Genetic Forma Recombination in Permutation Flowshop Problems

Carlos Cotta, José M. Troya
Departamento de Lenguajes y Ciencias de la Computación
E.T.S.I. Informática, Universidad de Málaga
Complejo Politécnico (2.2.A.6), Campus de Teatinos
29071-Málaga (Spain)

{ccottap, troya}@lcc.uma.es

Abstract

This paper analyzes different representations for permutation flowshop problems. This is done using forma analysis to assess the quality of these representations with respect to makespan optimization. Classical recombination operators are studied and empirically evaluated in this context. It is shown that the best operators work on representations in which absolute positions of tasks are relevant. Subsequently, some new operators operating on these representations are proposed. These new operators are designed to exhibit specific properties regarding implicit mutation and forma transmission. Their performance is shown to be competitive with traditional operators.

Keywords

Forma analysis, fitness variance of formae, representation, recombination operators, permutation flowshop.

1. Introduction

The scheduling of production processes is a task to which great efforts are devoted due to its economical importance. Unfortunately, it has been shown that finding the optimal scheduling for a general production process is an NP-hard problem (Garey and Johnson, 1979). This implies that traditional algorithmic techniques such as Dynamic Programming (Bellman and Dreyfus, 1962) or Branch and Bound (Lawler and Wood, 1966) are not adequate because of their lack of scalability. Therefore, the interest of many researchers has been directed to the design of heuristics providing good suboptimal solutions for these problems. In this sense, modern heuristic techniques (Reeves, 1993) constitute a valuable alternative.

Genetic algorithms (Holland, 1975) are one of the most representative members of these modern heuristic techniques. They maintain a pool of tentative solutions for the problem under consideration, and use the principles of natural evolution, namely adaptation and survival of the fittest, to guide the generation of new promising solutions. These solutions are constructed using some reproductive operators, traditionally a recombination and a mutation operator. The former is intended to combine the positive features of (usually) two solutions to create a new solution, and it has been traditionally given a central rôle in the functioning of the algorithm. As to the latter, its mission is to preserve the diversity in the solution pool.

Although genetic algorithms have been successfully applied to a wide variety of problems, it has been proved that they are no-better than any other search algorithm (including random search) if no knowledge on the problem under consideration is included in them. This fact was initially stated by Hart and Belew (1991) and later by Wolpert and Macready (1995) in the so-called “No Free Lunch Theorem”. Essentially, this implies that the elements of the algorithm have to be carefully selected to match the characteristics of the problem being solved. To be precise, choosing appropriate representation and operators is crucial. For this purpose, Forma-

Analysis (Radcliffe, 1991) provides some tools to guide this selection process. These tools have been used in the present paper to analyze the functioning and performance of traditional genetic recombination operators, as well as to define some new operators with desired properties.

The remainder of the paper is organized as follows. Section 2 briefly reviews Forma Analysis. This tool is used to discuss different representations of the addressed problem (optimization of a permutation flowshop) in section 3. These representations are empirically evaluated with respect to a heuristic measure: intra-forma variance of fitness. Then, traditional operators are studied in terms of these representations in section 4. The principles of Forma Analysis along with the empirical results obtained in section 3 allow introducing new operators in section 5. Finally, section 6 presents concluding remarks and outlines future work.

2. Background on Forma Analysis

This section is aimed at providing the theoretical background upon which the analysis presented in this work is grounded. First, the traditional view of genetic algorithms as schema manipulators is reviewed. Then, it is reformulated in terms of more general entities (formae). Next, some properties of formae and the way they are manipulated are discussed. Finally, an operator-based view of representations is presented.

2.1. Genetic Algorithms and Schemata

Schema analysis has been the theoretical tool for studying the behavior of genetic algorithms for a long time. This analysis is based on the concept of *schema*, which can be seen as a partially specified solution. A more rigorous formulation requires the definition of a coding function r mapping solutions from a solution space S to *chromosomes* in a chromosome space C. Chromosomes usually consist of a list of genes ($G_1, \ldots, G_n$), each of which is taken from a set of alleles A_i, i.e., $C \equiv A_1 \times A_2 \times \cdots \times A_n$.

This coding is denoted as *genetic representation* (Radcliffe and Surry, 1994) and constitutes the basis for defining equivalence relation among solutions (or, strictly speaking, among representations). To be precise, two chromosomes are considered equivalent under one of these equivalence relations if they share the same alleles in certain genes. Therefore, each of these equivalence relations can be specified as a string $j \in \{\square, \blacksquare\}^n$, where $\square$ represents a wildcard and $\blacksquare$ a gene that must match. For example, assume binary genes, i.e., $A_i = \{0,1\}$, i=1..n. The chromosomes h_1= 0011 and h_2=1010 are equivalent under $\square\blacksquare\blacksquare\square$, but not under $\blacksquare\blacksquare\square\square$.

