American Journal of Industrial and Business Management, 2016, 6, 774-789
Published Online June 2016 in SciRes. http://www.scirp.org/journal/ajibm
http://dx.doi.org/10.4236/ajibm.2016.66071

Scientific Research Publishing

A Study of Crossover Operators for Genetic Algorithm and Proposal of a New Crossover Operator to Solve Open Shop Scheduling Problem

Ellur Anand[1], Ramasamy Panneerselvam[2]

[1]Alliance School of Business, Alliance University, Bangalore, India
[2]Department of Management Studies, School of Management, Pondicherry University, Pondicherry, India
Email: ellur.anand@alliance.edu.in, panneer_dms@yahoo.co.in

Received 2 May 2016; accepted 21 June 2016; published 24 June 2016

Abstract

Open Shop Scheduling Problem (OSSP) is a combinatorial optimization problem for more than two machines and n jobs. Open Shop Scheduling Problem is another kind of scheduling problem along with flow shop and job shop scheduling problems. The open shop scheduling problem involves scheduling of jobs, where the sequence of the operations of each job can be arbitrarily chosen and need not be same. This means that the operations of the jobs can be performed in any sequence. In the absence of sequences for the jobs, for a given set of jobs, finding different parameters like maximum completion time C_{max} becomes highly difficult and complex. One can use complete enumeration method or branch and bound method to solve this problem optimally for small and medium size problems. The large size problems of open shop problem with more than two machines and with n jobs can be solved by either a heuristic or meta-heuristics such as genetic algorithm, simulated annealing algorithm, etc. to obtain very near optimal solution. The performance of the genetic algorithm is affected by crossover operator performed between two parent chromosomes. Hence, this paper explores various crossover operators used, while using evolutionary based genetic algorithm to solve open shop scheduling problems. It further attempts to propose a new crossover operator using three chromosomes.

Keywords

Open Shop Scheduling Problems, Genetic Algorithm, Crossover Operator, Chromosome Encoding

How to cite this paper: Anand, E. and Panneerselvam, R. (2016) A Study of Crossover Operators for Genetic Algorithm and Proposal of a New Crossover Operator to Solve Open Shop Scheduling Problem. *American Journal of Industrial and Business Management*, **6**, 774-789. http://dx.doi.org/10.4236/ajibm.2016.66071

1. Introduction

Flow shop scheduling problem, job shop scheduling problem and open shop scheduling problem differed in the way of operations in each job need to be processed on "m" machines [1] [2]. Flow shop scheduling problem involves scheduling of jobs, where each job has a sequence and the sequences of the jobs are the same. Job shop scheduling problem involves scheduling of jobs, where each job has a sequence but the sequences of all the jobs are not the same. Open shop scheduling problem involves scheduling of jobs, and each of the jobs does not have any sequence. The sequences of the jobs of this problem can be arbitrarily chosen and need not be same. The open shop scheduling problem is also called as moderated job shop (Panneerselvam [3]), which is between flow shop scheduling problem and job shop scheduling problem.

Let "m" be a set of machines, where processing on each machine represents an operation and "n" is the number of jobs with m operations that need to be processed on m machines. Each machine can process only one job at a time and each job can be processed only on one machine at a time. The objective is to complete the processing of all the jobs in such a way that the total time taken to complete all the jobs is minimized. This is called as makespan or maximum completion time (C_{max}). In this paper, Genetic Algorithm is considered to arrive at a sequence that provides near-optimal solution.

Genetic Algorithm has the following important steps [4].

1. Generate initial population of N chromosomes.
2. Evaluate the fitness function f(x) for each chromosome x in the population.
3. Sort the population by the objective function (fitness function) value in the desirable order depending on the type of objective function (maximization means descending order and minimization means in ascending order).
4. Copy a percentage of population to form a subpopulation.
5. Then randomly select two parent chromosomes from the subpopulation and perform crossover to produce their offspring.
6. Perform mutation in each of the offspring for a given probability.
7. Repeat the process till a specified number of generations is completed.
8. Finally identify the chromosome in the larger population which has the best fitness function value for implementation.

Crossover operator is an important stage in the Genetic Algorithm (GA), which involves mating of the selected chromosomes to produce child chromosomes with better fitness which in turn becomes parent chromosomes in the next generation. More than 60% of research papers consider minimization of maximum completion time or C_{max} as the objective function for the open shop scheduling problem, while the remaining papers consider the average completion time or total completion time, multi-criteria objective or to minimize the tardiness or earliness as the objective function.

The selection of an appropriate crossover operator for research work based on GA is highly significant step to reduce the maximum completion time C_{max}. This paper tries to make an attempt to describe various crossover operators used by various researchers to minimize the maximum completion time. This paper is organized as follows: Section 2 provides an overview of literature concerning GA with an objective of minimizing Cmax for an OSSP. The literature review covers various research works using different crossover operators. Section 3 describes various methods of encoding chromosome before performing crossover of chromosomes. It further explains decoding of the encoded chromosome. Section 4 discusses various crossover operators with a proper illustration. Section 5 deals with the new crossover operator proposed by the authors of the paper. Section 6 demonstrates the use of proposed new crossover operator by considering an instance and summarizes the result finally.

2. Literature Review

Anand and Panneerselvam [5] carried out a comprehensive review of literature on open shop scheduling problem. Senthilkumar and Shahbudeen [6] proposed a heuristic based on Genetic Algorithm (GA) to minimize the makespan of the open shop scheduling problem and they compared it to an earlier work of Panneerselvam [3]. The authors adopted two-point crossover method for the chromosomes that are encoded using implicit permutation coding. They developed a heuristics (HU1) that surpassed the performance of an earlier heuristic developed by Panneerselvam [3]. Louis and Xu [7] worked on storing the past results in memory as cases and then injecting them into the population of new problem situation to solve the problem using genetic algorithm. They called

it as case based reasoning (CBR) and the objective was to minimize the makespan. It is very similar to the way a patient is handled and treated by a doctor. It is much more convenient for a doctor to treat a patient with his medical history at hand than the patient without medical history. The authors adopted one point crossover of two digit encoded chromosomes. They adopted Partially Mapped Crossover with single digit encoded chromosomes too. They concluded that case based genetic algorithm performs better than randomly initialized genetic algorithm even up to 300 generations, for a problem size of 7 × 7 Open shop re-scheduling problems.

Zobolas *et al*. [8] developed a hybrid meta-heuristic by combining genetic algorithm with a variable neighbourhood search algorithm for the open shop scheduling problem to optimize the makespan. They used double-digit subscription encoding to encode the chromosomes and used linear order crossover operator to perform crossover of chromosomes. They tested the developed meta-heuristics against 192 different benchmark instances of Taillard [9], Brucker [10], Gueret and Prins [11]. In several of the instances, they found that the results of the heuristic perform better than the results of the heuristics developed by earlier researchers. Kokosinski and Studzienny [12] adopted a two-digit subscripts permutation encoding without repetition and with repetition. Linear order crossover (LOX) method with swap and invert mutation operators is used along with two- element tournament selection scheme with elitist policy. Two heuristics, viz. Longest Processing Time (LPT)- Task and LPT-Machine were used along with genetic algorithm to obtain better results. The objective here was to minimize the makespan.

