanaka, 1996. *"Genetic algorithms for flow shop scheduling problems"*. Computers & Industrial Engineering, 30(4): 1061-1071.

Naderi, B. and R. Ruiz, *"The distributed permutation flow shop scheduling problem"*. Computers and Operations Research, 37(4): 754-768.

Navala, H., 2011. *"Use of genetic algorithm based approaches in scheduling of FMS: A Review"*. International Journal of Engineering Science and Technology, 3(3): 1936-1942.

Nawaz, M., E. Enscore and I. Ham, 1983. *"A heuristic algorithm for the m-machine n-job flow shop sequencing problem"*. Omega, 11: 91-95.

Octavia, T., I.H. Sahputra and J. Soewanda, 2007. *"Robust-hybrid genetic algorithm for the flow-shop scheduling problem"*. Journal TeknikIdustri, 9(2): 144-155.

Oguz, C. and F. Ercan, 2005. *"A genetic algorithm for hybrid flow shop scheduling with multiprocessor tasks"*. Journal of Scheduling, 8: 323-351.

Oguz, C., 2006. *"Data for hybrid flow shop scheduling with multiprocessor tasks"*. Koc University. Available from: http://home.ku.edu.tr/coguz/.

Rajendran, C., 1993. *"Heuristic algorithm for scheduling in a flow shop to minimize total flow time*". International Journal of Production Economics, 29: 65-73.

Reeves, C.R., 1995. *"A genetic algorithm for flowshop sequencing*". Computers and Operations Research, 22(1): 5-13.

Ruiz, R. and C. Maroto, 2005. *"A comprehensive review and evaluation of permutation flow shop heuristics"*. European Journal of Operational Research, 165: 479-494.

Ruiz, R., C. Maroto and J. Alcaraz, 2006. *"Two new robust genetic algorithms for the flowshop scheduling problem"*. OMEGA, The International Journal of Management Science, 34: 461-476.

Sadegheih, A., 2006. *"Scheduling problem using genetic algorithm, simulated annealing and the effects of parameter values on GA performance"*. Applied Mathematical Modelling, 30: 147-154.

Stluka, P., 1998. *"Use of artificial intelligence for scheduling of batch operations*". Ph.D. thesis at VSCHT Praha.

Stutzle, T., 1998. *"Applying iterated local search to the permutation flow shop problem".* Technical Report, AIDA-98-04, FG Intellektik, TU Darmstadt.

Taillard, E., 1990. *"Some efficient heuristic methods for the flow shop sequencing problem"*. European Journal of Operation Research, 47: 65-74.

Taillard, E., 1993. *"Benchmarks for basic scheduling problems"*. European Journal of Operational Research, 64: 278-285.

Tang, L. and J. Liu, 2002. *"A modified genetic algorithm for flow shop sequencing problem to minimize mean flow time"*. Journal of Intelligent Manufacturing, 13: 61-67.

Tyagi, N. and R.G. Varshney, 2012. *"A model to study genetic algorithm for flowshop scheduling problem"*. Journal of Information and Operations Management ISSN, 3(1): 38-42.

Verna, R. and S. Dhingra, 2011. *"Genetic algorithm for multiprocessor task scheduling"*. International Journal of Computer Science and Management Studies ISSN, 11(2): 181-185.

Wang, L., L. Zhang and D.Z. Zheng, 2006. *"An effective hybrid genetic algorithm for flow shop scheduling with limited buffers"*. Computer and Operation Research, 33: 2960-2971.

Widmer, M. and A. Hertz, 1989. *"A new heuristic for the flow shop sequencing problem"*.European Journal of Operational Research, 41: 186-193.

Wolfgang, B. and H. Bradley, 2002. *"Pitfalls with adaptive methods for combinatorial optimization: The traveling salesman - a case study"*.In Proceedings of the International Conference on Mathematics and Engineering Techniques in Medicine and Biological Sciences (METMBS), 1: 243-248.

Zdansky, M. and J. Pozivil, 2002. *"Combination genetic / tabu search algorithm for hybrid flowshops optimization"*. Conference on Scientific Computing, Czech Republic, pp: 230-236.

[(http://people.brunel.ac.uk/~mastjjb/jeb/orlib/files/flowshop1.txt]