It is usual to consider the equivalence classes induced by these equivalence relations instead of the relations themselves. In the previous example, the equivalence relation $\square\blacksquare\blacksquare\square$ induces four equivalence classes, namely $\square$00$\square$, $\square$01$\square$, $\square$10$\square$ and $\square$11$\square$. Each of these equivalence classes is a *schema*. It can be easily seen that each chromosome belongs to ("is an instance of" in the standard terminology) 2^n schemata, i.e., one schema for each of the 2^n equivalence relations that can be defined. In this scenario, it is desirable that chromosomes within the same equivalence class have similar phenotypical properties, which should be reflected in a correlation of their fitness values. Under this assumption, evaluating a chromosome provides information about all schemata it belongs to. This phenomenon is known as implicit parallelism and has been one of the most powerful explanations of the functioning of genetic algorithms to date.

Schema analysis provides a view of genetic algorithms in which the population can be considered a pool of schemata whose distributions change by means of the application of genetic operators. The behavior of these operators is analyzed in terms of the *gains* and *losses*

in the distribution of manipulated schemata. The final result is the well-known Schema Theorem (Holland, 1975):

$$n_\xi(t+1) \geq n_\xi(t) \cdot \sigma_\xi(t) \cdot \left[1 - \sum_{\omega \in \Omega} p_\omega \cdot p_\omega^\xi\right] \quad (1)$$

In the above expression, $n_\xi(t)$ represents the number of instances of schema ξ in the population at time t, $s_\xi(t)$ is the probability of selecting an instance of schema ξ at time t, p_ω is the probability of applying operator ω, and p_ω^ξ measures the disruption rate caused by the application of ω to an instance of ξ. Equation (1) combines the effects of the different operators used in the algorithm to provide an estimation of the schema distribution in the next generation.

2.2. Generalizing Schemata: Formae

Although schemata have been a valuable (in fact, fundamental) tool for providing insights into the internal functioning of a genetic algorithm, schema analysis is limited for several reasons. The most important reason (or at least the most relevant for the purposes of this paper) is the impossibility of encapsulating within a schema arbitrary phenotypical properties. Consider that the total number of schemata is $(\gamma+1)^n$, assuming the use of a γ-ary alphabet. However, the number of arbitrary subsets of solutions is $P(S) = 2^{\gamma^n}$, vastly more than the number of schemata[1]. Thus, only a very small fraction of these subsets can be expressed as a non-trivial schema, i.e., a schema containing at least one ■ symbol. This can be exemplified as follows: consider a base-γ representation of integers; it is impossible to define a schema other than $\square\square\cdots^n\cdots\square$ whose membership be shared by all multiples of κ, given that κ is not multiple of γ and a large enough (i.e., κ^2) interval of representation (Fig. 1).

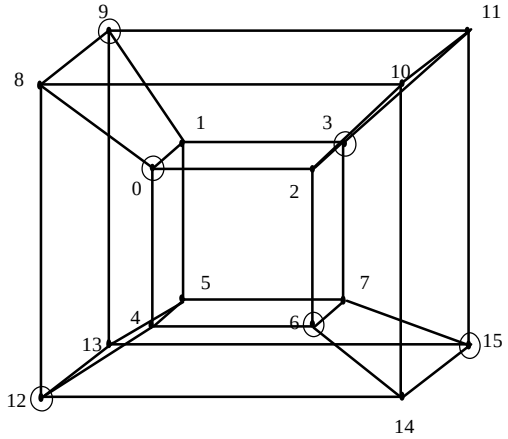


Figure 1. This hypercube contains all schemata (vertices -order 4-, edges -order 3-, facets -order 2-, cubes -order 1-, hypercube -order 0-) than can be defined on a binary chromosome with n=4 bits. Only the whole hypercube contains all multiples of 3 (rounded vertices).

In this situation, a more abstract representation of solutions is required. To be precise, solutions could be represented by a list of their relevant properties. Following the above example, 15 could be represented as $\{\dot{1},\dot{3},\dot{5},\dot{15}\}$, i.e., the list of its divisors. This feature-based representation is denoted as *allelic representation* (Radcliffe and Surry, 1994), and can

[1]Antonisse (1989) has argued that the $\square$ symbol should be interpreted as a family of symbols $\square_\Xi$, where Ξ is an arbitrary subset of alleles for the corresponding gene. Under this assumption, the degree of implicit parallelism is higher since the number of schemata raises up to $(2^\gamma\text{-}1)^n$, but it is still superexponentially small with respect to 2^{γ^n}.

be used to define generalized equivalence relations. These equivalence relations induce equivalence classes grouping solutions with desired features. Each of these classes is called a *forma*. For example, consider the following equivalence-relation template:

$$\xi_\kappa(\eta,\zeta) = \begin{cases} 1 & \textit{if } (\overset{\bullet}{\kappa} \in \eta) \Leftrightarrow (\overset{\bullet}{\kappa} \in \zeta) \\ 0 & \textit{otherwise} \end{cases} \quad (2)$$

Each instance ξ_κ of this template defines two formae ξ_κ^0 and ξ_κ^1, the latter containing those integers that are multiples of κ and the former those integers that are not. Thus, it can be used to group the vertices as required in Fig. 1. An obvious requirement is that these formae be effectively processed by the algorithm to make them relevant in its functioning.