Ching-Fang Liaw [13] focussed on hybrid genetic algorithm with tabu search as local optimization procedure for the open shop scheduling problem to minimize the makespan. The author has used simple permutation encoding method to represent a chromosome and experimented with several crossover operators, viz. partially mapped crossover (PMX), order crossover (OX), cycle crossover (CX), order-based crossover and position-based crossover. Linear order crossover (LOX) has given best results for the problem under consideration. Andresen *et al*. [14] discussed application of linear order crossover (LOX) with elitist strategy for the open shop scheduling problem with the objective of minimizing the sum of completion times or mean flow time. They adopted simple permutation encoding method. Tests are performed with different crossover probabilities 0.2, 0.3, 0.4, 0.5 and 0.6. Tabu search has been used as the local search tool. Matta [15] considered multiprocessor open shop (MPOS) scheduling problem with genetic algorithm using two-digit permutation encoding of chromosome to optimize the total completion time or makespan.

Several researchers used different types of crossover operators and encoding operators that served their purposes and objectives. There are situations, where certain chromosome encoding operator cannot be used. Binary encoding operator cannot be used to represent the chromosome for open shop scheduling problems. This paper tries to understand and explore different encoding and crossover operators. It also proposes a new crossover operator called Three-chromosome Juggling crossover operator.

3. Encoding of Chromosomes

Encoding of a chromosome or a string is an important stage in GA. This involves the process of representing the information contained in the problem in the form of an appropriate chromosome or string. Binary encoding is highly used encoding method. Binary encoding for OSSP is not an appropriate and feasible encoding method. The different encoding methods applied to the open shop scheduling problem with 4 machines and four jobs are presented below.

3.1. Permutation Encoding

This is the simplest way to encode chromosome for an OSSP but difficult to decode. For the problem size of 4 machines and 4 jobs (4 × 4), each job has four operations and the total number of operations irrespective of the jobs is 16. The operation numbers from 1 to 16 irrespective of the jobs are shown in **Table 1**, which is the encoding scheme of this problem.

The sixteen operations are assigned randomly to sixteen gene positions as shown in **Figure 1** form a chromosome.

The genes of the chromosome can be decoded as shown in **Table 2**. The sequence of the genes of the job 1 (1, 2, 3 and 4) are listed as per order of appearance of them in the chromosome as shown against the job 1 in **Table 2**. The sequence of the genes of the job 2 (5, 6, 7, and 8) are listed as per order of appearance of them in the chromosome as shown against the job 2 in **Table 2**. The sequence of the genes of the job 3 (9, 10, 11 and

Table 1. Permutation encoding of open shop scheduling problem of size 4 × 4.

	Machine 1	Machine 2	Machine 3	Machine 4
Job 1	1	2	3	4
Job 2	5	6	7	8
Job 3	9	10	11	12
Job 4	13	14	15	16

6	10	5	2	7	11	15	3	16	9	8	14	4	12	1	13

Figure 1. A typical chromosome with permutation encoding for OSSP of size 4 × 4 shown in **Table 1**.

Table 2. Sequence of genes of jobs.

Job	Sequence of genes
1	2 – 3 – 4 – 1
2	6 – 5 – 7 – 8
3	10 – 11 – 9 – 12
4	15 – 16 – 14 – 13

12) are listed as per order of appearance of them in the chromosome as shown against the job 3 in **Table 2**. The sequence of the genes of the job 4 (13, 14, 15, and 16) are listed as per order of appearance of them in the chromosome as shown against the job 4 in **Table 2**.

The sequence of operations for each job is decoded as shown in **Table 3**. The gene numbers of the job 1 shown in **Table 3** correspond to the respective machine numbers without any change. So, there is no need of decoding obtain the operations of the job. The operation numbers for the genes of the job 2 are obtained by subtracting 4 from each of them. The operation numbers for the genes of the job 3 are obtained by subtracting 8 from each of them. The operation numbers for the genes of the job 4 are obtained by subtracting 12 from each of them.

3.2. Two-Digit Permutation Encoding

Table 4 shows the two-digit encoding of all the operations for a 4 × 4 size problem proposed by Louis and Xu [6]. The two digits shown in each job-machine combination represents the corresponding job number and machine number. The total number of digits to represent will be 2 × m × n. For the problem in **Table 2**, the number of digits used is 2 × 4 × 4 or 32 digits. A chromosome based on the encoding shown in **Table 4** can be as shown in **Figure 2**.

The chromosome can be decoded as shown below.

First the two-digit genes of the job 1 which appear from left to right of the chromosome are written in the same order against the job 1 in **Table 5**. In the same way, the two-digit genes of the remaining jobs are written in the **Table 5**.

Now, the decoding of the genes of each job shown **Table 5** is done by ignoring the first digit (job number) of each of the two-digit genes of that job as shown in **Table 6**.

3.3. Two-Digit Subscript Permutation Encoding

This is same as the two-digit permutation encoding except in the way of representation as shown in **Table 7** for a problem of 4 × 4 open shop scheduling problem. The combination of job and operation is shown as a subscript of O, which means operation.

Table 3. Sequence of operations.

Job	Sequence of genes	Sequence of operations
1	2 (Machine 2) – 3 (Machine 3) – 4 (Machine 4) – 1 (Machine 1)	2 – 3 – 4 – 1
2	6 [6 – 4 = 2 (Machine 2)] – 5 [5 – 4 = 1 (Machine 1)] – 7 [7 – 4 = 3 (Machine 3)] – 8 [8 – 4 = 4 (Machine 4)]	2 – 1 – 3 – 4
3	10 [10 – 8 = 2 (Machine 2)] – 11 [11 – 8 = 3 (Machine 3)] – 9 [9 – 8 = 1 (Machine 1)] – 12 [12 – 8 = 4 (Machine 4)]	2 – 3 – 1 – 4
4	15 [15 – 12 = 3 (Machine 3)] – 16 [16 – 12 = 4 (Machine 4)] – 14 [14 – 12 = 2 (Machine 2)] – 13 [13 – 12 = 1 (Machine 1)]	3– 4 – 2 – 1

Table 4. Two-digit permutation encoding of open shop scheduling problem of size 4 × 4.

	Machine 1	Machine 2	Machine 3	Machine 4
Job 1	11	12	13	14
Job 2	21	22	23	24
Job 3	31	32	33	34
Job 4	41	42	43	44

24	44	12	22	14	32	31	43	23	11	34	41	21	13	33	42

Figure 2. A typical chromosome for an OSSP of size 4 × 4 with two-digit encoding shown in **Table 4**.

Table 5. Sequence of genes of jobs.

Job	Sequence of genes
1	12 – 14 – 11 – 13
2	24 – 22 – 23 – 21
3	32 – 31 – 34 – 33
4	44 – 43 – 41 – 42

Table 6. Sequence of jobs.

