EIJEST

The Egyptian Int. J. of Eng. Sci. and Technology
Vol. 17, No. 1 (Jan. 2014)

IMPACT OF GENETIC ALGORITHM OPERATORS ON ITS PERFORMANCE IN SOLVING FLOW SHOP SCHEDULING PROBLEMS*

Gamal M. Nawara, Adel A. Ibrahim, Raafat H. Elshaer, Hani A. Al–rawashdeh +

Industrial Eng. Dept., Faculty of Eng., Zagazig University, Egypt

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 operators, (selection, crossover and mutation process), give different forms that can be combined to give various GAs.

In this paper we investigate the impact of selection, crossover and mutation process on the quality of the GA solution in solving the flow shop scheduling problems. In this study, four selection methods, seventeen crossover methods and eight mutation methods are investigated. The computational results show that there are significant differences among the investigated methods on the performance of the proposed GA.

KEY WORDS: Flow Shop Scheduling; Genetic Algorithm; Makespan; Selection Methods; Crossover Methods; Mutation Methods.

IMPACT DES OPERATÉURS ALGORITHME GÉNÉTIQUE SUR SA PERFORMANCE POUR LA RÉSOLUTION DE PROBLÉMES D'ORDONNANCEMENT DE FLUX BOUTIQUE

RESUME

L'objectif principal de la boutique de débit horaire est d'obtenir la meilleure séquence qui optimise divers objectifs tels que makespan, le temps d'écoulement total, les retards ou nombre d'emplois tardives, etc. En raison de la nature combinatoire du problème de flow shop (FSP), il est un grand nombre de méthodes d'intelligence artificielle a proposé de le résoudre. L'algorithme génétique (GA), une de ces méthodes, est considéré comme un algorithme de recherche précieux capable de trouver une solution raisonnable dans un temps de calcul court. Opérateurs GA, (sélection, croisement et processus de mutation), donnent des formes différentes qui peuvent être combinés pour donner différents gaz.

Dans cet article, nous étudions l'impact de la sélection, le croisement et processus de mutation sur la qualité de la solution GA dans la résolution des problèmes d'ordonnancement d'atelier d'écoulement. Dans cette étude, quatre méthodes de sélection, dix-sept méthodes de croisement et de mutation huit méthodes sont étudiées. Les résultats des calculs montrent qu'il existe des différences significatives entre les méthodes d'enquêtes sur la performance de la proposition de GA.

MOTS CLES : Flux Ordonnancement D'atelier; Algorithme Génétique; Makespan; Méthodes De Sélection, Les Méthodes Crossover; Méthodes Mutation.

---

* Received: 29/5/2013, accepted: 4/12/2013, Ref. No. 152, (Original paper).
+ Contact author (adel9938@hotmail.com).

1. INTRODUCTION

FSP is solvable to optimality in polynomial time when number of machines are limited to two, $m = 2$,. 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 **[3].**

1.1 Flow Shop Scheduling Problem (FSP)

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.

1.2 A GA for the FSP

GA 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.

It is well known that the GA efficiency depends to a high degree upon the selection of the good genetic algorithm operators and parameters. The different forms of selection, crossover and mutation process in GA method can be combined to give various GAs that can be impact on the quality of the solution. The following are generic steps for FSP GA [20]:

Step 1. Based on the later review, the permutation encoding is adapted for all **FSP** genetic algorithms. That is, [3 4 2 1 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.

2. LITERATURE REVIEW

Chen et al generated a GA based heuristic for FSP with makespan criterion, in which the initial population was generated by CDS and RA [4]. A set of 200 problems were generated for 20 different combinations of job size and number of machines, $n \in \{7,10,15,25\}$ and $m \in \{4,5,8,10,15\}$. According to generated results of trial examples, the GA default operators are proposed as: roulette wheel selection (RWS), and partially mapped crossover (PMX).

