Volume 4, Issue 6, June 2014 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Comparative Analysis of Crossover Operator in Job Shop Scheduling Problem

Er.Shobhit Gupta
Assistant Professor,
Deptt. Of CSE &IT,
KITM,Kurukshetra, India

Ruchi Gaba
PG Scholar
KITM,Kurukshetra
India

Abstract— ***Genetic algorithms (GA) are wide class of global optimization methods. A genetic algorithm is one of a class of algorithms that searches a solution space for the optimal solution to a problem. Many works have been reported for this problem using Genetic Algorithm (GA). The GA is one of the most powerful optimization methods based on the mechanics of natural evolution. One of the problems in using genetic algorithms is the choice of crossover operator. A very famous best-known NP-hard problem is job shop scheduling problem. Job shop scheduling problem is one of the most important problems in the combinatorial optimization problems and it is applied to various fields about engineering. The work proposed here intends to test the performance of different Crossover used in GA and compare the performance for each of them This research presents an investigation on comparison of PMX, OX, CX crossover operators to solve JSSP problem. The makespan is the measure used to evaluate the genetic crossover operators. The main conclusion is that there is a crossover operator having the best average performance on a specific set of solved instances. The objective is therefore to improve the performance of GA by using these crossover operators.***

Keywords: Genetic Algorithms, NP-Hard problem, makespan, JSSP, PMX,CX,OX.

I. AN OVERVIEW OF GENETIC ALGORITHM

Genetic algorithm (GA) technology has been around for over 30 years[1]. However, applications based on this technology are just recently (past 5 years) being fielded in earnest. Genetic algorithms are inspired by Darwin's theory about evolution. Solution to a problem solved by genetic algorithms is evolved. GA is a process which mimics the way biological evolution works. "A genetic algorithm (GA) is a search technique to find approximate solutions to combinatorial optimization problems. Genetic algorithms are a particular class of evolutionary algorithms that use techniques inspired by evolutionary biology such as inheritance, mutation, natural selection, and recombination (or crossover)."

A genetic algorithm is a type of searching algorithm. It searches a solution space for an optimal solution to a problem. The key characteristic of the genetic algorithm is how the searching is done[10]. The algorithm creates a "population" of possible solutions to the problem and lets them "evolve" over multiple generations to find better and better solutions.

1. Create a **population** of random candidate solutions named *pop*.
2. Until the algorithm termination conditions are met, do the following (each iteration is called a generation):
(a) Create an empty population named *new-pop*.
(b) While *new-pop* is not full, do the following:
i. **Select** two **individuals** at random from *pop* so that individuals which are more **fit** are more likely to be selected.
ii. **Cross-over** the two individuals to produce two new individuals.
(c) Let each individual in *new-pop* have a random chance to **mutate**.
(d) Replace *pop* with *new-pop*.
3. Select the individual from *pop* with the highest **fitness** as the solution to the problem.

Figure 1: The Genetic Algorithm

The **population** is the collection of candidate solutions that we are considering during the course of the algorithm. Over the generations of the algorithm, new members are "born" into the population, while others "die" out of the population. A single solution in the population is referred to as an **individual**. The fitness of an individual is a measure of how "good" the solution represented by the individual is. The better the solution, the higher the fitness – obviously, this is dependent on the problem to be solved[13]. The **selection** process is analogous to the survival of the fittest in the natural world. Individuals are selected for "breeding" (or **cross-over**) based upon their fitness values – the fitter the individual, the more likely that individual will be able to reproduce. The cross-over occurs by mingling the two solutions together to produce

two new individuals. During each generation, there is a small chance for each individual to **mutate**, which will change the individual in some small way.

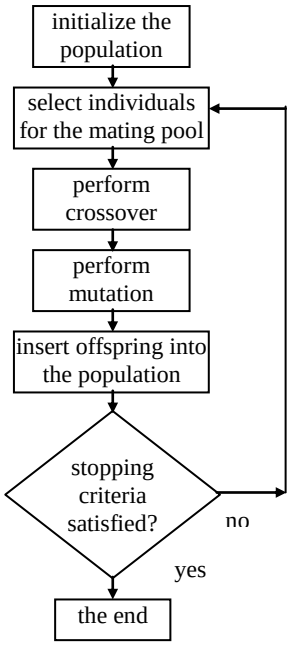


Figure 1: Flowchart of a genetic algorithm

II. JOB SHOP SCHEDULING PROBLEM

Scheduling refers to the act of assigning resources to tasks or assigning tasks to resources. The job-shop scheduling problem is one of the most complicated and typical. It aims to allocate m machines to perform n jobs in order to optimize certain criterion.