In this sense, Radcliffe (1991) and Vose (1991) have shown that the schema theorem is still valid if ξ is an arbitrary forma (or *predicate* according to Vose´s terminology) instead of a schema, given that the disruption coefficients p_ω^ξ are adequately calculated. The operators are therefore required to effectively manipulate these formae. Otherwise, the disruption rates would be so high that these formae would become irrelevant. Some considerations on how to manipulate formae are discussed in subsection 2.4. Previously, some basic concepts on forma-based representations are presented.

2.3. Basic Concepts on Forma-Based Representations

Formae, as defined before, are equivalence classes induced by certain equivalence relations. It is then appropriate to consider some notions in analogy with linear algebra. First, two formae are said to be *compatible* if their intersection is non-empty, i.e., if there exists at least one solution that belongs to both formae. In the previous example, formae ξ_3^1 and ξ_5^1 are compatible but ξ_3^0 and ξ_9^1 are not.

It can be easily seen that each solution can be specified by the list of compatible formae it belongs to. It is clearly desirable that the set of equivalence relations inducing these formae can be used to distinguish between any pair of different solutions (i.e., two different solutions are not equivalent under at least one of the members of the set). In this situation, this set is said to *cover* the set of solutions.

Finding a set of equivalence relations covering the solution space S is important since it allows a homogeneous treatment of all solutions. This set of equivalence relations is *independent* if, and only if, none of its members can be generated as the intersection of other members. An equivalence relation for which this does not happen is called *redundant*.

Now, a set of equivalence relations is said to be a *basis* for another set if, and only if, each member of the latter can be constructed as the intersection of some members of the former. Finally, a set of equivalence relations is *orthogonal* if, and only if, given any tuple of formae, each of these formae generated by a different member of the set, their intersection is non-empty, i.e., any combination of formae induced by different equivalence relations is valid. For example, define ξ'_κ as the intersection of every ξ_λ such that λ is a power of κ. Then, the set $\Psi=\{\xi'_\kappa \mid \kappa \text{ is prime}\}$ is orthogonal. Traditional schemata are usually orthogonal as well. An example of non-orthogonal formae is shown in next subsection.

2.4. Characterizing Forma Manipulation

There exist some properties that can be studied when analyzing the behavior of an operator with respect to the formae it manipulates. These can be summarized in *respect*, *transmission* and *assortment* (Radcliffe and Surry, 1994).

Respect means that every child generated by an operator is a member of every forma to which both parents belong, i.e., the child will exhibit any feature present in both parents. This

property can be seen as the exploitative side of recombination: in the early stages of search, solutions are very different and thus they do not share membership to many formae but, as the search advances, promising formae increase their number of instances according to Equation (1). A respectful recombination ensures that these formae are transmitted from parents to children.

Transmission is a related property. A recombination operator is said to be transmitting if, and only if, at least one parent belongs to each forma of which the child is a member. This property tries to capture the classical rôle of recombination in which the features of the parents are combined but no new feature is introduced. If this is not the case, the operator is said to introduce *implicit mutation.* Notice that transmission does not imply respect and vice versa, as shown in Fig. 2.