Reeves [16] proposed a **GA** for finding the minimum makespan of **n**-job, **m**-machine permutation **FSP**. The initial population is randomly generated. The author used ranking selection (**RKS**), in selecting parent 1 whereas parent 2 is chosen randomly. The default **GA** parameters used were: one point crossover (**1PX**), and two point crossover version2 (**2PXV2**), and two types of mutation, arbitrary two-job change mutation (**AR2JM**), and shift mutation (**SHM**). On Taillard benchmarks problems, 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 [12] 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. They used roulette wheel selection(**RWS**) and examined the following **10** crossover operators: **1PX**, three versions of two point crossover (**2PXV1, 2PXV2, 2PXV3**), two version of position based crossover(**PBXV1,PBXV1**), edge recombination crossover (**ERX**), partially matched crossover (**PMX**), and cycle crossover (**CX**)and the following four mutation operators: adjacent two-job change (**AD2JM**), arbitrary two-job change mutation (**AR2JM**), arbitrary three-job change (**AR3JM**), and shift mutation (**SHM**). They showed that**2PXV2** and **SHM** are effective for this problem. Using simulations on the test problems, they found that the following specifications worked best (, **2PXV2**, and **SHM**). 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**).

Tang and Liu [20] proposed a **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** uses the following operators, **RWS, PMX, SHM**. To evaluate the performance of the proposed **GA**, Computational experiments were carried out on a number of randomly generated problem instances, $\boldsymbol{n} \in \{50, 75, 100, 125, 150\}$ and$\boldsymbol{m} \in \{5, 10, 15, 20\}$.

Eliter et al [8] proposed a**GA** -based heuristic for the **FSP** with makespan criterion. They used p_s-1 schedules produced by **CDS** method and Dannenbring's method to generate the initial population. Based on Chen et al, the authors used **RWS**, linear order crossover (**LOX**) and **SHM**. 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.

Iyer and Saxena [11] 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 **RWS**, **1PX** ,longest common subsequence crossover (**LCSX**),arbitrary two-job change mutation (**AR2JM**), and the following parameters based on Bagchi and Deb [2]. The authors found that the (**LCSX**) dominates the (**1PX**) in most of the simulation runs and it demonstrates an ability to improve even after a large number of iteration, while the (**1PX**) improves very slowly.

Ruiz et al [18] proposed a hybrid genetic algorithm (**HGA**) that uses a simple form of local search based on the **NEH** heuristic. The objective is to minimize the makespan. They have chosen two selection schemes, **RKS** and tournament selection (**TTS**) and one mutation method, **SHM**. They used eight crossover operators, 4 new and 4 from literature, similar job order crossover (**SJOX**), Similar block order crossover" (**SBOX**), partially mapped crossover (**PMX**), order crossover (**OX**), one-point order crossover (**1PX**) and two-point order crossover version2 (**2PXV2**).

Sadegheih [19] proposed a **GA** with the following characteristic: ranking selection (**RKS**), order-based crossover (**OBX**).

Wang et al [22] proposed a Hybrid Genetic Algorithm (**HGA**) for permutation **FSP** with limited buffers with the objective to minimize the makespan. They used **RWS**, four different crossover operators: linear order crossover **LOX**, **PMX**, **1PX**, and non-abel group based crossover (**NAX**) and three mutation operators: **AR2JM**, **INMV1**, and]. **SHM** .

Octavia et al [15] discussed the application of **HGA** to solve practical **FSP**. The **HGA** was run on the following sets of operators: **RWS**, **SBOX** and **SHM**.

Adusumilli et al [1] proposed a **GA** for two machines **FSP** to minimize some of finishing time of arbitrary number of jobs. The proposed **GA** operators are **RWS**, **PMX**, **and AR2JM**.

Kahraman et al [9] proposed a **GA** for **HFSP** with the objective of minimizing makespan. They gave an evaluation of the different parameters and operator of the algorithms using the following experiment: two selection methods: **RWS** and **TTS** with probabilities (0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, and 1.0), six crossover operators (**PBX**, **OX**, **PMX**, **CX**, **LOX** and **OBX**), and five mutation operators: (**AD2JM**, **AR2JM**, **AR3JM**, **SHM** and **INMV1**). The proposed algorithm is tested on Carlier and Neron's benchmark problems. The authors found that the best parameters set are: **RWS** with selection probabilities (0.1, 0.2, and 0.4), **PBX**and **INMV1**.