JSSPs are well known combinatorial optimization problems, which consist of a finite number of jobs and machines. Each job consists of a set of operations that has to be processed, on a set of known machines, and where each operation has a known processing time[2]. A schedule is a complete set of operations, required by a job, to be performed on different machines, in a given order. In addition, the process may need to satisfy other constraints such as (i) no more than one operation of any job can be executed simultaneously and (ii) no machine can process more than one operation at the same time. The objectives usually considered in JSSPs are the minimization of makespan, the minimization of tardiness, and the maximization of throughput. The total time between the starting of the first operation and the ending of the last operation, is termed as the makespan. In JSSPs, the size of the solution space is an exponent of the number of machines, which makes it quite expensive to find the best makespan for larger problems. By larger problem, we mean a higher number of jobs and (or) a higher number of machines[5]. Most JSSPs that have appeared in the literature are for ideal conditions. However, in practice, process interruptions such as machine breakdown and machine unavailability are very common, which makes JSSPs more complex to solve.

A classical job-shop scheduling problem contains N jobs and M machines where each job needs to process for *M* operations. Every operation is assigned to a particular machine. Each machine is capable of processing only one kind of operation. In addition, there are several other basic conditions that have to be considered for the classical JSSPs as stated below[8].

- The number of operations for each job is finite.
- The processing time for each operation in a particular machine is defined.
- There is a pre-defined sequence of operations that has to be maintained to complete each job.
- Delivery times of the products are undefined.
- No setup cost and tardiness cost.
- A machine can process only one job at a time.
- Each job visits each machine only once.
- No machine can deal with more than one type of task.
- The system cannot be interrupted until each operation of each job is finished.
- No machine can halt a job and start another job before finishing the previous one.
- Each and every machine has full efficiency.

The job shop scheduling problem is a generalization of the classical job shop problem. In the static job-shop scheduling problem, finite jobs are to be processed by finite machines. Each job consists of a predetermined sequence of task operations, each of which needs to be processed without pre-emption for a given period of time on a given machine. Tasks of the same job cannot be processed concurrently and each job must visit each machine exactly once. Each

operation cannot be commenced until the processing is completed, if the precedent operation is still being processed. A schedule is an assignment of operations to time slots on a machine[12]. The makespan is the maximum completion time of the jobs and the objective of the JSSP is to find a schedule that minimizes the makespan.

The job-shop scheduling problem (JSSP) [3]can be described as a set of *n* jobs denoted by *J j* where *j* =1,2,..., *n* which have to be processed on a set of *m* machines denoted by *Mk* where *k* =1,2,...,*m*. Operation of *j* th job on the *k* th machine will be denoted by *Ojk* with the processing time *Pjk* . Once a machine starts to process a job, no interruption is allowed. The time required for all operations to complete their processes is called makespan. In this paper, our intention is to minimize this makespan value. Every time when makespan is optimised, a schedule can be described by the processing orders of operations on the machines.

"Job-Shop Scheduling Problem (JSSP) is one of the well known hardest combinatorial optimization problems. JSSP being amongst the worst members of the class of NP-hard problems" (Gary and Johnson, [1979]), there is still a lot of room for improvement in the existing techniques. Because of its large solution space JSSP is considered to be comparatively one of the hardest problems to solve[14]. "If there are *n* jobs and *m* machines the number of theoretically possible solutions is equal to (*n*!)*m*" (Noor [2007]). Among these solutions an optimal solution, for a certain measure of performance, can be found after checking all the possible alternatives. But the checking of all the possible alternatives can only be possible in small size problems. For example, a very simple problem of 5 jobs and 8 machines will give 4.3x1016 numbers of alternatives. Even with a high performance computer, that can evaluate one alternative per micro second, complete enumeration of this problem to find out the optimal solution would take more than 1000 years of continuous processing (Hitomi [1996], Morshed [2006]).

Solution Techniques to Handle JSSP

A number of solution techniques to handle the JSSP have been developed over the years. A broad classification of the scheduling techniques is given in Jain [1998]. Initially the techniques are divided into two classes as approximation and optimization techniques[4].

