Performances' study on crossover operators keeping good schemata for some scheduling problems

Marie-Claude Portmann
INRIA-LORIA, MACSI Team
Ecole des Mines de Nancy, Parc de Saurupt
54042 Nancy Cedex, FRANCE
Phone: 33-3-83-58-41-85
portmann@mines.u-nancy.fr

Antony Vignier
INRIA-LORIA, MACSI Team
Campus scientifique, BP 239
54506 Vandoeuvre les Nancy Cedex, FRANCE
Phone: 33-3-83-59-30-40
email: avignier@loria.fr

Abstract

In this paper, we deal with genetic algorithms (GAs) solving some permutation scheduling problems. Considering only the permutation codes and only the crossover-phase of a GA on a very large population, a comparison study of different crossover operators was done earlier according to some performance indicators. Here, we develop a tool generating a large scale of scheduling instances and of permutation codes (i.e. the initial population). We again run only the crossover-phase of a GA on this very large population with some crossover operators, and they are considered independently. This new comparison is made on permutation scheduling problems. Statistics are then computed in order to validate or not the results obtained previously with the performance indicators.

1 INTRODUCTION

The schemata theory of Holland, in 1975, explains why the simplest genetic algorithm, called SGA in (Goldberg, 1989), converges up to a population where every individual is good. In fact, this theory provides an answer to the question "In which ways a string is a representative of other string classes with similarities at some string position ?". This theory uses some strong hypotheses: the binary encoding for genes, the use of the classical one-point crossover operator (the beginning of the string of the first mate completed with the end of the string of the second mate), no strong relation between 2 genes not closed together in the chromosome. With these assumptions, chromosomes that represent good schemata (i.e with a good fitness average) are more and more numerous in the population during iterations.

When the above assumptions are not verified, other characteristics associated to the chromosomes are needed. They can be identified to the schemata so that, the number of good chromosomes inside the population is increasing when good characteristics are kept by the crossover operators.

In order to try to extend the schemata theory to more difficult problems, we have to tackle with two difficulties: to define the good characteristics that must be kept for a given problem and for the corresponding chosen encoding, and to design crossover operators that are able to keep good characteristics for a given problem and for the corresponding chosen encoding.

Since 1994 we have been focusing our researches on concrete problems for which solutions can be described with orders or permutations of n elements. These problems are called permutation problems. The Travelling Salesman Problem (TSP) is the most famous one of this family. One machine scheduling and flow-shop scheduling problems are also other well-known scheduling problems. Hence, we consider this problem family and in order to find the good set of characteristics of the chromosomes, we ask the following question: for a given problem and for a given criterion, how define the genes' values or the links between the genes' values, so that a created individual is good and thus must be kept. For the TSP, the answer may be "*keep the small edges of both mates*". For the one-machine scheduling problem and considering the weighted tardiness criterion, the answer may be "*keep the relative partial orders of each couple of operations as much as possible*". In order to measure how much edges or relative operation positions crossover operators are able to keep, (Djerid, Portmann and Villon, 1996) designed a set of performance indicators. Their role is to compare the ability of each permutation crossover operator to keep characteristics contained in the mates.

(Djerid and Portmann, 1999) give some analytical and experimental results that allowed us to compare a set of crossover operators. In this previous work, only permutations were considered; more precisely, no concrete problems have been taken into account. Two conclusions have been obtained: either"a given crossover operator has good performances for a given indicator and bad performances for the other indicators" or "a given crossover is a good tradeoff for two indicators". This latter case is very interesting, for complex problems, two sets of good characteristics may be kept simultaneously.

Nevertheless, it has not been proved that if a crossover operator is efficient to keep the good characteristics of the permutation encoding, it is also good for the associated concrete problem that has to be solved.