Job	Sequence of operations
1	2 – 4 – 1 – 3
2	4 – 2 – 3 – 1
3	2 – 1 – 4 – 3
4	4 – 3 – 1 – 2

Table 7. Two-digit subscripts permutation encoding for 4 × 4 OSSP.

	Machine 1	Machine 2	Machine 3	Machine 4
Job 1	O_{11}	O_{12}	O_{13}	O_{14}
Job 2	O_{21}	O_{22}	O_{23}	O_{24}
Job 3	O_{31}	O_{32}	O_{33}	O_{34}
Job 4	O_{41}	O_{42}	O_{43}	O_{44}

3.4. Implicit Permutation Encoding

In this encoding method, the numbers do not explicitly mention the job on particular machine, but implicitly convey which job to be run on which machine as shown in **Figure 3**.

The chromosome is divided into four equal parts as shown in **Figure 3**. The parts are numbered as Machine 1, Machine 2, Machine 3 and Machine 4 from left to right. The numbers in each part represent the job numbers from 1 to 4.

The decoding of the genes of the chromosome shown in **Figure 3** is shown in **Table 8**. The decoding of the genes for the machine is explained below.

1. In the first part of the chromosome, the very first job number is 2. Hence, 1 is entered in the row 2 under the gene position of the column for Machine 1.
2. Then randomly select an unassigned operation (operation number) of the job 2 and assign it to the machine 1.
3. Next, from the same part, the second job is 3. So, the gene position 2 is entered in the row 3 under the gene position of the column for Machine 1.
4. Then randomly select an unassigned operation (operation number) of the job 3 and assign it to the machine 1.
5. The next job is 1 at the gene position 3 of the part 1 of the chromosome. So, the gene position 3 is entered in the row 1 under the gene position of the column for Machine 1.
6. Then randomly select an unassigned operation (operation number) of the job 1 and assign it to the machine 1.
7. Next, the gene position 4 which contains the job 4. So, the gene position 4 is entered in the row 4 under the gene position of the column for Machine 1.
8. Then randomly select an unassigned operation number of the job 4 and assign it to the machine 1.

In the same way, the decoding of the genes for the rest of the machines can be carried out. The final decoded sequence of the operations of the jobs based on **Table 8** is shown in **Table 9**.

4. Crossover Operator

Crossover operator is an important step in genetic algorithm, where the parent chromosomes are taken in pair and their genes are exchanged in certain order to obtain off spring. These offspring become next generation

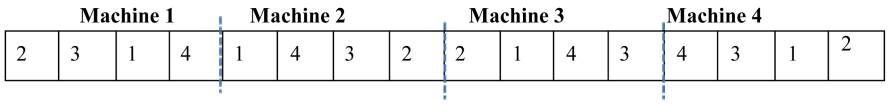

Figure 3. A typical chromosome for an OSSP of size 4 × 4 with implicit permutation coding.

Table 8. Decoded genes in terms of operation number of machines.

Job	Machine 1		Machine 2		Machine 3		Machine 4	
	Gene position	Operation	Gene position	Operation	Gene position	Operation	Gene position	Operation
1	3	4	1	3	2	1	3	2
2	1	2	4	1	1	4	4	3
3	2	1	3	2	4	3	2	4
4	4	3	2	4	3	2	1	1

Table 9. Sequence of jobs.

Job	Sequence of operations
1	4 – 3 – 1 – 2
2	2 – 1 – 4 – 3
3	1 – 2 – 3 – 4
4	3 – 4 – 2 – 1

parent chromosomes [4] [16]. This paper uses two-digit subscripts encoding to describe different crossover operators, which are as listed below.

- One-point Crossover
- Two-point Crossover
- Linear Order Crossover (LOX)
- Cycle Crossover
- Position-based Crossover
- Order-based Crossover
- Partially Mapped Crossover (PMX) pcxd.

4.1. One-Point Crossover

The steps of one-point crossover method are listed below.

a) Mark one point in the chromosome to create two substrings of genes in Parent Chromosome 1 (PC1) as shown in **Figure 4**.

b) Copy the substring 1 from PC1 to Offspring 1 (OF1) as it is. At the same time cancel the corresponding genes in PC2, which are shown with grey boxes in **Figure 4**.

c) The non-grey cells are copied to OF1 in linear order to fill substring 2 of OF1 as shown in **Figure 4**.

4.2. Two-Point Crossover

The steps of two-point crossover method are listed below.

a) Mark two points randomly in the chromosome to create three substrings of genes, viz. Substring 1, Substring 2 and Substring 3 as shown in **Figure 5**.

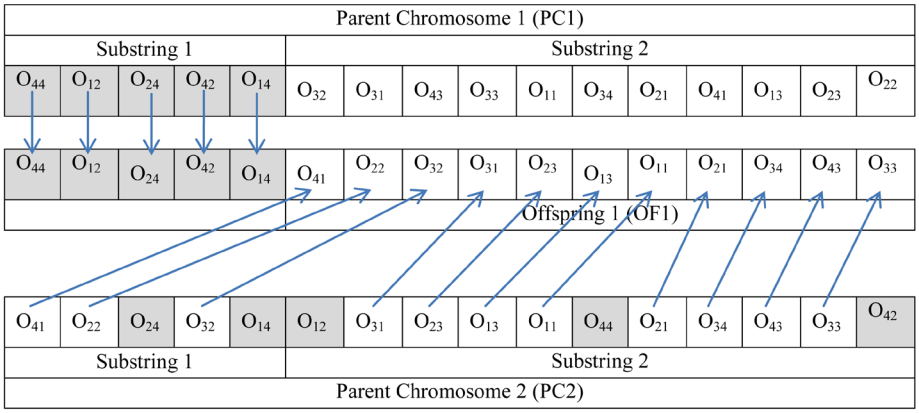

Figure 4. Diagrammatic illustration of one-point crossover method.

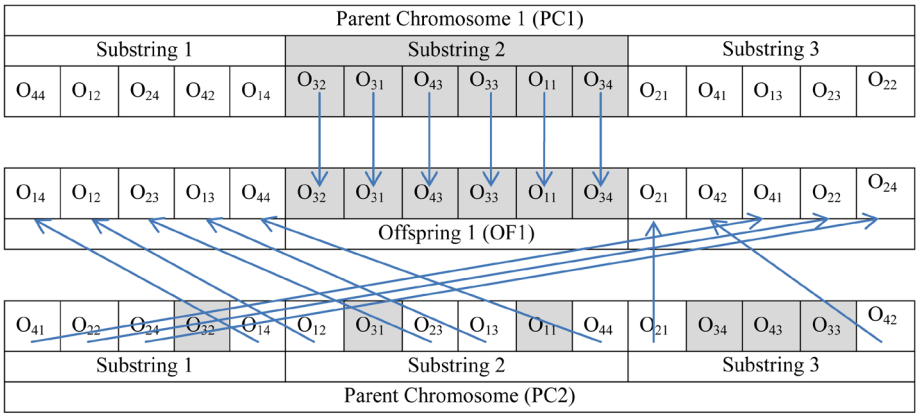

Figure 5. Diagrammatic illustration of two-point crossover.

b) Copy the Substring 2 from Parent Chromosome 1 (PC1) to Offspring 1 (OF1) as it is. At the same time cancel the corresponding genes in PC2 as shown with grey boxes in **Figure 5**.

c) Start filling from the left side of the substring 3 of OF1 with genes starting from non-grey cells of substring 3 of PC2 and then substring 1 and substring 2 in cyclic order. The non-grey cell after the second cut point of PC2 will be the first gene after the second cut point of OF1 to be filled and so on as shown in **Figure 5**.