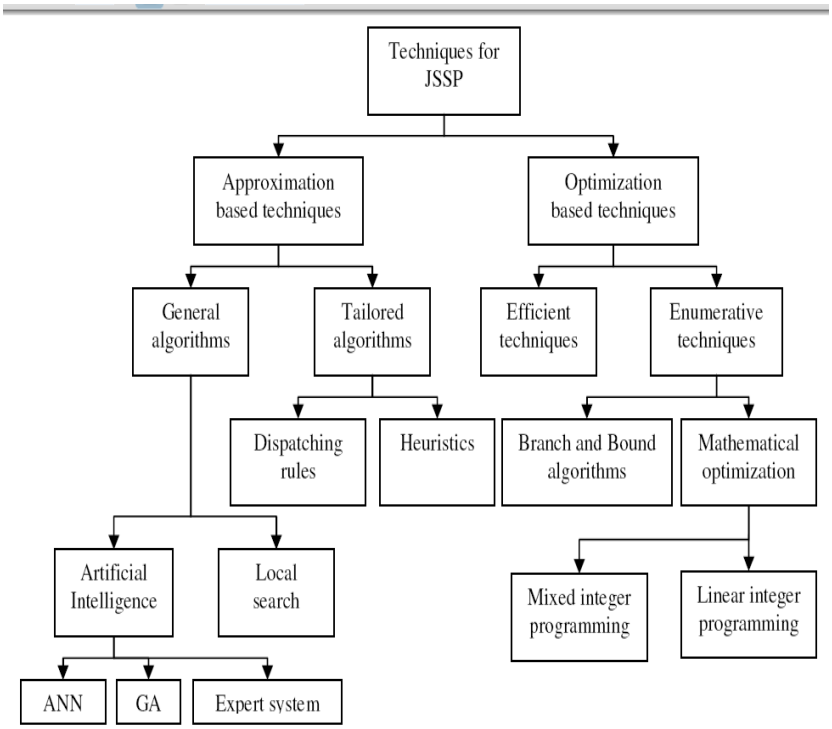


Fig.2: Solution approaches to JSSP

III. GENETIC ALGORITHM WITH PMX, CX, OXOPERATORS

John Holland proposed Genetic Algorithm in 1975 [1]. In the field of artificial intelligence a genetic algorithm is a search heuristic that mimics the process of natural evolution. Genetic Algorithm belongs to class of evolutionary algorithm.GA begin with various problem solution which are encoded into population, a fitness function is applied for evaluating the fitness of each individual, after that a new generation is created through the process of selection, crossover and mutation. After the termination of genetic algorithm, an optimal solution is obtained. If the termination condition is not satisfied then algorithm continues with new population.

Crossover operators are the backbone of the genetic algorithm. Crossover is a genetic operator that combines (mates) two chromosomes (parents) to produce a new chromosome (offspring)[6]. The idea behind crossover is that the new chromosome may be better than both of the parents if it takes the best characteristics from each of the parents. Crossover occurs during evolution according to a user-definable crossover probability.

The following types of crossover:-

- **One Point**: A crossover operator that randomly selects a crossover point within a chromosome then interchanges the two parent chromosomes at this point to produce two new offspring.
  Consider the following 2 parents which have been selected for crossover. The "|" symbol indicates the randomly chosen crossover point.
  Parent 1: 11001|010
  Parent 2: 00100|111
  After interchanging the parent chromosomes at the crossover point, the following offspring are produced:-
  Offspring1: 11001|111
  Offspring2: 00100|010
- **Two Point:** A crossover operator that randomly selects two crossover points within a chromosome then interchanges the two parent chromosomes between these points to produce two new offspring.

  Consider the following 2 parents which have been selected for crossover. The "|" symbols indicate the randomly chosen crossover points
  Parent 1: 110|010|10
  Parent 2: 001|001|11
  After interchanging the parent chromosomes between the crossover points, the following offspring are produced:-
  Offspring1: 110|001|10
  Offspring2: 001|010|11

Reproduction makes clones of good strings but does not create new ones. Crossover operators are applied to mating pool with hope that it creates a better offspring.Here three crossover operators PMX, CX and OX are discussed.

1. **Partially Matched Crossover:** The partially matched crossover (PMX) was proposed by Goldberg and Lingle. In partially matched crossover operator two crossover points are selected randomly from the parent's chromosomes to produce the offspring. The two crossover points give a matching selection which is used to affect a cross through position by position exchange operations. Partially Matched Crossover is a crossover for mating individuals consisting of enumerated type chromosomes in unique gene representation. Partially-Matched Crossover (PMX) is applied by choosing two random different crossing points in the strings and exchanging and swapping the segment contents from one individual to another. PMX is used to avoid duplicate allele values after crossover.