Kim and Jeong [10] proposed a flow shop scheduling with no-wait flexible lot streaming (FSS-nwFLS) using adaptive **GA** to minimize the makespan. An adaptive **GA** composed of three main steps. The first step is **PBX** of products. The second step is an iterative hill-climbing algorithm to improve the current generation. The last step is the adaptive regulation of the crossover and mutation rates. The proposed **GA** use randomly generated initial population, **RWS.** 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 [7] 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. Selection methods: **RWS** and **TS** with probabilities Є $\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$, six crossover operators: **(PBX**, **OX**, **PMX**, **CX**, **LOX** and **OBX**), five mutation operators: (**AD2JM**, **AR2JM**, **AR3JM**,**SHM** and **INMV1**). The proposed approach was tested on a set of 240 problems.

Verma and Dhingra [21] described multiprocessor task scheduling in the form of permutation **FSP**, which has an objective function for minimizing the makespan. They proposed the following GA: randomly generated initial population, **RWS**, two crossover operators: **2PXV2**and **PMX,** and **INMV1**.

Chen et al [4] proposed a self-guided **GA** for permutation to minimize FSP’s makespan. In the proposed algorithm they used **TTS**, **2PXV2**, and**AR2JX**. They conducted extensive computational to compare the self-guided **GA** with several other algorithms using the 120 Taillard instances.

3. SUMMARY

A summary of different GAs operators mentioned in the reviewed literature: selection methods, crossover and mutation operator methodsare shown in Table 1. We can notice that problem sizes range from n $\in$ [8 – 500] and m $\in$ [2 – 40].

Based on Table 1, we count **4** selection methods, **17** crossover operators and **8** mutation operators used in designing different genetic algorithms for solving FSP as shown in Tables 2, 3 and 4, respectively.

Table 1: Summary of Genetic Algorithms Operators Used for Solving FSP

Author(s)	***Year***	***Criterion***	**GA operators**			**Max problem size**	**System specification**
			Selection Method (Sm)	**Crossover Method (Cm)**	**Mutation Method (Mm)**		
Chen et al.	1995	makespan	RWS	PMX	---	(25, 15)	Programmed using Fortran 77 & run on a SUN 4/490 Workstation.
Reeves	1995	makespan	RKS RMS	1PX* 2PXV1	AR2JM SHM*	(75,20)	Programmed in Pascal, & run on a Sequent S82 Computer
							---
Murata et al.	1996	makespan	RWS	1PX 2PX(1*,2,3) PMX(1,2) ERX PMX CX	AD2JM AR2JM AR3JM SHM*	(50.10)	
Tang and Liu							Pentium PC Computer
	2002	mean flow time	RWS	PMX	SHM	(150,20)	
Eliter et al.							Programmed in Pascal, and run on a Pentium IV(256 MB) Computer
	2004	makespan	RWS	LOX	SHM	(40,40)	
							Programmed in C, and run on PCs Pentium Processors
Iyer and Saxena							
	2004	makespan	RWS	1PX LCSX*	AR2JM	(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	RKS TTS*	SJOX SBOX* SJ2OX SB2OX PMX* OX 2PX 1PX	SHM	(500,20)	Implemented in Turbo C++
Sadegheih							Coded in C++ and run on a PC with AMD Athlon 1.0 GHz CPU
	2006	makespan	RKS	OBX	AD2JM* INMV1 SHM	(8,7)	
Wang et al.							Implemented in Microsoft Visual Basic 6.0
	2006	makespan	RWS	LOX	SHM	(100,20)	Coded in a C++ &