4.3. Linear Order Crossover (LOX)

The steps of linear order cross over method are presented below.

a) Mark two points randomly to create three substrings, Substring 1, Substring 2 and Substring 3of genes as shown in **Figure 6**.

b) Copy the Substring 2 from Parent Chromosome 1 (PC1) to Offspring 1 (OF1) as it is. At the same time cancel the corresponding genes in PC2 as shown with grey boxes in **Figure 6**.

c) The non-grey cells are copied to OF1 in linear order (from left to right) to fill the Substring 1 and Substring 3 of OF1 as shown in **Figure 6**.

4.4. Cycle Crossover

The steps of cyclic crossover method are presented below:

a) From the two parent chromosomes PC1 and PC2 and find a cycle starting from first gene of PC1 as shown in **Figure 7**.

$O_{44} \rightarrow O_{33} \rightarrow O_{32} \rightarrow O_{21} \rightarrow O_{22} \rightarrow O_{43} \rightarrow O_{11} \rightarrow O_{31} \rightarrow O_{42} \rightarrow O_{41} \rightarrow O_{44}$

Here, the cycle begins with O_{44} reaches O_{33} and then O_{32} and so on till it ends with O_{44} again.

b) Mark the genes in grey colour that form the cycle in both parent chromosomes PC1 and PC2 as shown in **Figure 7** and **Figure 8**.

c) Copy the non-grey cells of PC2 to non-grey cells of PC1 to get the first offspring OF1 as shown in **Figure 8**.

d) Copy the non-grey cells of PC1 to non-grey cells of PC2 to get the second offspring OF2 as shown in **Figure 9**.

4.5. Position-Based Crossover

The working logic of position based crossover method is presented below.

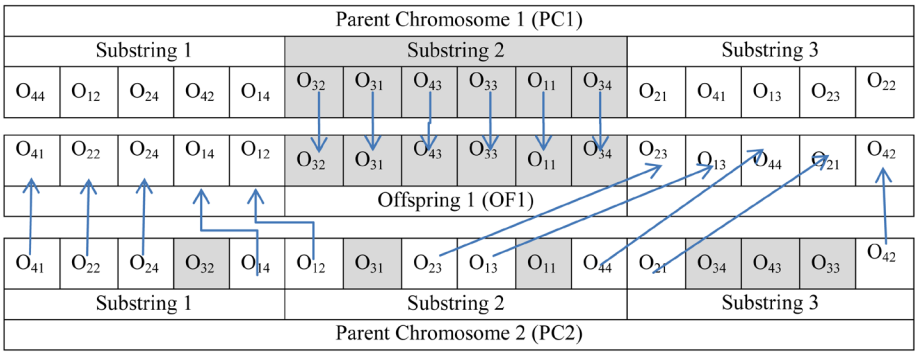

Figure 6. Diagrammatic illustration of linear order crossover (LOX).

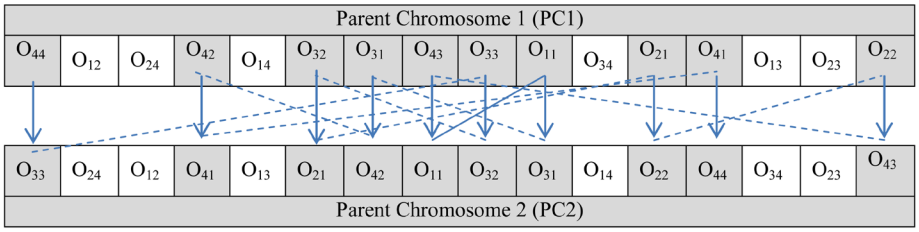

Figure 7. Diagrammatic illustration of cycle crossover operator.

Figure 8. Generating OF1 using cycle crossover operator.

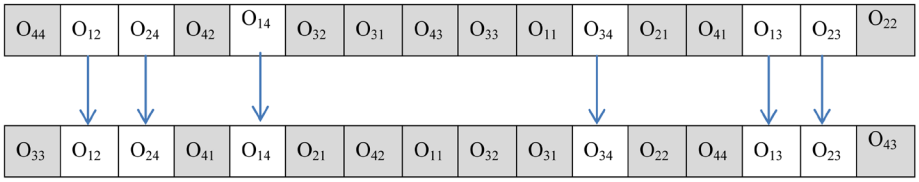

Figure 9. Generating OF2 using cycle crossover operator.

a) Randomly select some genes, which are marked with grey cells in PC1 and copy them to OF1 as it is as shown in **Figure 10**.

b) Mark the selected genes of PC1 in PC2 with grey cells.

c) Copy the non-grey cells of PC2 onto non-grey cells of OF1 in linear order (from left to right) as shown in **Figure 10**.

4.6. Order-Based Crossover or Order Crossover

The crossover method is a slight modification of the position-based crossover operator. The steps of this method are presented below.

a) Randomly select some genes, which are as marked with grey cells, from PC1 and mark the corresponding genes in PC2with grey cells as shown in **Figure 11**.

b) Copy the non-grey cells of PC2 onto OF1 as it is.

c) Fill the balance cells of OF1 with the genes, which are marked with grey colour in PC1 in linear order (from left to right).

4.7. Partially Mapped Crossover (PMX)

The steps of the partially mapped crossover method are listed below.

a) Mark two points on PC1 and PC2 that will create three substrings, viz. Substring 1, Substring 2 and Substring 3, as shown in **Figure 12**.

b) Copy Substring 1 and Substring 3 of PC1 to OF1 and Substring 2 of PC2 to OF1.

c) This will result in duplicity of genes O_{12}, O_{23}, O_{13}, and O_{44}. These genes appear twice in OF1. To escape from duplicity, we partially map the genes in Substring 2 of PC1 and PC2. This is equivalent to find the union of the genes in the substring 2 of PC1 and the genes in the substring 2 of PC2.

O_{12} can be replaced by O_{32}; O_{23}can be replaced by O_{43}; O_{13} can be replaced by O_{33} and O_{44} can be replaced by O_{34}.

d) This will yield legal OF1 without any duplicity as shown in **Figure 12**. The Offspring 1 (OF1) after the partial mapping replacements is as shown in **Figure 13**.