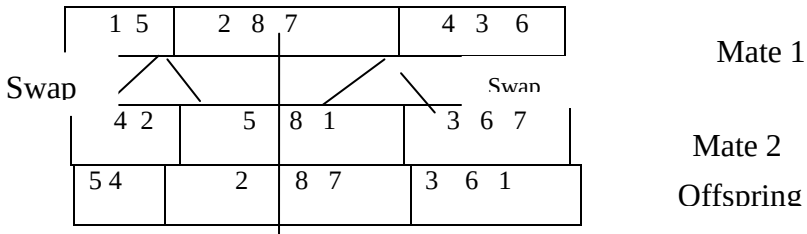


Fig.3 Partially matched crossover

2. **Ordered Crossover**
   The ordered crossover (OX) was proposed by Davis which produces an offspring by selecting a sequence of parent and preserve the relative order of cities in other parent. In ordered crossover operator two cut points are randomly selected from parent.s chromosomes. Here to produce the offspring O1 the genes between the cut points are replaced by the genes in the second parent.

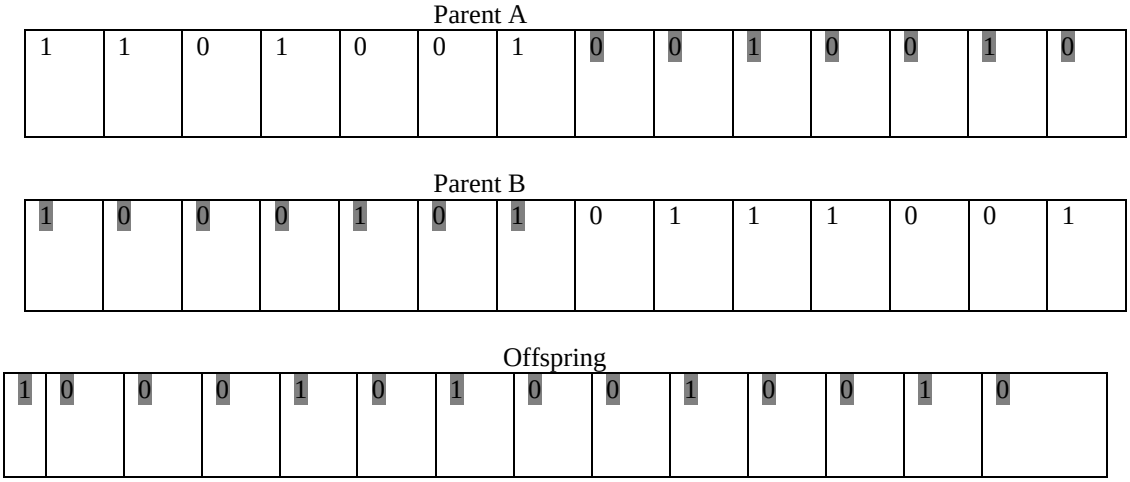


Fig.4 Ordered Crossover

3. **Cyclic Crossover**: Cyclic crossover operator performs recombination under the constraint that each gene comes from the parent or other.