Author(s)	***Year***	***Criterion***	**GA operators**			**Max problem size**	**System specification**
			Selection Method (Sm)	**Crossover Method (Cm)**	**Mutation Method (Mm)**		
Octavia et al				PMX* 1PX NAX			developed at MIT & run under GNU g++ Complier
Adusumilli et al.	2007	makespan	RWS	SBOX	AD2JM	n=120 factory	Implemented in Borland Delphi and run on a PC P4 Processor with 3 GHz, 512 MB
Kahraman et al.	2008	makespan	RWS	PMX	AD2JM AR2JM AR3JM SHM INMV1*	(20,2)	Implemented in the Java using an IBM P 1.4 GHz Computer with 512 MB
Kim and Jeong	2008	makespan	RWS* TTS	PBX* OX PMX CX LOX OBX	---	(15,10)	Implemented in Borland Delphi and run on a PC P4 Processor with 3 GHz, 512 MB
Engin et al.	2009	makespan	RWS	PBX	AR2JM AD2JM* SHM INMV1	(50,5)	Implemented using MATLAB at command line Pentium 4 with 3GHz Processor and 512 MB.
Verma and Dhingra	2011	makespan	RWS* TTS	PBX* OX PMX CX LOX OBX	INMV1	(100,8)	
Chen et al.	2011	makespan	RWS	2PX PMX	AD2JM	(15,4)	
	2012	makespan	TTS	2PX	SHM	(200,20)	

***Shows the best GA operators (Sm, Cm and Mm).**

Next section, an experiment is designed to investigate the impact of these methods on the performance of the GA in solving FSP problem**.**

4. EXPERIMENTAL SET UP

A genetic algorithm is built based on the steps mentioned in the subsection (1.2) using C # 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.

4.1 Experimental Design

The tournament selection method is used with 3 different tour sizes, 2, 3 and 4. They are coded as TTS2, TTS3 and TTS4 respectively. Therefore, the number of selection methods became 6. And a two types of crossover operator methods are used first time for FSP (Maximal Preservative Crossover (MPX) and Alternating Position Crossover (APX)).Therefore, the number of crossover operator methods becomes 17.And we used one types of mutation operator methods used first time for FSP (Scramble mutation (SCM)).Therefore, the number of mutation operator methods became 8.

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

The full factorial experiment is 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 * 8Mm* 4 Ps * 5 Pc * 10 Pm = 163200.

Table 3: Crossover operators methods used in FSP

#	Crossover Operator Methods(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

4.2 Test Problem

Every combination of the experiment is tested 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

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

of this problem is 1278 time units. The problem data and information can be downloaded from the OR Library.

5. 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 runs are computed. The performance measure used is the number of combinations that give the upper bound of the test problem (i.e. ta001) based on a specific operator. This measure called N_{α}. Where $\alpha \in \{S_m, C_m, M_m\}$, S_m={RMS,RKS,RWS,TTS2,TTS3 and TTS4}, C_m={1PX, 2PXV1, 2PXV2, 2PXV3, PBXV1, PBXV2, LOX, PMX, LCSX, OX, OBX, CX, ER, MPX, SJOX, SBOX, APX}, and M_m={AD2JM, AR2JM, AR3JM, SHM, INMV1, INMV2, DM, SCM}.
The computational results show that 9041 out of 163200 combinations give the upper bound value and the remainders are worse than the upper bound.
In the following subsections, the impact of the three GA operators will be investigated based on N_{α} = 9041 and tested using chi square test.

5.1 Impact of The Selection Methods (Sm)

Fig.1 shows the distribution of the 9041 combinations on the selection methods. Applying the chi square test gives P-value equals 0.0000 which means significant differences among the selection methods. This is clear as shown in Fig. 1 that the tournament selection with tour size=4 (i.e. **TTS4**) gave best results than other selection methods, then **(TTS3, TTS2, and RKS)** respectively, and we see that the **RMS and RWS** gave worst results. This finding actually is surprising where the tournament method have not been paid attention from the researchers as shown in Table 1. Moreover, amazing finding is that the random selection method gets better results than the roulette wheel selection.

5.2 Impact of Crossover Operator Methods (Cm)

Fig. 2 shows the distribution of the 9041 combinations on the crossover operator methods. Applying the chi square test give P-value equals 1.4059E-226which means significant differences among the crossover operator methods. This is clear as shown in Fig. 2 that the crossover operator methods (**2PXV3**) gave best results than other crossover operator methods, then , **(1PX, SBOX, 2PXV1, OBX, SJOX, PBXV1, PBXV2, APX, LOX, 2PXV2, LCSX and PMX)** respectively. From the fig. we observe that the crossover operator methods (ERX, MPX, OX and CX) gave worst results.