5. Three-Chromosome Juggling Crossover (TCJC) Operator (Proposed Crossover Operator)

The three chromosome juggling crossover (TCJC) operator is classified into TCJC forward and TCJC backward.

5.1. TCJC Forward

Three-chromosome Juggling Crossover (TCJC Forward) Operator can be used for chromosomes which are encoded using two-digit encoding or two-digit subscripts encoding methods. This TCJC operator works as follows:

a) Let there be three parent chromosomes, which are labelled as two odd chromosomes (PC1 and PC3) and

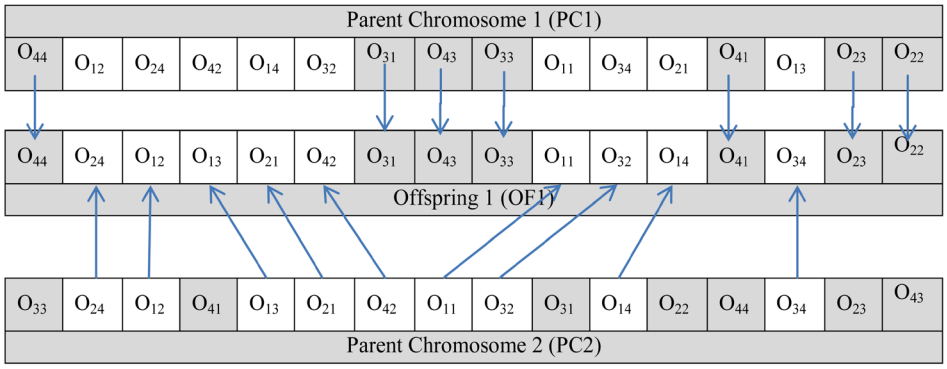

Figure 10. Diagrammatic illustration of position-based crossover operator.

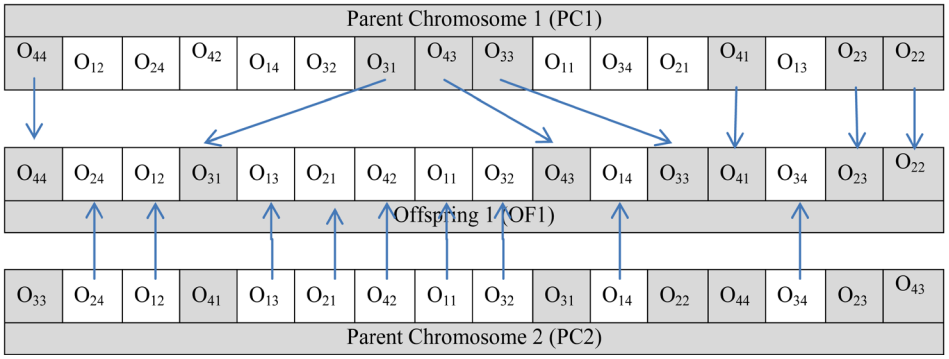

Figure 11. Diagrammatic illustration of order-based crossover operator.

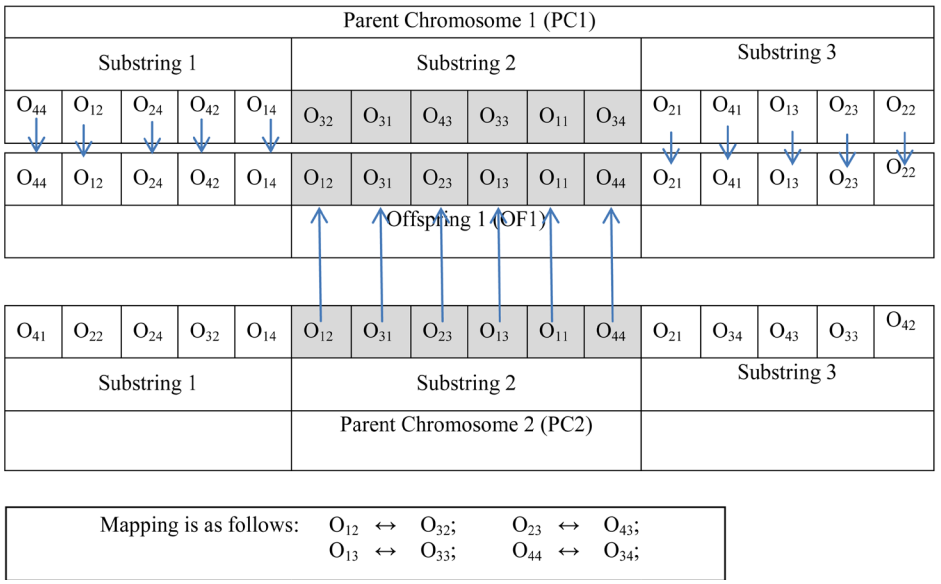

Figure 12. Diagrammatic illustration of preliminary stage partially mapped crossover operator.

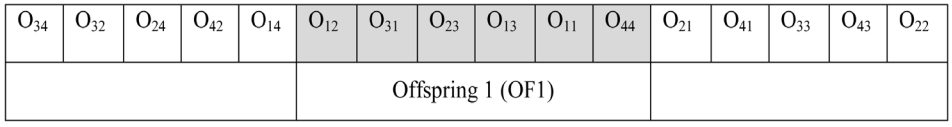

Figure 13. Offspring1 after replacing mapped genes.

one even chromosome (PC2) in the first pick of three chromosomes from the population of chromosomes as shown in the ovals in **Figure 15**.

b) For the first offspring OF1, use the subscripts of 3rd parent chromosome PC3. If the sum of the subscripts of third chromosome is even, pick the first available gene from left of chromosome PC2 (even chromosome) and contribute it to OF1. Then cancel the gene in both the chromosomes (PC1 and PC2) to avoid duplicity. If the sum of the subscripts of third chromosome PC3 is odd, then pick the first available gene from left of the chromosome PC1 (odd chromosome) and contribute to OF1.Then cancel the gene in both the chromosomes (PC1 and PC2) to avoid duplicity. These are explained using respective conditions as follows.

For all O_{ij} of third chromosome PC3:

If i + j = even, pick first available gene from left of the chromosome PC2 (even chromosome),

If i + j = odd, pick first available gene from left of the chromosomePC1 (odd chromosome).

The implementation of these steps for each of the genes of the chromosome PC3 is explained below.

i) First gene in third chromosome is O_{11}. Since the sum of subscripts of this gene is even, pick the first available gene from left of the even chromosome PC2 and place it as the first gene of the offspring OF1.

ii) Cancel the gene O11 in both the chromosomes (PC1 and PC2) to indicate that this gene has already been selected for OF1 as shown in **Figure 14**.

iii) Starting from the second gene of the odd chromosome PC3 onwards, continue to perform Step i and Step ii.

iv) To get Offspring 2 (OF2) from the same set of chromosomes PC1, PC2 and PC3, juggle the chromosomes such that sum of subscripts of PC1 will decide the gene to be picked from PC2 (even chromosome) or PC3 (odd chromosome). The process of obtaining OF2 is shown in **Figure 15**.

v) Two offspring OF1 and OF2 are generated from first three parent chromosomes PC1, PC2 and PC3. Next two offspring OF3 and OF4 will be obtained by juggling PC2, PC3 and PC4 and ignoring PC1 and so on.