Thus this paper completes the previous works, verifying whether what is interesting to keep for permutations is also interesting to be kept for a concrete problem in which two sets of characteristics must be kept. The paper is organised as follows. Section 2 presents the context and previous works that is to say performances indicators, crossover operators and results. Section 3 is dedicated to a new development on a particular permutation scheduling problem we considered here. The tool, the set of experimentation and the results obtained are then presented. Finally we conclude.

2 CONTEXT AND PREVIOUS WORKS

2.1 PERFORMANCE INDICATORS

Each indicator is a distance between the created offspring and their parents. We normalize every indicator so that their values belong to the interval [0, 100]. In order to keep the interesting characteristics of the mates, the first indicators have to be maximized, while the latter must be minimized.

2.1.1 Edge Based Indicators

They are appropriate for TSPs or some one-machine problems with sequence dependent set-up times. The interesting characteristic of the mates is either the symmetric or asymmetric distance between two consecutive values of the permutation (including the distance between the points situated in the extreme positions for the TSP). If the corresponding distances are small in the mates, then the mates are good. By keeping as many edges (or arcs) as possible from the good mates, the offspring might also be good. The two indicators proposed count the edges or the arcs kept from the mates and are designed as follows.

We say that (v_1, v_2) are consecutive values inside a given chromosome if, in this permutation, v_1 immediately precedes v_2, or v_2 immediately precedes v_1, or v_1 is in the last position n and v_2 in the first position (i.e. 1), or v_2 in the last position n and v_1 in the first position.

To count the edges kept, we denote by $NES(v_1, v_2)$ an integer function whose value is equal to 2 if (v_1, v_2) are consecutive values in parent 1, in parent 2 and in the offspring created by the crossover operator applied to parent 1 and parent 2. $NES(v_1, v_2)$ is equal to 1 if (v_1, v_2) are consecutive values in parent 1 or in parent 2, and in the offspring created by the crossover operator applied to parent 1 and parent 2. In the other case, $NES(v_1, v_2)$ is equal to 0.

We denote by $NEST$ the sum of $NES(v_1, v_2)$ for all couples (v_1, v_2) with $v_1 < v_2$. The first indicator (Symmetric Edge Based Indicator) $SEBI$ is equal to $NEST * 100/(2n)$.

To count the arcs kept, we denote by $NE(v_1, v_2)$ an integer function whose value is equal to 2 if (v_1, v_2) are consecutive values with v_1 preceding v_2 in parent 1, in parent 2 and in the created offspring. $NE(v_1, v_2)$ is equal to 1 if (v_1, v_2) are consecutive values with v_1 preceding v_2 in parent 1 or in parent 2, and in the created offspring. In the other cases, $NE(v_1, v_2)$ is equal to 0.

We denote by NET the sum of the $NE(v_1, v_2)$ for all couples (v_1, v_2) with $v_1 \neq v_2$. The second indicator (Edge Based Indicator) EBI is equal to $NET * 100/(2n)$.

$SEBI$ (resp. EBI) is equal to 0 when the created offspring contains no edges (resp. arcs) existing in both parents. $SEBI$ and EBI are equal to 100 when parent 1, parent 2 and the considered created offspring are identical. These indicators must be maximized. We are here interested in the mean value of these indicators when a given crossover operator is applied to a large random set of mates.

2.1.2 Precedence Constraint Based Indicators

We denote by $NPC(v_1, v_2)$ an integer function whose value is equal to 4 if v_1 precedes v_2 in parent 1 and in parent 2 and in the created offspring. $NPC(v_1, v_2)$ is equal to 1 if v_1 precedes v_2 in parent 1 or in parent 2 and in the created offspring. In the other cases, $NPC(v_1, v_2)$ is equal to 0.

We denote by $NPCT$ the sum of the $NPC(v_1, v_2)$ for all couples (v_1, v_2) with $v_1 < v_2$. The third indicator (Precedence Constraint Based Indicator) $PCBI$ is equal to $[NPCT * 100]/[2n(n-1)]$.

This indicator must be maximized in order to keep as many precedence constraints existing in the parents as possible. In addition to the precedence constraint based indicators, a particular property may be verified by some crossover operators.