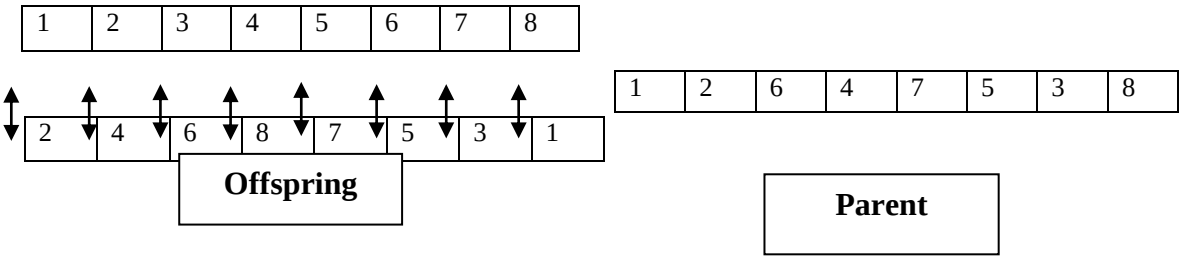


Fig.5 Cyclic Crossover

IV. SIMULATION RESULTS

This research implies that maximum number of jobs is to be performed in less time. Any operation can be applied to any machine in any time span. We have to schedule our jobs in minimum time with better efficiency. This minimum time required to complete the jobs is called makespan. Any operation can be applied on any machine with any random number of jobs. Initially number of machines and number of jobs are defined. The population is generated with various numbers of different entries in the table regarding operation name, start time and end time for particular job. The generated population will be scheduled and selection is performed using tournament selection and then crossover and mutation operators are applied to get the best efficient sequence in which operations are to be performed on the given number of machines. The efficient sequence is gathered using three crossover operators PMX, CX, OX which produce different results using same values of number of machines and jobs. The result obtained will be different in each case.

STEP 1: Enter the number of jobs and machines

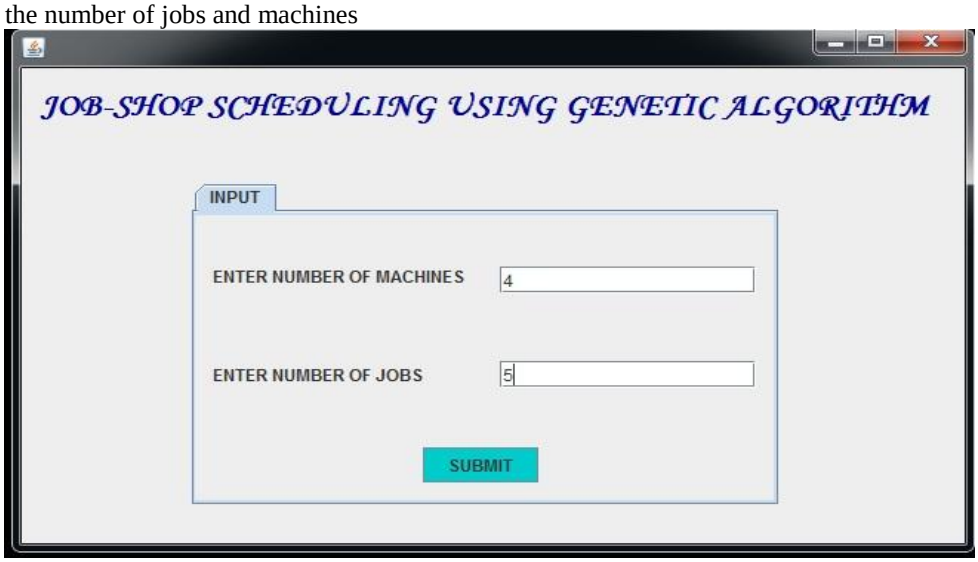


Fig 6 Number of machines and jobs

STEP 2: Enter the number of operations in each job. Also enter the starting and execution time of each operation. Enter population size and generate population.

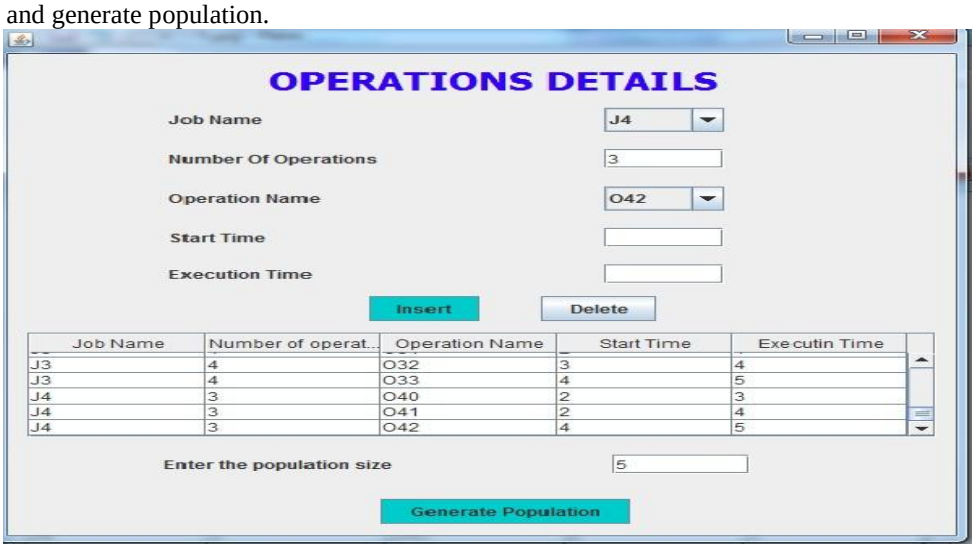


Fig 7 Number of operation in each job

STEP 3: Population of chromosomes is generated. Now using PMX, OX, CX operator generate results .