5.2. TCJC Backward

Three-chromosome Juggling Crossover (TCJC backward) Operator can be used for chromosomes which are

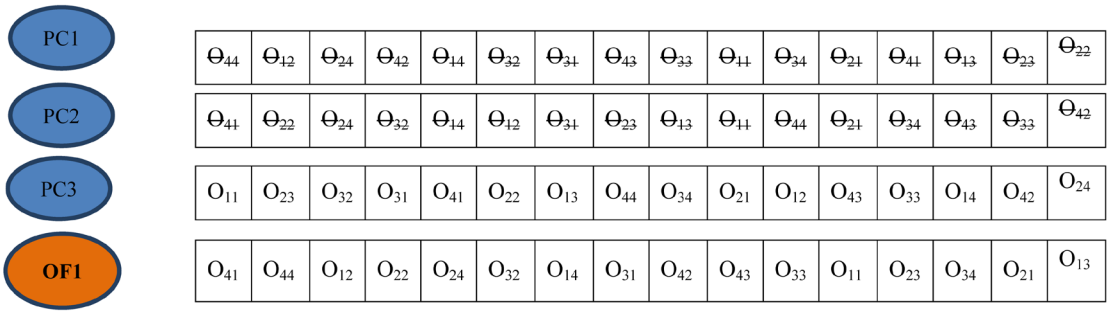

Figure 14. Diagrammatic illustration of a proposed new crossover operator: three-chromosome juggling crossover (TCJC) forward operator—OF1.

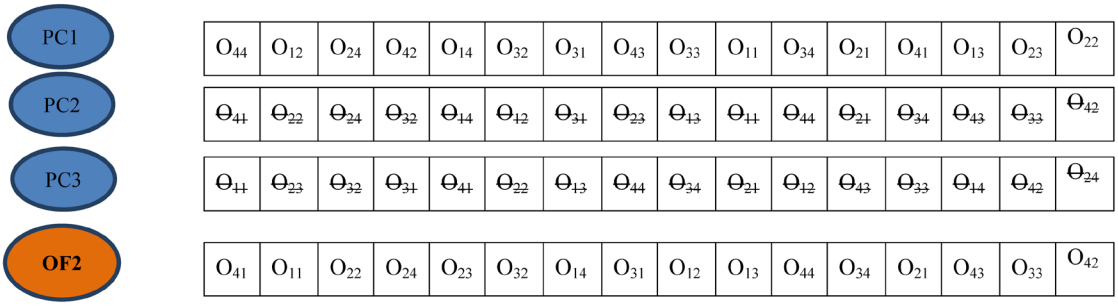

Figure 15. Diagrammatic illustration of a proposed new crossover operator: three-chromosome juggling crossover (TCJC) forward operator—OF2.

encoded using two-digit encoding or two-digit subscripts encoding methods. This TCJC backward operator works as follows:

a) Let there be three parent chromosomes, which are labelled as two odd chromosomes (PC1 and PC3) and one even chromosome (PC2) in the first pick of three chromosomes from the population of chromosomes as shown in the ovals in **Figure 15**.

b) For the first offspring OF1, use the subscripts of 3rd parent chromosome PC3. If the sum of the subscripts of third chromosome is even, pick the first available gene from right of chromosome PC2 (even chromosome) and contribute it to OF1. Then cancel the gene in both the chromosomes (PC1 and PC2) to avoid duplicity. If the sum of the subscripts of third chromosome PC3 is odd, then pick the first available gene from right of the chromosome PC1 (odd chromosome) and contribute to OF1.Then cancel the gene in both the chromosomes (PC1 and PC2) to avoid duplicity. These are explained using respective conditions as follows.

For all O_{ij} of third chromosome PC3:

If i + j = even, pick first available gene from right of the chromosome PC2 (even chromosome),

If i + j = odd, pick first available gene from right of the chromosomePC1 (odd chromosome).

The implementation of these steps for each of the genes of the chromosome PC3 is explained below.

i) First gene in third chromosome is O_{11}. Since, the sum of subscripts is even, pick the first available gene from right of the even chromosome PC2 and place it as the first gene of the Offspring OF1.

ii) Cancel the gene O11 in both the chromosomes (PC1 and PC2) to indicate that this gene has already been selected for OF1 as shown in **Figure 16**.

iii) Starting from the second gene of the odd chromosome PC3 onwards, continue to perform Step i and Step ii.

iv) To get Offspring 2 (OF2) from the same set of chromosomes PC1, PC2 and PC3, juggle the chromosomes such that sum of subscripts of PC1 will decide the gene to be picked from PC2 (even chromosome) or PC3 (odd chromosome). The process of obtaining OF2 is shown in **Figure 17**.

v) Two offspring OF1 and OF2 are generated from first three parent chromosomes PC1, PC2 and PC3. Next two offspring OF3 and OF4 will be obtained by juggling PC2, PC3 and PC4 and ignoring PC1 and so on.

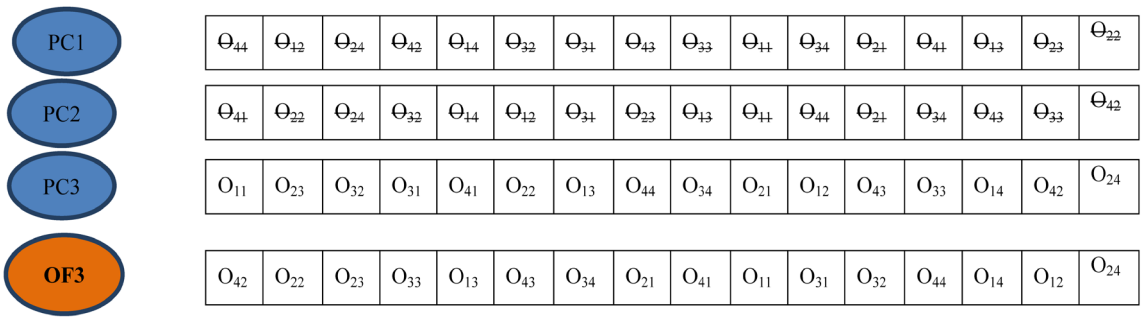

Figure 16. Diagrammatic illustration of a proposed new crossover operator: three-chromosome juggling crossover (TCJC) backward operator—OF1.