Property 1 *If job i precedes job j in both parent 1 and parent 2, then job i precedes job j in the created offspring.*

This property is very useful for scheduling problems: either because precedence constraints may exist (and must be verified) between some operations of the sequence in the definition of the scheduling problems, or because in some scheduling problems, it may be proved that there exists at least one optimal solution in which job i precedes job j and this for many couples of jobs (for example, these precedence constraints may be added to the one-machine scheduling problem with mean tardiness criterion by applying Emmons dominance properties (Emmons, 1969), (Djerid, Portmann and Villon 1996). In this latter case, the solution space to be explored for finding an optimal solution may be considerably reduced. If property 1 is always verified for a given crossover operator, then $PCBI$ is maximized or at least very good.

2.1.3 Position Based Indicators

Here, we consider the simplest distance between two permutations. We assume that the rank of the elements inside a permutation was good and that the rank in the created offspring must not be too far from the rank in the mates. We denote by $PP_i(k)$ (resp. $PO_i(k)$), $i \in (1,2)$, the position of the element k in parent i (resp. offspring i). We denote by PB_{ij}, $i \in (1,2)$, $j \in (1,2)$, the euclidean distance computed on two vectors which gives the position of each gene in offspring j and parent i. The fourth indicator (Position Based Indicator) PBI is given by equation 1. This last indicator must be minimized.

$$PBI = \frac{\sum_{i=1}^{2}\sum_{j=1}^{2} PB_{ij}}{4} \times \frac{6}{n(n+1)(2n+1)} \qquad (1)$$

2.2 CROSSOVER OPERATORS

In this section, we use letters to name jobs to be scheduled.

2.2.1 OX Operator

(Davis, 1985) proposed the OX operator for Order Cross operator. It was designed for the traveling salesman problems and then (Oliver, Smith and Holland, 1987) modified it. OX crosses orders encoded directly as permutations, named "permutation encoding". The algorithm is given below and an example is given in Table 1.

1. The cross part of mate 2 (resp. 1) is copied into the cross part of offspring 1 (resp. 2).
2. The remaining elements of mate 1 (resp. 2) are copied in the empty genes of offspring 1 (resp. 2) in the order of mate 1 (resp. 2) beginning by the right part and finishing by the left part.

Table 1: Designing of offspring 1 and 2 for OX

mate 1	a	b	c	d	e	f	g	h	i	j
mate 2	c	b	a	g	h	i	j	f	d	e
offspring1	e	f	j	g	h	i	a	b	c	d
offspring2	h	i	j	d	e	f	c	b	a	g

2.2.2 LOX Operator

(Falkenauer and Bouffouix,1991) proposed the LOX operator, for Linear Order Cross-over. It is a modified version of OX proposed to solve job-shop scheduling problems. They are "two-point" cross-over operators. As OX, LOX works on permutation encoding. LOX is similar to OX, but left and right are exchanged in phase 2. An example is given in Table 2.

Table 2: Designing of offspring 1 and 2 for LOX

mate 1	a	b	c	d	e	f	g	h	i	j
mate 2	c	b	a	g	h	i	j	f	d	e
offspring1	a	b	c	g	h	i	d	e	f	j
offspring2	c	b	a	d	e	f	g	h	i	j

2.2.3 1X Operator

(Davis, 1985) proposed the 1X operator. It is the simplification of OX or LOX if a one-point cross-over is used instead of a two-point cross-over. It is interesting to consider it because it verifies property 1. In one part of the offspring the gene values of one parent is copied, in the other part the remaining gene values are copied in the order of the other parent. An example is given in Table 3.