5.3 Impact of Mutation Operator Methods (Mm)

The distribution of the 9041 combinations on the mutation operator methods is shown in Fig. 3. Applying the chi square test give P-value equals 0.0000 which means significant differences among the mutation operator methods. This is clear as shown in Fig. 4 that the adjacent two-job change mutation **(AD2JM)** gave best results than other

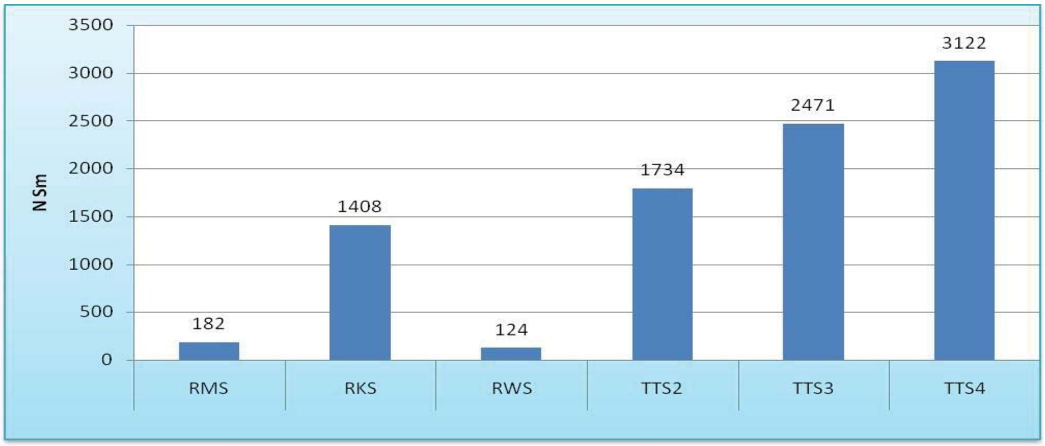

Fig. 1: Number of combinations obtained upper bound based on selection method

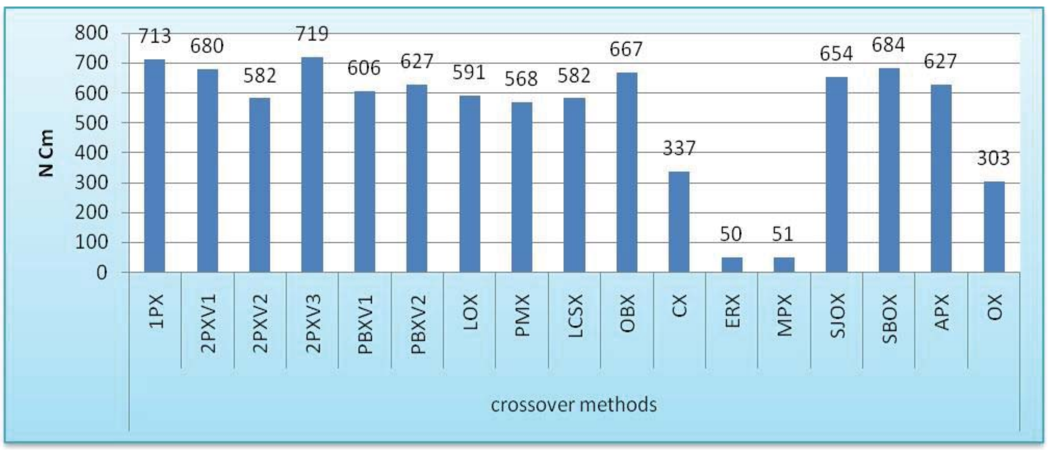

Fig. 2: Number of combinations obtained upper bound based on crossover method

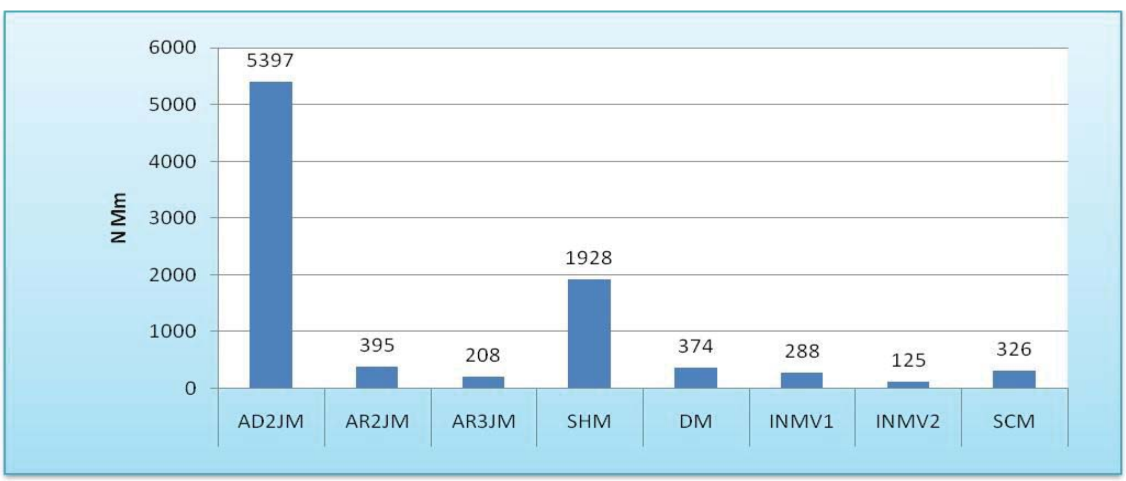

Fig. 3: Number of combinations obtained upper bound based on mutation method

crossover operator methods, then shift mutation **(SHM)**. And we see that the other mutation operator methods gave worst results.

6. CONCLUSIONS

In this study we investigate the impact of the GA operators (selection methods, Crossover operator methods and Mutation operator methods) on the performance of GA in solution of FSP, we find the following GA operators: 4 selection methods shown in Table 2, 17 crossover operators shown in Table 3
and 8 mutation operators shown in Table4 used in designing different genetic algorithms for solving FSP. Full factorial experiment is designed and tested on known benchmark problem. The extensive computational results show that:

- Tournament selection method with three tour sizes (TTS2, TTS3 and TTS4) give best results from all the selection methods. This finding actually is surprised us where the tournament method almost have not been paid attention from the researchers as shown in Table 1. In addition, though in most of the previous research papers roulette wheel selection (RWS) was used widely, we found its performance is worse than the random selection.

- Two-point crossover version 3(2PXV3)gives better results than all other crossover methods. Also, many crossover methods show almost the same high performance like 1PX, 2PXV1, OBX and SBOX. Moreover, four methods, CX, EPX, MPX, and OX, should be avoided when designing GA because of its bad performance results.

- Adjacent two-job change mutation **(**AD2JM) gives the best results from all the mutation methods. The shift mutation (SHM) is considered the second best so we note that it is used in most of the previous research papers 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. So, we recommend the researchers to build up based on our findings.

REFERENCES

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

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

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

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

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

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

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

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

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

10. Kim, K., and Joeng, J., **"Flow shop scheduling with no-wait flexible lot streaming using adaptive genetic algorithm".** International Journal Advance Manufacturing Technology 44, 2009. pp. 1181-1190.

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

12. Murata, T., Ishibuchi, H., and Tanaka H., **"Genetic algorithms for flow shop scheduling.**

13. Naderi, B. and Ruiz, R., **"The distributed permutation flow shop scheduling problem".** Computers and Operations Research, 37(4), 2010. pp. 754-768.

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

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

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

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

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

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

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

21. Verma, R., and Dhingra, S., **"Genetic algorithm for multiprocessor task scheduling".** International journal of computer science and management studies, 11(2), 2011, pp 181-185. 181-185.

22. Wang L., Zhang L, and Zhang, D. Z. "**An effective hyprid genetic algorithm for flow shop scheduling with limited buffers"** computer and operations research, 33, 2006, pp 2960-2971.