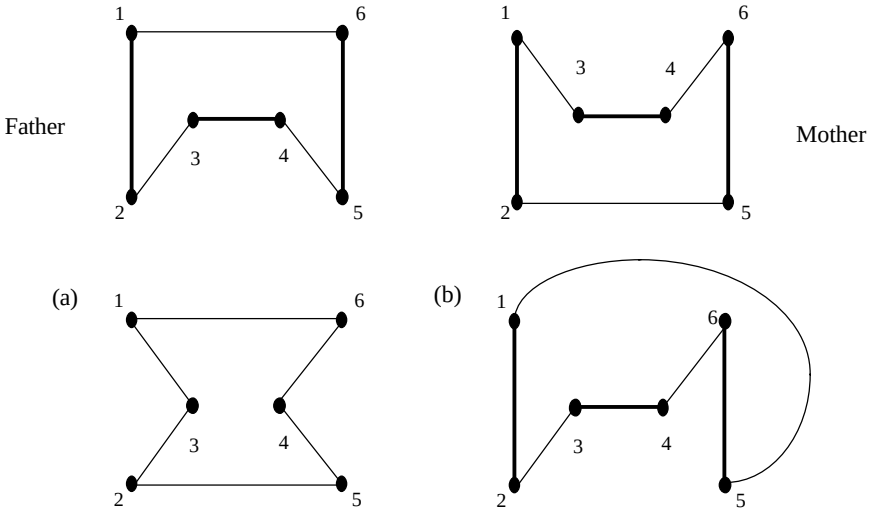


Figure 2. Two solutions for the symmetric traveling salesperson problem are recombined. A strictly-transmitting non-respectful recombination is shown in example (a). The undirected edges 12^u, 34^u and 56^u are common to both parents, but they are not present in the child. However, each edge in the latter is taken from a parent. Example (b) shows a non-transmitting respectful recombination. The child contains every common edge but 15^u is not present in any parent

Finally, assortment represents the exploratory side of recombination. An operator is said to be *properly* assorting if, and only if, it can recombine any two instances of compatible formae producing a child in their intersection. It the operator requires to recombine the children with the parents or among themselves several times to achieve this effect, it is said to be *weakly* assorting.

The properties of assortment and respect are not always compatible. This can be seen in Fig. 2. The undirected edges 45^u and 46^u are compatible, but combining them excludes the common edge 34^u. In such a situation, the representation is said to be *non-separable*. Orthogonal representations (e.g., traditional schemata) are separable, but the reverse is not always true.

2.5. An Operator-Based View of Representations

It is very common to identify the genetic representation with the internal encoding of solutions. Under this assumption, it makes sense to distinguish between the *genotype* (the internal encoding) and the *phenotype* (the solution itself). However, this is no longer true for the allelic representation. By specifying and processing the relevant properties of a solution, the algorithm actually manipulates phenotypes (Radcliffe, 1992). The choice of internal representation has no qualitative influence on the algorithm.

This consideration is important from the point of view of operators. Traditionally, a representation (i.e., encoding) of solutions was chosen and the operators were designed to work on this representation (in fact, manipulating schemata). Forma analysis provides an alternative scenario in which the relevant properties of solutions are identified and the operators are subsequently designed to process these properties, whether they can be represented as a linear string of genes or not. The internal representation is to some extent secondary.

It is thus the choice of operators what determines the representation by means of the formae they manipulate. This implies that a change of operators is equivalent to a change of representation. In fact, several representations may coexist in the algorithm if different operators are used. This duality is not complete though. As shown by Radcliffe (1994), fixing the operators and changing the genetic representation is less flexible.

This operator-based view of representations has a direct implication on the analysis of the algorithm. When studying a certain operator, the formae it manipulates must be identified to determine on which representation it is working. Then, that representation can be analyzed to assess its quality, using the results to predict the performance of the algorithm. This has been the approach used in the present paper. More precisely, several idealized representations of the problem under consideration have been studied. Next, different operators have been analyzed to determine which the dominant representation is in each case. Finally, the process has been reversed and new operators have been designed to work on the most promising representations.

3. Representation of Permutation Flowshops

After having presented the essentials of forma analysis, this section examines different representations for the addressed problem: the optimization of a permutation flowshop. The characteristics of this problem are previously described in Section 3.1.

3.1. Description of the Problem

Production scheduling comprises a family of different problems (see Hillier and Lieberman, 1967). In this paper, we consider the so-called *n*-job *m*-machine permutation flowshop problem. This problem involves a set of machines $M_1, \ldots, M_m$ and a set of tasks $T_1, \ldots, T_n$. All tasks must be processed by all machines in the same predefined order and subject to the following constraints:

1. Each task T_i requires exclusive use of machine M_j during t_{ij} time units.
2. Each task flows from one machine to the next one without any delay, and waits in an unbounded buffer while the machine is busy.
3. All tasks are processed in any machine in the same order.

The goal is to schedule the tasks optimizing a certain objective criterion. In this work, the objective is to minimize the total completion time of the system, also known as *makespan* (C_{max}). This problem is usually labeled as $n/m/P$,no-wait/C_{max}, where *P* stands for permutation. In effect, solutions are constrained by the constraint #3 to be permutations of the tasks.

3.2. Forma-based Representation of Permutations

Since the solutions of the problem under consideration are expressed as permutations, this subsection is devoted to discuss several representations for that purpose. Each of these representations is intended to capture different properties of solutions such as relative ordering or absolute positions.

3.2.1. Precedence Formae

As stated in (Fox and McMahon, 1991), a permutation of the elements of a certain set Z induces a total ordering in it. This total order relation can be represented by means of a Boolean precedence matrix. The matrix element e_{ij} placed in row i, column j is TRUE if, and only if, the symbol labeled as i occurs in the sequence before the element labeled as j (Fig. 3).