Table 3: Designing of offspring for 1X

Mate 1	a	b	c	d	e	f	g	h	i	j
Mate 2	c	b	a	g	h	i	j	f	d	e
Offspring1	a	b	c	d	g	h	i	j	f	e
Offspring2	c	b	a	g	d	e	f	h	i	j
Offspring3	c	b	a	d	e	f	g	h	i	j
Offspring4	a	b	c	g	h	i	j	f	d	e

2.2.4 kX Operator

(Caux, Pierreval and Portmann, 1995), (Djerid, Portmann and Villon, 1996) proposed a generalization of the 1X operator with several cross-over points, named kX operator. Its interest is that it verifies also property 1. Nevertheless, for great values of k, the created offspring are too much similar to one of the parent. So, it is the reason why we only use here the 2X cross-over operator that is to say with k equal to 2. k cross-points are generated. Four offspring are generated. Offspring 1 (resp. 2) consists in copying the odd numbered sub strings of mate 1 (resp. 2) and reordering in the order of mate 2 (resp. 1) each even numbered sub string of mate 1 (resp. 2). Offspring 3 (resp. 4) is obtained in the same manner by reversing even and odd sub string roles. An example is given in Table 4.

Table 4: Designing of offspring for 2X

mate 1	a	b	c	d	e	f	g	h	i	j
mate 2	c	b	a	g	h	i	j	f	d	e
offspring1	a	b	c	f	d	e	g	h	i	j
offspring2	c	b	a	g	h	i	j	f	d	e
offspring3	c	b	a	d	e	f	g	h	i	j
offspring4	a	b	c	g	h	i	d	e	f	j

2.2.5 ERX Operator

(Whitley, Starkweather and Fuquay, 1989) proposed the ERX operator (for edge recombination cross-over). It was designed for the symmetric traveling salesman problems and have been used by many other authors. It tries to use the edges which are contained in both mates as much as possible. The algorithm to generate one offspring is given below and an example lies in Table 5.

1. Designing of the Edge table: we assign a list of its neighbors in mate 1 or mate 2 to each job (the sign - means that the corresponding job is a neighbor in both mates). The first and the last operation are considered as neighbors for the TSP.
2. An arbitrary first operation is chosen in the operation with the smallest list of neighbors and is called the current operation.
3. The following iterative algorithm is used:
   (a) Select an operation which is a neighbor of the current one and which has the fewest remaining neighbors (breaking ties randomly), arbitrary remaining operation if the current operation has no remaining neighbor.
   (b) The operation is added to the sequence and becomes the new current one.
   (c) If the n operations are not selected go to (a).

Table 5: Designing of the only offspring for ERX

mate 1	a	b	c	d	e	f	g	h	i	j
mate 2	c	b	a	g	h	i	j	f	d	e

Designing of the Edge table

a	-b	g	j	
b	-a	-c		
c	-b	d	e	
d	c	-e	f	
e	c	-d	f	
f	d	e	g	j
g	a	f	-h	
h	-g	-i		
i	-h	-j		
j	a	f	-i	

Designing of the Edge table

offspring	$\underline{b}$	$\underline{a}$	$\underline{j}$	i	h	g	f	$\underline{d}$	e	c

where $\underline{x}$ = random breaks ties

2.2.6 ARX and ARXM Operators

ARX operator for Arc Recombination Cross-over, has been proposed in (Djerid, 1997), as far as we know. It is similar to ERX by replacing the concept of edge by the concept of arc (i.e. for a given job only its successors in both mates are inserted in the corresponding line of the "Arc table"). Here, we slightly modify this operator by limiting to the three first jobs of one mate the random job which is put at the beginning of the offspring sequence. We will call this modified operator ARX for Arc Recombination for scheduling problem Cross-over. As a matter of fact, as far as tardiness is concerned, it was surely very bad to begin the sequence with a completely random job (we assume here that mates have good performances for the weighted tardiness criterion).

2.2.7 Random Crossover Operator

A permutation is selected with the equiprobability $\frac{1}{n!}$ without considering the parent permutations, as for adopted children who have not inherited any genes from their adoptive parents. It is similar to the Monte Carlo generation very often used in scheduling area. It is a reference for the performance indicators: a good operator designed for particular problems linked to a given performance measure must have considerably better results than RD crossover operators relatively to this performance measure.