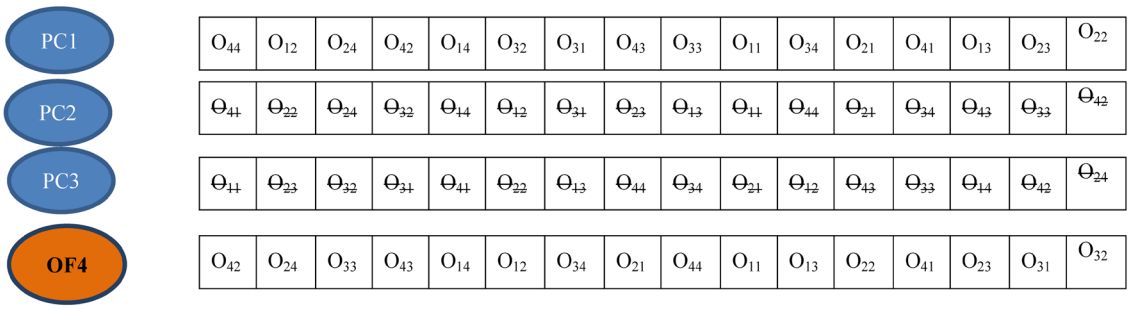

Figure 17. Diagrammatic illustration of a proposed new crossover operator: three-chromosome juggling crossover (TCJC) backward operator—OF2.

6. Demonstration

This section presents the workings of TCJC forward and TCJC backward operators using numerical examples.

6.1. TCJC Forward

Consider an open shop scheduling problem with 4 jobs and 4 machines as shown in **Table 10** to explain TCJC forward operator.

Let the parent chromosome PC1 be as shown in **Figure 18**. The details of computation of C_{max} using the genes of the parent chromosome PC1 are shown in **Table 11**. The makespan of scheduling the operations as the genes of the parent chromosome PC1 is 519.

Now, consider another chromosome known as parent chromosome PC2 as shown in **Figure 19**. The makespan of scheduling the genes of the parent chromosome PC2 is 397. Now, consider another chromosome known as parent chromosome PC3 as shown in **Figure 20**. The makespan of scheduling the genes of the parent chromosome PC2 is 375.

The determination of offspring 1 (OF1) using *Three-Chromosome Juggling Crossover Operator—Forward* operator applied to PC1, PC2 and PC3 is shown in **Figure 21**. The makespan of this offspring is 495.

The determination of offspring 2 (OF2) using *Three-Chromosome Juggling Crossover Operator—Forward* operator applied to PC1, PC2 and PC3 is shown in **Figure 22**. The makespan of this offspring is 476.

Table 10. Data of open shop problem with 4 jobs and 4 machines.

	Job1	Job2	Job3	Job4
Machine 1	85	23	39	55
Machine 2	85	74	56	78
Machine 3	3	96	92	11
Machine 4	67	45	70	75

Table 11. Details of computation of C_{max} using parent chromosome PC1.

Notation	Machine	Job	Processing time	Start Time	Finish Time
O_{44}	4	4	75	0	75
O_{12}	1	2	23	0	23
O_{24}	2	4	78	75	153
O_{42}	4	2	45	75	120
O_{14}	1	4	55	153	208
O_{32}	3	2	96	120	216
O_{31}	3	1	3	216	219
O_{43}	4	3	70	120	190
O_{33}	3	3	92	219	311
O_{11}	1	1	85	219	304
O_{34}	3	4	11	311	322
O_{21}	2	1	85	304	389
O_{41}	4	1	67	389	456
O_{13}	1	3	39	311	350
O_{23}	2	3	56	389	445
O_{22}	2	2	74	445	**519**

O_{44}	O_{12}	O_{24}	O_{42}	O_{14}	O_{32}	O_{31}	O_{43}	O_{33}	O_{11}	O_{34}	O_{21}	O_{41}	O_{13}	O_{23}	O_{22}

Figure 18. Parent chromosome PC1.

O_{41}	O_{22}	O_{24}	O_{32}	O_{14}	O_{12}	O_{31}	O_{23}	O_{13}	O_{11}	O_{44}	O_{21}	O_{34}	O_{43}	O_{33}	O_{42}

Figure 19. Parent chromosome PC2.

O_{11}	O_{23}	O_{32}	O_{31}	O_{41}	O_{22}	O_{13}	O_{44}	O_{34}	O_{21}	O_{12}	O_{43}	O_{33}	O_{14}	O_{42}	O_{24}

Figure 20. Parent chromosome PC3.

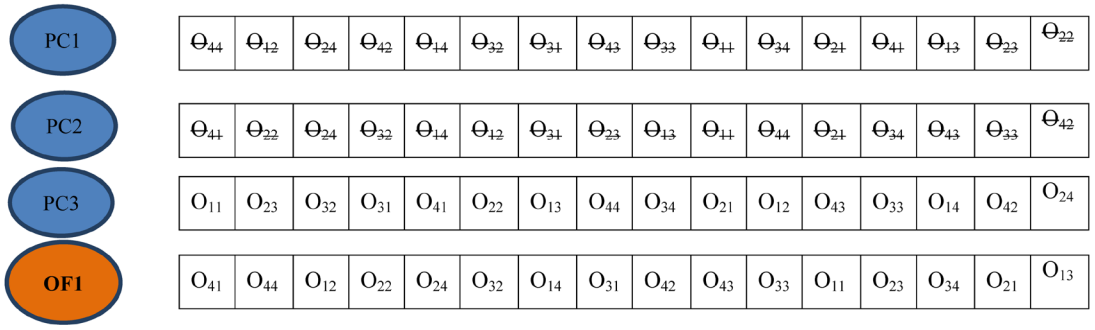

Figure 21. Determination of offspring OF1 using TCJC forward operator.

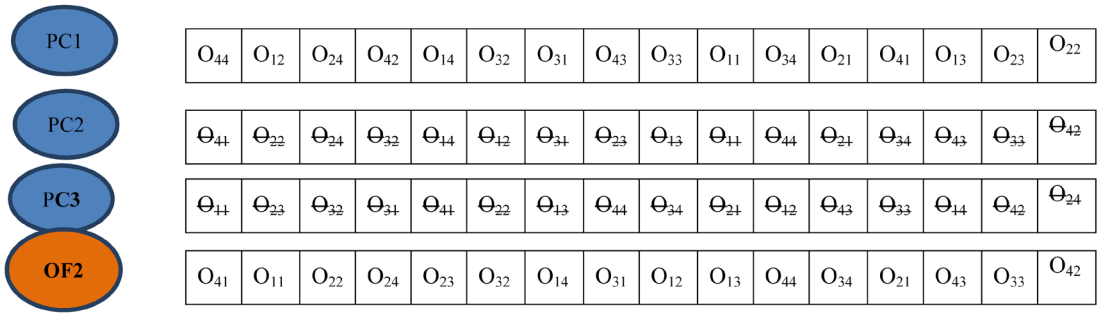

Figure 22. Determination of offspring OF2 using TCJC forward operator.

6.2. TCJC Backward

Two more off-springs can be obtained by using TCJC operator operating from backward direction. Let them be OF1 and OF2. The off-spring 1 (OF1) is generated as shown in **Figure 23**. Its makespan is 761. The offspring 2 (OF2) generated is shown in **Figure 24**. Its makespan is 466.