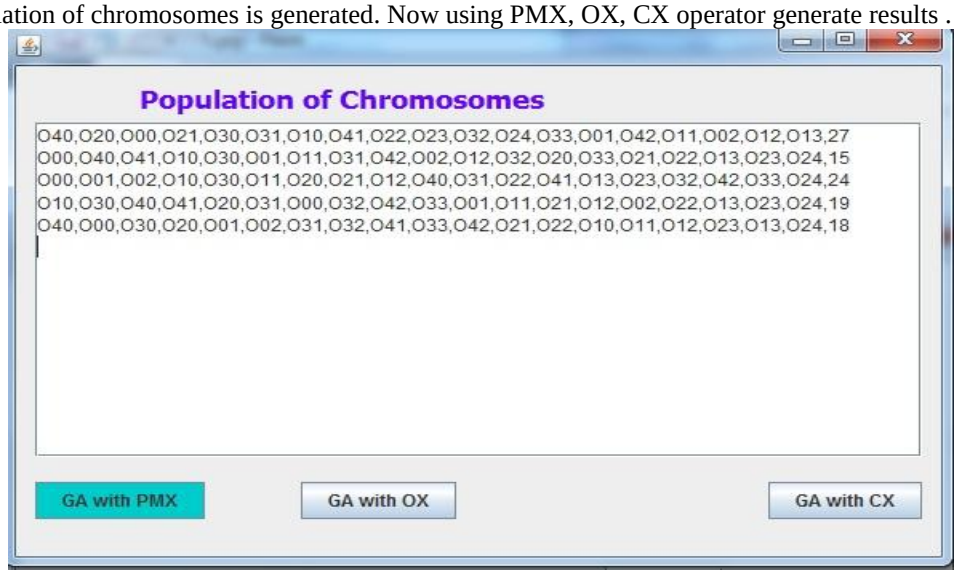


Fig 8 Population is generated

STEP 4: Obtain the optimized solution using PMX operator

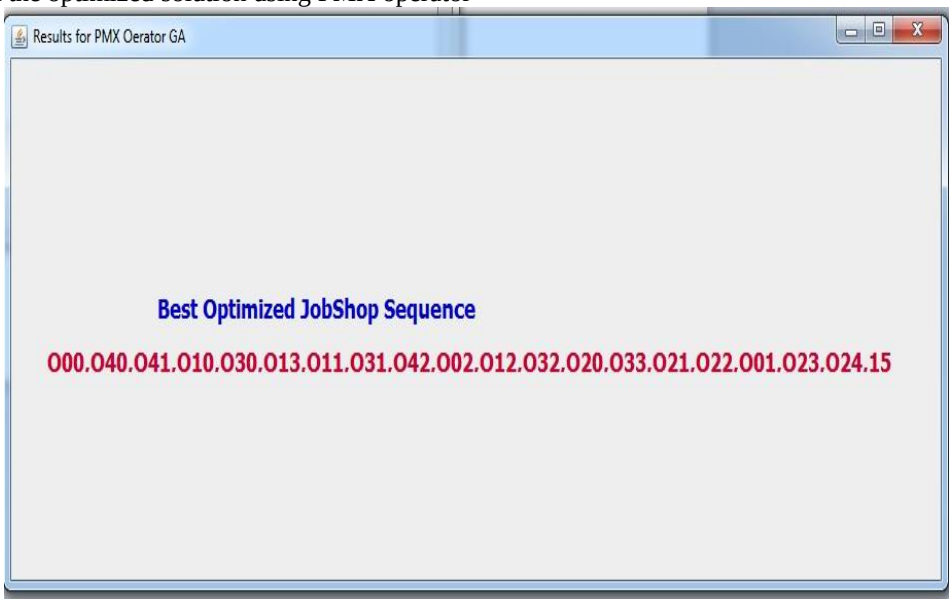


Fig 9 Using PMX operator

STEP 5 Obtain the optimized solution using OX operator.