2.3 PREVIOUS RESULTS

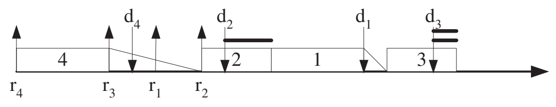
In a previous paper, (Djerid, Portmann and Villon, 1996) compared some permutation crossover operators by using indicator performances. They also applied the cross-over operators on a big set of generated permutation parents. They assumed then that each indicator allows them to measure how many characteristics of the parents (edges, arcs, relative orders or positions) are kept in the generated children. The obtained results can be summarized as follows. According to their suppositions, ERX was the best cross-over operator for keeping edges, ARX followed it immediately, and 1X, OX or even LOX (given by decreasing order of performances) were not too bad for indicator SEBI. ARX was the best operator for keeping arcs, and again 1X, OX and immediately after LOX were not so bad for indicator EBI, while ERX was very bad. 1X and kX were the best operators for keeping relative orders between each couple of operations (indicator PCBI), LOX is also correct, but OX, ERX and ARX are very bad. 1X and kX are good for keeping the positions (indicator PBI), LOX is also good, OX, ERX and ARX are very bad. For every indicator the cross-over random is particularly bad. In the previous work, only permutations, cross-over operators and indicators were considered. It was assumed that good parents have good characteristics: edges of small values, arcs of small values, operation orders, in which jobs with small due dates and great tardiness penalties stand before jobs with greater due dates or smaller penalties, etc. It was also assumed that keeping the corresponding characteristics of the good parents must give good children. Nevertheless, these assumptions are not always held when particular instances of some specific problems are considered.

3 NEW DEVELOPMENTS AND RESULTS

3.1 PERMUTATION SCHEDULING PROBLEM

We consider a one-machine scheduling problem in which we have different families of products to sequence. For each product, we have ready dates (r_i) and due date (d_i). There exist setup times depending on the sequence between 2 products belonging to 2 different families. So between 2 products of the same family, no setup time is considered. The criterion to minimize is the sum of weighted tardiness. According to (Blazewicz, Ecker, Schmidt and Weglarz 1994), the problem is noted $1 \mid r_i, d_i, S_{sd} \mid \sum w_i T_i$. A Gantt chart is given in Figure 1 to illustrate an example of the kind of sequence we can obtain.

Figure 1: Gantt chart



If we analyze the problem, it can be said that a solution should be good when setup times are not too important and/or the products with a small due date and/or with a big penalty are sequenced very early. For these reasons, if we want the efficiency of the genetic algorithm not to be due to random, some adapted operators must be designed for this kind of problem. So, it is necessary to find a good trade-off between keeping arcs (corresponding to small setup times or no setup times) and keeping partial order (representing the more urgent product and especially when the penalty of the product is high).

3.2 DESCRIPTION OF THE TOOL

We have implemented a tool able to establish some simple statistics. First of all, we have to give different parameters used by a generator of instances. Each set of parameters are: number of products, number of product types, average of processing time, standard deviation of processing times, bounds for the ready dates, for the slacks, for the tardiness penalties and for the called "rough" set-up times. In fact, setup times are corrected in order to give them more or less importance relatively to average processing times. After generating a set of one-machine scheduling problems, it is possible to generate different permutations and to compute the value of the problem criterion for each permutation. Then according to these values, we affect a mark from 1 to 5 to each individual. These individuals represent parents and are going to be crossed according to different cross-over operators. Then offspring can also be evaluated and then we assign again a mark to each offspring from 0 to 6: 0 (resp. 6) when the offspring is strictly better (resp. worse) than the best of the parents. The marks from 1 to 5 are computed with the same bounds on the criterion as for the parents marks. Now, it is possible to see, for a given cross-over operator, the percentage of times when good offspring are obtained from good parents or to see, for a given cross-over the percentage of time a good or a bad offspring is obtained from bad parents.