	1	2	3	4	5
1	F	T	T	T	T
2	F	F	F	F	T
3	F	T	F	T	T
4	F	T	F	F	T
5	F	F	F	F	F

Figure 3. Precedence matrix of the permutation 13425 where T=true and F=false.

This matrix contains all the information that can be extracted from a given permutation, and constitutes the basis for the first considered representation: the *precedence representation*. Using this representation, a permutation is manipulated as an unordered list of precedence relations (the TRUE entries of the precedence matrix). For example, permutation 13425 is represented as {⟨1,2⟩, ⟨1,3⟩, ⟨1,4⟩, ⟨1,5⟩, ⟨2,5⟩, ⟨3,2⟩, ⟨3,4⟩, ⟨3,5⟩, ⟨4,2⟩, ⟨4,5⟩}. Now, it is possible to consider the following family of equivalence relations:

$$\xi_{ab}(\zeta,\nu) = \begin{cases} 1 & if\ \ (\langle a,b\rangle \in \zeta) \Leftrightarrow (\langle a,b\rangle \in \nu) \\ 0 & otherwise \end{cases} \tag{3}$$

Thus, two solutions are equivalent under ξ_{ab} if, and only if, the precedence relation between elements a and b is the same in both solutions. Clearly, each ξ_{ab} induces two equivalence classes in the space of permutations. These classes will be denoted by ξ^0_{ab} and ξ^1_{ab}, the latter containing all permutations in which a occurs before b and the former those ones in which the reverse is true. Each of these classes is a *precedence forma,* and will be called a *negative* and a *positive* forma respectively. Although handling negative formae is generally complicated, this is not the case since it is trivial that $\xi^0_{ab} \equiv \xi^1_{ba}$.

It can be seen that $E=\{\xi_{ab} \mid a < b : a, b \in Z\}$ covers the space of permutations. Notice that not all subsets of formae induced by E can represent a valid permutation, e.g., a subset containing ξ^1_{ab}, $\xi^1_{bc}$and ξ^0_{ac}. In other words, E is non-orthogonal. Furthermore, E is even non-separable. As an example, consider the subsequences 1234 and 4231. It is perfectly valid to combine ξ^1_{12} (from the first parent) with ξ^0_{13} (from the second parent). However, such a combination implies ξ^0_{23} which is incompatible with ξ^1_{23}, a common forma. These are important facts to be considered when designing operators to handle precedence formae.

3.2.2. Adjacency Formae

Precedence formae are intended to carry all micro-topological properties of the permutation and, in certain situations, may be an excessively fine grain for the purposes of the optimizer. In such case, it is possible to consider a more coarse representation based on adjacency. With such a representation, a permutation can be represented as a vector of exactly $|Z|$ pairs, each one indicating which element is the immediate predecessor of each one. For example, permutation 13425 is represented as {⟨0,1⟩, ⟨1,3⟩, ⟨2,5⟩, ⟨3,4⟩, ⟨4,2⟩}. A dummy element (0) has been included to be used as the first element of the sequence. This representation is also

known as *directed-edge* representation and is commonly used for problems such as the asymmetric traveling salesperson problem.

As for precedence formae, a family of equivalence relations can be defined:

$$\eta_{ab}(\zeta,\nu)=\begin{cases}1 & if\ \left(\langle a,b\rangle\in\zeta\right)\Leftrightarrow\left(\langle a,b\rangle\in\nu\right)\\ 0 & otherwise\end{cases} \tag{4}$$

Thus, two solutions are equivalent under η_{ab} if, and only if, they both have (or do not have) the directed edge $\langle a,b\rangle$. The equivalence classes induced by η_{ab} are denoted by η^0_{ab} and η^1_{ab}. Unlike precedence formae, negative adjacency formae are difficult to manipulate. In fact, simply determining whether the intersection of a set of negative formae is empty or not is NP-Hard. This follows from the fact that a set of negative adjacency formae defines an incomplete graph, and determining whether their intersection is non-empty is equivalent to find a Hamiltonian Path, a well-known NP-hard problem. Furthermore, a respectful operator considering negative formae should exclude all edges not considered in any parent, i.e., it should not introduce any new edge and hence would be transmitting positive formae. However, adjacency formae are not only non-orthogonal (e.g., η^1_{ab} and η^1_{cb} are incompatible) but also non-separable (e.g., η^1_{ab}, η^1_{bc} and η^1_{ca} are pairwise compatible but their intersection is empty). Therefore, simultaneously respecting and transmitting positive formae reduces in most situations to return one of the parents. For this reason, the operator for adjacency manipulation discussed in this work only considers positive formae.