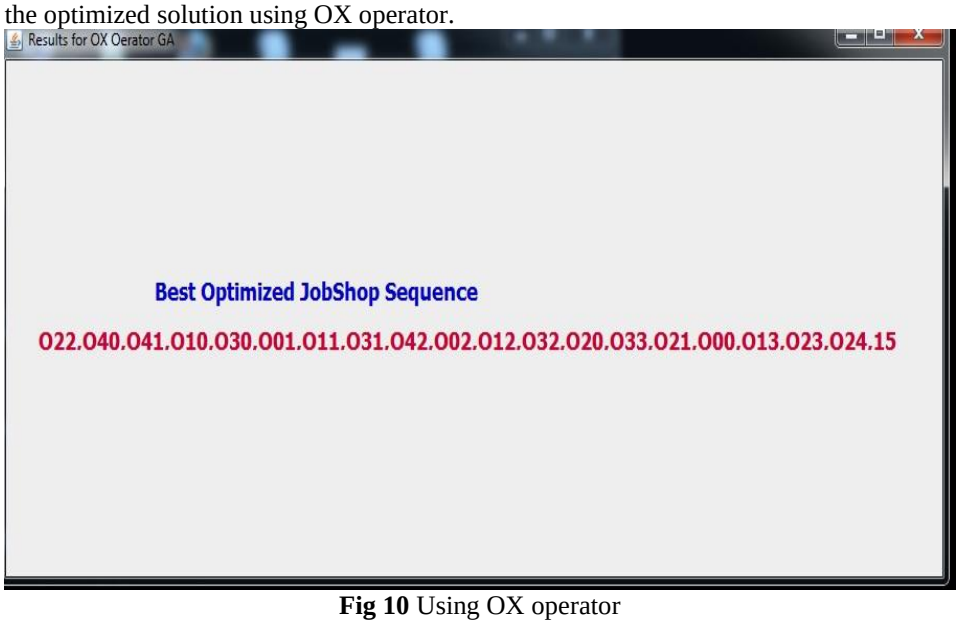


Fig 10 Using OX operator

STEP 5 Obtain the optimized solution using CX operator.