3.3 SET OF EXPERIMENTATION

The set of our experiments is defined as follows. We consider 20 or 100 products to schedule. We consider a population of 500 solutions (i.e. permutations) and we make 500 crossing over. Then we obtain 1000 children.

For set A, we have twenty products, ten different families of products and the ready dates are equal to 0 for every product. The average of processing time is equal to 50 and the standard deviation is equal to 25. The average value of slacks $(d_i - p_i - r_i)$ is equal to 250 with a deviation of 250. The tardiness penalties are randomly chosen between one and ten. There are ten different families.

For set B, we have hundred products and fifteen different families of products. The others parameters are identical to set A.

For set C, we have twenty products, release dates varying from 0 to 500. The others parameters are identical to set A.

For set D, we have hundred products, release dates varying from 0 to 1500, the penalty is equal to 1 and there are ten different families of products.

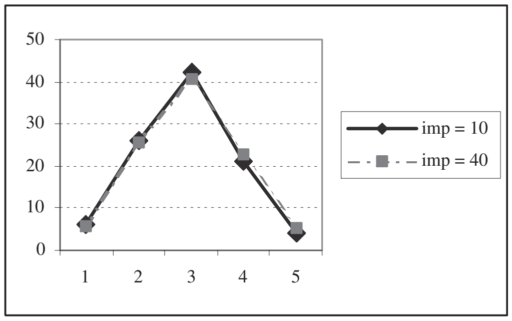
The others parameters are identical to set A. The minimum value of setup times is ten and the maximum is sixty. We generate matrices whose dimensions are 15x15 so that we can extract easily a matrix 10x10 when the number of different families is ten. Then we give more or less importance to setup times with a parameter Imp that is either equal to ten or fourty. When Imp is equal to fourty, it means that the average of setup times is four times greater than the average of processing time. When Imp is equal to ten, the average of setup times and of processing times are almost identical. Then for each set, we obtain two subsets of results according to the value of Imp. As cross-over operators, we choose Random LOX, 1X, kX, ERX and ARXM.

Different statistics are then done. To identify the type of statistics, we use a string composed of six parameters abcdef. "ab" represents the interval of mark for the first parent, "cd" the interval of marks for the second parent and "ef" the interval of marks for the offspring. Then if we want to see the percentage of time two very good parents (i.e. with mark equal to 1) give a very good offspring (i.e a mark equal to 0 or 1) then the string we use is 111101.

3.4 NEW RESULTS

Here, we try to validate the previous results on a particular scheduling problem. The problem was chosen because considering both setup times which depend on the sequence and the weighted tardiness penalties. It seems very interesting to keep arcs, relative orders and even positions. The role of the parameter Imp is to give more or less importance to the setup times and consequently to the arc conservation. For each set of experiments (sets A, B, C and D) and for Imp = 40 or 10, we first analyze the structure of the populations. The average structure of the initial population corresponding to the instances of set A (Imp = 40 and Imp = 10) is given in the figure 2.

Figure 2: Structure of the initial population



In fact, both initial and created populations corresponding to any set of instances and any value of Imp have always the same quality repartition. Very few individuals have got marks equal to 0, 1, 5 or 6, between 20% and 25% of the population have marks equal to 2 and the same for 4 and between 40% and 45% of the population have marks equal to 3. We obtain the same structure for each generated population because we cross the whole set of parent couples without considering their marks. Since crossover random creates always random subset of the whole set of solutions ($n!$ permutations), it is probable that the whole set of solutions has the same structure. It means that, for the generated instances of this particular scheduling problem, there are few good solutions, few bad solutions and a lot of solutions with medium values for the criterion. In order to compare the crossover operators, we distinguish the set of children depending on the marks of their parents. This approach provides us with a lot of children subsets. Here, we propose only a selection of values, they are presented in table 6 for the instance set D and Imp equal to 10.