3.2.3. Position and Block Formae

The representation of permutations based on position formae is the most natural way of representing them: the *path* representation (Michalewicz, 1992), i.e., an ordered list of elements of *Z*. To be precise, let the equivalence relation φ_i be defined as

$$\varphi_i(\zeta,\nu)=\begin{cases}1 & if\ \zeta_i=\nu_i\\ 0 & otherwise\end{cases} \tag{5}$$

Each φ_i defines $|Z|$ position formae (a forma for each symbol that may appear at a given position). Position formae are equivalent to *o*-schemata (Goldberg and Lingle, 1985). To be precise, position forma $\varphi_{i,a}$ is equivalent to the following *o*-schema: $\square\cdots^{i-1}\cdots\square a\square\cdots^{|Z|-i}\cdots\square$, i.e., it contains all permutations in which the element labeled as *a* occurs in the *i*th position.

A higher-level view of permutations based in position formae is possible: the *block* representation. A block is a set of contiguous elements in a permutation, which can then be represented as a set of blocks. Block formae can be naturally expressed as the intersection of adjacent position formae:

$$\beta_i^{e1..en}\equiv\bigcap_{j=1}^{n}\varphi_{i+j-1}^{ej} \tag{6}$$

Block formae constitute a mechanism to link into a macro-forma several basic position formae. As precedence and adjacency formae, both position and block formae are non-orthogonal. However, they are separable, i.e., they can be simultaneously respected and assorted. This fact will be examined later.

3.3. Fitness Variance of Formae

Radcliffe and Surry (1994) suggest that the fitness variance of formae can be used to assess to which extent a representation carries useful fitness information. The rationale for this is the fact that a high fitness variance introduces noise in the sampling of formae the algorithm carries out, thus making it wander through the solution space. Consider that the term $s_\xi(t)$ in

Equation (1) depends on the observed fitness of forma ξ at time t, $\hat{u}_\xi(t)$, measured as the mean fitness of all instances of ξ in the population. Ideally, this quantity should be identical to the real fitness $u_\xi(t)$. Since this is not usually the case in practice, the smaller the fitness variance, the smaller the deviation of the observed fitness can be in an arbitrary population. A collection of experiments has been done to test this hypothesis.

The experiments have been done using four problem instances taken from the OR-library by Beasley (Beasley, 1990) of sizes ranging from 30×10 up to 75×20, and three randomly generated instances of sizes between 100×20 and 200×50. The representations considered are: adjacency, precedence, position and block. For each representation and forma size, 250 random formae have been generated and 500 solutions containing each forma have been tested. The results for two of these instances are shown in Fig 4. The results for the remaining instances look identical.

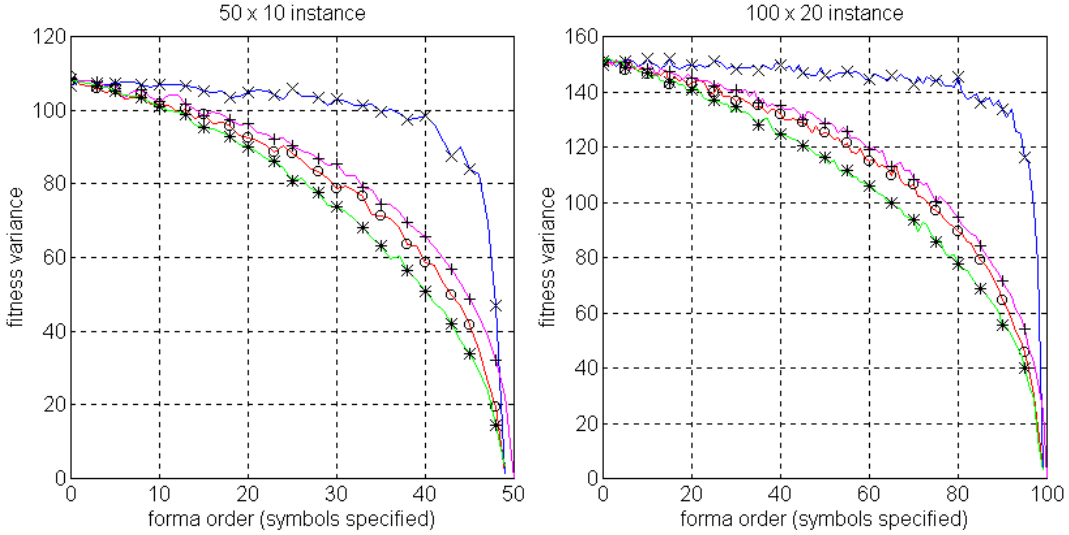


Figure 4. Fitness variance of formae for the four representations considered: adjacency (x), precedence (+), position (o) and blocks (*). The left graph corresponds to a 50×10 instance taken from the OR-Library and the right graph to a randomly generated 100×20 instance.