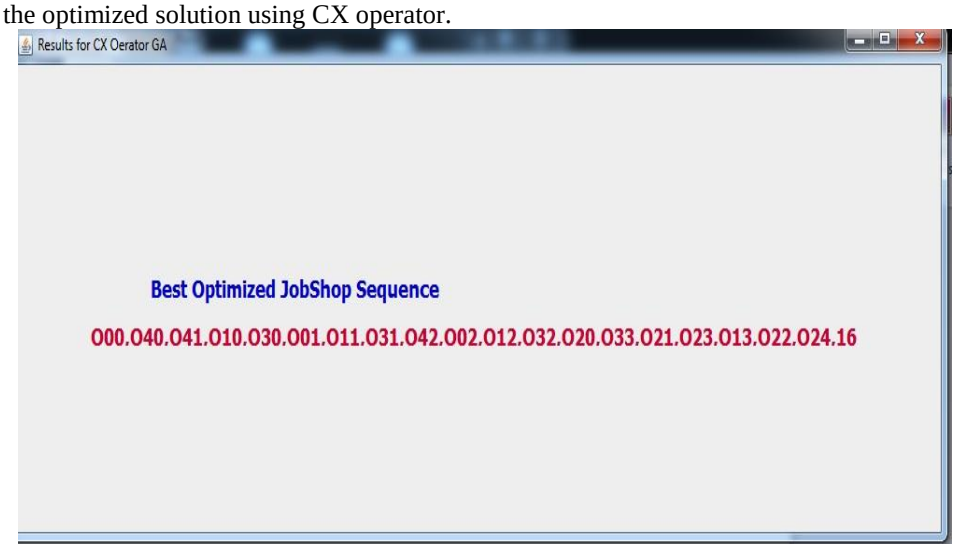


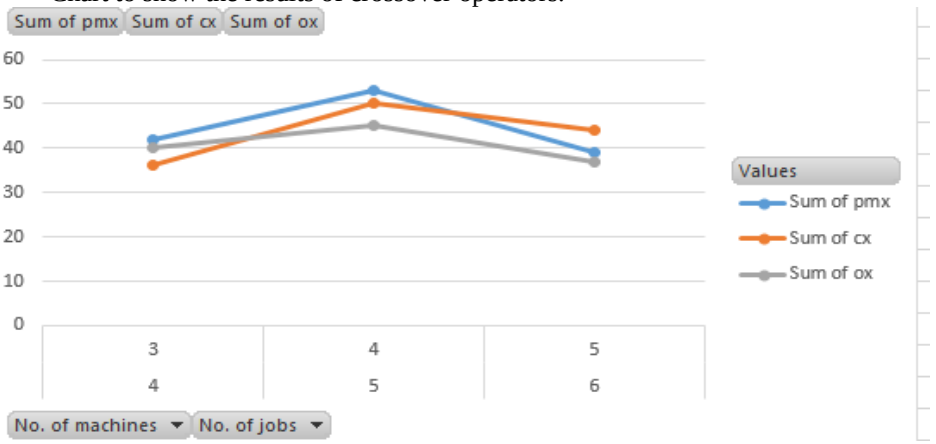
Fig 11 Using CX operator

Time taken by the crossover operators on different no of machines and jobs as shown in the table

No. of machines	No. of jobs	PMX	CX	OX
6	5	39	44	37
5	4	53	50	45
4	3	42	36	40

Table 1. Time taken by crossover operators

Chart to show the results of crossover operators.



V. CONCLUSION

The experimental results show that the minimum length of the schedule are obtained from OX i.e. Ordered Crossover operator. This will provide the best optimal solution with less time and best sequence for operations to be performed on machines. The results show that the OX crossover outperforms the PMX and CX crossover operator. The best scheduled result will proved to produce best makespan and fewer throughputs from the desired population generated. OX improves the GA's from premature convergence or time or both.

REFERENCES

[1] Holland, J.A. Adaptation in natural and artificial systems. Ann Arbor: University of Michigan, 1975.

[2] Mahanim Omar, Adam Baharum, Yahya Abu Hasan," A JOB-SHOP SCHEDULING PROBLEM (JSSP) USING GENETIC ALGORITHM (GA)" Precedings of the 2nd IMT-GT Regional conference on Mathematics,Statistics and application.Universtiy Sains Malaysia Penag,13-15 June 2006.

[3] E.Falkenauer and S.Bouffouix "A Genetic Algorithm for Job Shop".

[4] Bhuvan Sharma,Xin Yao"Characterising GA approaches to JSSP"

[5] James C. Werner,Mehmet E. Aydin,Terence C. Fogarty" Evolving genetic algorithm for Job Shop Scheduling problems" Proceedings of ACDM 2000,PEDC, Unviersity of Plymouth, UK.

[6] JORGE MAGALHÃES-MENDES "A Comparative Study of Crossover Operators for Genetic Algorithms to Solve the Job Shop Scheduling Problem" WSEAS TRANSACTIONS on COMPUTERS. Issue 4, Volume 12, April 2013.

[7] Liang Sun, Xiaochun Cheng "Solving Job Shop Scheduling Problem Using Genetic Algorithm with Penalty Function " International Journal of Intelligent Information Processing Volume 1, Number 2, December 2010

[8] Ab. Rahman Ahmad and Faisal. RM *Job Shop Scheduling using GA*. National Conference Management Science and Operations Research 2003. (2003).

[9] C. Bierwirth , D. C. Mattfeld and H. Kopfer. *On Permutation Representations for Scheduling Problems.* Parallel Problem Solving from Nature IV, Springer-Verlag, pp . 310-318 (1996).

[10] G. Mintsuo and C. Runwei.. *GA and Engineering Design*, John Willey & Sons Inc, New York USA.(2002).

[11] J.F. Gonçalves, , Jorge José de Magalhães Mendes and M. C. Resende. *A Hybrid Genetic Algorithm for the Job Shop Scheduling Problem.*.AT&T Labs Research Technical Report TD- 5EAL6J, September 2002. (2002).

[12] M. Pinedo and X. Chao. *Operation Scheduling with Applications in Manufacturing and Services.* McGraw-Hill International Editions. (1999).

[13] M. T. Jensen and T. K. Hansen. *Robust Solutions to Job Shop Problems.* Proceedings of the 1999 Congress on Evolutionary Computation, pages 1138-1144. (1999).

[14] S. French. *Sequencing and Scheduling : An Introduction to the Mathematics of the Job Shop.* John Willey & Sons Inc, New York USA.(1982).

[15] T. Yamada and R. Nakano. *Genetic Algorithms for Job-shop Scheduling Problems*. Proceedings of Modern Heuristic for Decision Support. Pp. 67-81, UNICOM Seminar, 18-19 March 1997,London. (1997)