Table 6: Quality of the children depending on the quality of the parents

Family D		**Imp = 40**		
	Children	0 2	3 4	5 6
parents 1 2	LOX	71,18	28,82	0
	1X	76,39	23,61	0
	kX	59,03	40,62	0,35
	ERX	60,07	38,89	1,04
	ARXM	60,07	38,89	1,04
	random	28,47	67,36	4,17
parents 3 4	LOX	16,99	79,51	3,5
	1X	14,91	81,89	3,2
	kX	21,37	74,56	4,07
	ERX	26,16	69,46	4,38
	ARXM	20,26	73,68	6,06
	random	31,66	63,88	4,46
parents 4 5	LOX	4	77,5	18,5
	1X	5,5	76	18,5
	kX	5,5	83	11,5
	ERX	16	71	13
	ARXM	9	78	13
	random	31	65,5	3,5

The six first lines consider only the subset of children whose both parents have got marks equal to 1 or 2. Each line corresponds to a given cross-over operator. Each column gives the percentage of children of this subset who have got marks equal to 0, 1 or 2 for the first column, 3 or 4 for the second column, 5 or 6 for the last column. It may be seen, that, even if the whole population is probably poor of good (mark 2) or very good (mark 1) solutions, when parents have got marks 1 or 2, then, for the best operator, 76.39% of the created children are good or very good. The other cross-over operators are not bad with a performance greater than 60%, except random which gives 28.47% (i.e. approximately the percentage of good and very good solutions in the whole solution set).

The six following lines in table 6 consider only the subset of children whose both parents have got marks equal to 3 or 4. The other explanations for lines or columns are identical. Parents of medium or bad quality are crossed. The orders are probably bad and the cross-over operators keeping the orders are then the worst considering the percentage of good, very good or excellent children generated. ERX and ARX are very slightly better. In fact, when we cross parents which are neither very good nor good, then random is the best cross-over in order to obtain children of good quality. When crossing parents of medium or bad quality, most of the children are of medium or bad quality more than 64% for any cross-over operator including random (i.e. probably more than the percentage of medium or bad solutions in the whole solution set).

The six last lines only consider the subset of children whose both parents have got marks equal to 4 or 5. In this case, only random is able to give good or very good children and again, the majority of the created children are medium or bad (because it is the major part of the whole solution set and so the probability is greater for obtaining children having these marks).

Table 7: Percentage of children getting the same quality as both parents

Imp=10	**1 2⇒ 0 1 2**	**3 4⇒ 3 4**	**4 5⇒5 6**
LOX	73,9	82,3	17,7
1X	78,9	85,5	18,2
kX	73,2	83,1	14,5
ERX	43,1	69,9	6,2
ARXM	49,9	72,7	11,9
random	32,1	67,8	5,2

Table 8: Percentage of children getting the same quality as both parents

Imp=40	**1 2 ⇒ 0 1 2**	**3 4 ⇒ 3 4**	**4 5 ⇒ 5 6**
LOX	71,5	81,8	15,8
1X	76,7	85,2	19,3
kX	70,7	81	15,6
ERX	51,5	73,2	9,5
ARXM	55,5	75,4	13,8
random	31,6	68,6	4,7

Table 6 is only concerned by one set of instances (set D and Imp=40). In order to try to compare the cross-over operators on all the generated instances, we present average results in table 7 (Imp=10) and in table 8 (Imp=40). To obtain these tables, we consider the contents of table 6. We add elements in correspondence of the similar tables for the chosen Imp value and for set A, set B, set C and set D and we divide the obtained results by 4. This provides us with average values. Then we keep only the diagonal blocs of the tables (i.e. for a given quality of parents, the percentage of children who have got similar quality). x y $\Rightarrow$ z v means: among the children generated by parents of quality x or y (for both mates), what is the percentage of children who have got marks z or v. In this table, we may try to compare the quality of the crossover for the given scheduling problem. It may be noted that 1X is always the best for giving the greatest percentage of children who have got quality similar to quality of both parents. LOX and kX gives always the following values of the percentages. They are followed by ARX and ERX with ARX always better than ERX and random always the worst, while it gives always a percentage similar to the percentage of children of this quality in the whole solution set.