These results show that the higher fitness variance corresponds to the adjacency representation (which seems to be notably worse than the other ones), followed by precedence and position representations. The block representation exhibits the smallest variance. It is important to notice that the size of precedence formae (as a function of which fitness variance has been shown) corresponds to the number of elements for which their precedence relations have been fully specified. Otherwise, it would not have been possible to show the variance of precedence formae in the same graph (recall that the total number of precedence formae -and therefore the maximal order- is $O(|Z|^2)$).

In light of these results, the quality of a sequence with respect to makespan seems to be better determined by the absolute position of tasks rather than by relative orderings or immediate neighborhood. Moreover, the smaller fitness variance of block formae shows a good linkage between consecutive positions. Next sections try to study and confirm these findings in the context of traditional and new recombination operators for permutations.

4. Classical Recombination Operators

This section analyzes the functioning of classical recombination operators for permutations. The operators considered are: *partially mapped crossover* (PMX), *cycle crossover* (CX),

directed-edge recombination, three variants of *order crossover*, and the *intersection* and *union* operators. For each of these operators, the formae they manipulate are identified and this manipulation is characterized.

4.1. Variants of order crossover

Order crossover operators are intended to generate offspring that inherit relative ordering information from the ancestors. Thus, these operators seem to be manipulators of precedence formae. However, a closer inspection shows that their functioning is influenced by other factors.

The first variant of order crossover operator (OX#1) was first proposed by Davis (1985). It works selecting two cutpoints into the sequence, copying the elements between these points from one parent and preserving the relative ordering of the rest of elements in the second parent, starting after the second cutpoint and considering the sequence as a ring.

Using this operator, it is ensured that a block forma is transmitted (the subsequence between cutpoints). The size of this forma can be calculated considering that the probability p_i of the block having i elements is

$$p_i = \begin{cases} \dfrac{1}{n} & \text{if } i = 1 \\ \dfrac{2 \cdot (n - i + 1)}{n^2} & \text{if } i > 1 \end{cases} \tag{7}$$

where $n = |Z|$. The expected length of the sequence is then

$$E[i] = \sum_{1 \le i \le n} i \cdot p_i = \frac{1}{n} + \sum_{2 \le i \le n} i \cdot \frac{2 \cdot (n - i + 1)}{n^2} \approx \frac{n}{3} \tag{8}$$

Therefore, a block of $n/3$ elements is transmitted on average. Moreover, this block also carries a complete portion of the precedence matrix corresponding to the elements of the block, and the adjacency relations between them. The second parent should supply the precedence relations for the rest of elements. However, considering the sequence as a ring introduces a strong perturbation in this information. Consider that it is possible that an element placed at the head of the second parent be assigned a position at the tail of the offspring or vice versa, thus altering all precedence relations with the rest of elements (see Fig. 5-left). This situation is empirically tested in a section below.

The second variant (OX#2) was proposed by Syswerda (1991). It can be seen as a kind of uniform crossover for precedence formae. This operator selects some positions at random in the first parent and copies them into the offspring. The remaining positions are taken from the second parent, starting from the beginning and respecting their relative ordering.

If the mask used to select positions in the first parent is generated following a uniform distribution, this operator transmits on average $n/2$ position formae from that parent, including their relative ordering. Moreover, it is ensured that the precedence formae for the rest of elements are transmitted from the second parent. In addition, some position formae from the second parent could be also transmitted if the parents were similar. This is more likely to happen as the population losses diversity and has been empirically tested in subsection 4.6.

Finally, the third variant (OX#3) was also proposed by Davis (1991) and combines the features of the two operators above. As the first operator, two cutpoints are selected and the elements between them are copied. As the second operator, the rest of elements are copied from the beginning of the second parent respecting their relative ordering (Fig. 5-right). Thus, the operator transmits a block forma (with the corresponding precedence and adjacency formae) from one parent and precedence formae for $2n/3$ elements on average from the other

one. Moreover, some position formae from the second parent can be transmitted as well, analogously to OX#2. For these reasons, this should be the best variant of order crossover.