The demonstration has considered only 3 chromosomes. The results show that the makespan values are different for different for different chromosomes as well as for different offspring. It is important to note that at least 20 × m × n number of chromosomes should be randomly chosen as the initial population, where 'm' is number of machines and "n" is number of jobs. A sample of three chromosomes will form a set as shown in the demonstration to produce 4 off-springs.

7. Conclusion

This paper covers important crossover operators with the perspective of solving open shop scheduling problem to minimize makespan. One of the objectives of this paper was to develop a new crossover operator. This paper limits to providing a conceptual framework of newly developed crossover operator namely *Three Chromosome*

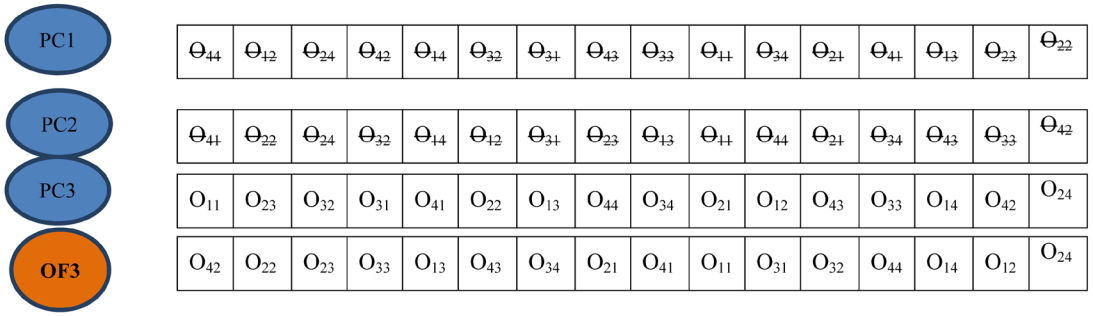

Figure 23. Determination of offspring OF1 using TCJC backward operator.

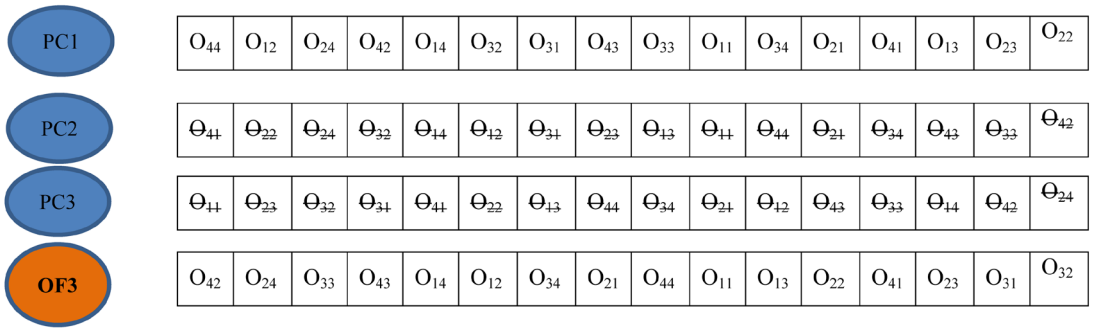

Figure 24. Determination of offspring OF2 using TCJC backward operator.

Juggling Crossover Forward Operator and *Three Chromosome Juggling Crossover Backward Operator* and not providing any numerical work to compare different crossover operators to select the best one. This paper provides an insight into a new type of crossover operator that can be used while applying genetic algorithm to solve open shop scheduling problem to minimize makespan.

8. Limitation of Research

The research paper makes an attempt but limits to propose the new crossover operator for only the Open Shop Scheduling problems. This limitation provides an opportunity to try for Job Shop and Flow Shop Scheduling problems too.

References

[1] Panneerselvam, R. (2012) Production and Operations Management. Third Edition, PHI Learning, New Delhi.

[2] Panneerselvam, R. (2006) Operations Research. Second Edition, PHI Learning, New Delhi.

[3] Panneerselvam, R. (1999) Heuristic for Moderated Job Shop Scheduling Problem to Minimize Makespan. *Industrial Engineering Journal*, **28**, 26-29.

[4] Panneerselvam, R. (2016) Design and Analysis of Algorithms. Second Edition, PHI Learning, New Delhi.

[5] Anand, E. and Panneerselvam, R. (2015) Literature Review of Open Shop Scheduling Problem. *Intelligent Information Management*, **7**, 33-52. http://dx.doi.org/10.4236/iim.2015.71004

[6] Senthilkumar, P. and Shahabudeen, P. (2006) GA Based Heuristic for the Open Job Shop Scheduling Problem. *International Journal of Advanced Manufacturing Technology*, **30**, 297-301. http://dx.doi.org/10.1007/s00170-005-0057-2

[7] Louis, S.J. and Xu, Z.J. (1996) Genetic Algorithms for Open Shop Scheduling and Re-Scheduling. *ISCA* 11*th International Conference on Computers and Their Applications*, San Francisco, 7-9 March 1996, 99-102.

[8] Zobolas, G.I., Tarantilis, C.D. and Ioannou, G. (2009) Solving the Open Shop Scheduling Problem via a Hybrid Genetic-Variable Neighborhood Search Algorithm. *Cybernetics and Systems*: *An International Journal*, **40**, 259-285. http://dx.doi.org/10.1080/01969720902830322

[9] Taillard, E. (1993) Benchmarks for Basic Scheduling Problems. *European Journal of Operational Research*, **64**, 278-285. http://dx.doi.org/10.1016/0377-2217(93)90182-M

[10] Brucker, P., Hurink, J., Jurisch, B. and Wostmann, B. (1997) A Branch and Bound Algorithm for the Open Shop Problem. *Discrete Applied Mathematics*, **76**, 43-59. http://dx.doi.org/10.1016/S0166-218X(96)00116-3

[11] Gueret, C. and Prins, C. (1999) A New Lower Bound for the Open Shop Problems. *Annals of Operations Research*, **92**, 165-183. http://dx.doi.org/10.1023/A:1018930613891

[12] Kokosinski, Z. and Studzienny, L. (2007) Hybrid Genetic Algorithms for the Open Shop Scheduling Problem. *International Journal of Computer Science and Network Security*, **7**, 136-145.

[13] Liaw, C.F. (2000) A Hybrid Genetic Algorithm for the Open Shop Scheduling Problem. *European Journal of Operational Research*, **124**, 28-42. http://dx.doi.org/10.1016/S0377-2217(99)00168-X

[14] Andresen, M., Brasel, H., Marc, M., Tusch, J., Werner, F. and Willenius, P. (2008) Simulated Annealing and Genetic Algorithms for Minimizing Mean Flow Time in an Open Shop. *Mathematical and Computer Modelling*, **48**, 1279-1293. http://dx.doi.org/10.1016/j.mcm.2008.01.002

[15] Matta, M.E. (2009) A Genetic Algorithm for the Proportionate Multiprocessor Open Shop. *Computers & Operations Research*, **36**, 2601-2618. http://dx.doi.org/10.1016/j.cor.2008.11.009

[16] Panneerselvam, R. (2012) Research Methodology. Second Edition, PHI Learning, New Delhi.