3.5 PERSPECTIVES

A difficulty in our experiments was the particular structure of the whole solution set, which increases the probability of obtaining solutions of medium quality. It might be seen that if using other permutation scheduling problems (such as, for example, permutation flow-shop problems) another structure of the population could be obtained (it is probably true for the makespan criterion for which many optimal solutions generally exist). Many further experiments might be done on the problem considered here. In particular, we might increase the importance of the setup times by increasing Imp and decrease the importance of the order by taking, for example, a common due date. Then we might see when ARXM is becoming much better than the order cross-over operators (LOX, OX, 1X, kX). Other permutation crossover operators may be integrated in our tool and tested. We may also verify on a complete GA that the final results are increased by using good crossover operators selected by our statistics.

4 CONCLUSIONS

We tried here to verify the following assumption: when a cross-over operator keeps good characteristics of the chromosome 'parents' and when we select the best chromosomes with a greater probability, then we must increase the number of good generated children. For permutation problems, it seems that the aptitude to keep characteristics may be measured by performance indicators, because the results with the chosen particular scheduling problem are in correspondence with the results obtained previously with performance indicators.

References

J. Blazewicz, K. Ecker, G. Schmidt, J. Weglarz (1994). Scheduling in Computer and Manufacturing Systems. *Springer-Verlag edition.*

C. Caux, H. Pierreval, M-C. Portmann (1995). Les algorithmes gntiques et leur application aux problmes d'ordonnancement. *Rairo-APII (Automatique Productique Informatique Industrielle)*, 29, 4-5, 409-443.

L. Davis (1985), Job-Shop Scheduling With Genetic Algorithms. *Proc. 1st Int. Conf. on Genetic Algorithms and Their Applications*, 136-140, Lawrence Erlbaum, Hillsdale.

L. Djerid, M-C. Portmann (1999), How to Keep Good Schemata Using Cross-over Operators for Permutation Problems, *international conference, IFORS'99*, Pkin, August 16-20, submitted to International Transactions in Operational Research.

L. Djerid (1997), Hybridation d'algorithmes gntiques et de mthodes classiques de recherche oprationnelle pour rsoudre des problmes d'ordonnancement, Nancy / Institut National Polytechnique de Lorraine, *PhD thesis.*

L. Djerid, M-C. Portmann, P. Villon (1996), Performance analysis of permutation cross-over genetic operators, *Journal of Decision Systems*,**4**(1/2), 157-177.

H. Emmons (1969), One-Machine Sequencing to Minimize Certain Functions of Job Tardiness. *Operations Research*, **17**(4), 701-715.

E. Falkenauer, S. Bouffouix (1991), A Genetic Algorithm for Job Shop. *Proceedings of The IEEE International Conference on Robotics and Automation*, 1, 824-829, Sacramento.

D. E. Goldberg (1989), Genetic Algorithms in Search, Optimization, and Machine Learning, *Addison-Wesley Publishing Company, Inc.*

J.H. Holland (1975), Adaptation in Natural and Artificial Systems, *Mit Press*, Cambridge, Mass.

I. Oliver, D. Smith, J. Holland (1987), A Study of Permutation Crossover Operators on The Traveling Salesman Problem. *Proc. 2nd Int. Conf. on Genetic Algorithms and Theirs Applications*, 224-230.

Y. Whitley, T. Starkweather, D. Fuquay (1989), Scheduling Problems and Traveling Salesman: The Genetic Edge Recombination Operators. *Proc. Third Int. Conf. on Genetic Algorithms and Their Applications*, Ed. Morgan Kaufmann, San Mateo, Calif, 133-140.