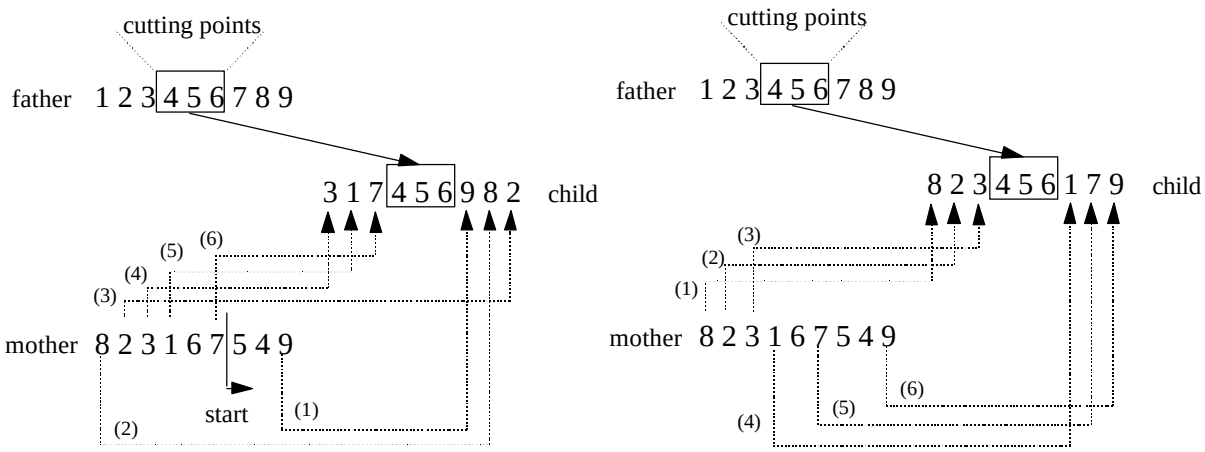


Figure 5. Examples of OX#1 (left) and OX#3 (right). The numbers in brackets show the order in which elements are copied to the child

4.2. Intersection and Union Operators

Since it is not possible to assort and simultaneously respect precedence formae (recall from subsection 3.2.1 that they are non-separable), two operators are defined for each task: Precedence Respectful Recombination (PRR) also known as Intersection Operator and Precedence Assorting Recombination (PAR) or Union Operator (see Fox and McMahon, 1991). Their functioning is explained below.

- *Precedence Respectful Recombination*: The purpose of this operator is to construct offspring containing any common precedence relation. It works as follows: first the intersection of the precedence matrices is computed. Common precedence formae are thus identified. Then, a non-common element is inserted from a random parent, appropriately updating the matrix (i.e., adding transitive links). The process is repeated until the matrix is completed.
- *Precedence Assorting Recombination*: This operator generates offspring that may carry any valid combination of precedence formae. To do this, the elements are partitioned into two disjoint sets, and the ordering of the elements in each set is taken from a different parent. The resulting subsequences are randomly merged.

4.3. Partially Mapped Crossover (PMX)

PMX is an operator proposed by Goldberg and Lingle (1985). It is designed to preserve many absolute positions from both parents. It works selecting two cutpoints in the first parent and copying the elements between them. This transfer also defines a set of mappings between the elements that have been copied and the elements in the corresponding positions in the second parent. Then, the rest of elements are copied in the positions they occur in the second parent. If one position is occupied by an element already copied from the first parent, the element provided by the mappings is considered. This process is repeated until the conflict is solved (see Fig. 6).

This operator transmits a block forma and has the property of being *respectful* (Radcliffe, 1994) with respect to position formae. This means that common position formae are transmitted to the offspring. Moreover, the implicit mutation rate is low since only $n/3$ new position formae would be introduced in the worst average case. These are two desirable

properties that will be reflected in a good performance of the operator as shown later.

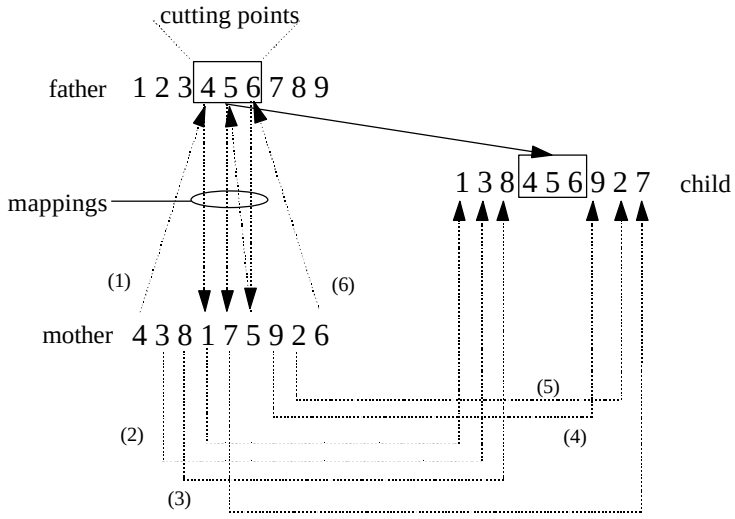


Figure 6. Example of PMX. The numbers in brackets show the order in which elements are copied to the child

4.4. Cycle Crossover (CX)

CX is an operator that was proposed by Oliver *et al.* (1987). It generates offspring in which every position come from one of the parents. Its functioning is based in the concept of *cycle*. A cycle is a minimal subset of elements such that the set of positions in which they appear is the same in both parents. This implies that it is possible to switch that subset from one parent to the other one while keeping a valid permutation. This operator copies the cycle that contains the first element of the first parent in the positions in which they occur in it, taking the rest of positions from the second parent (see Fig. 7).

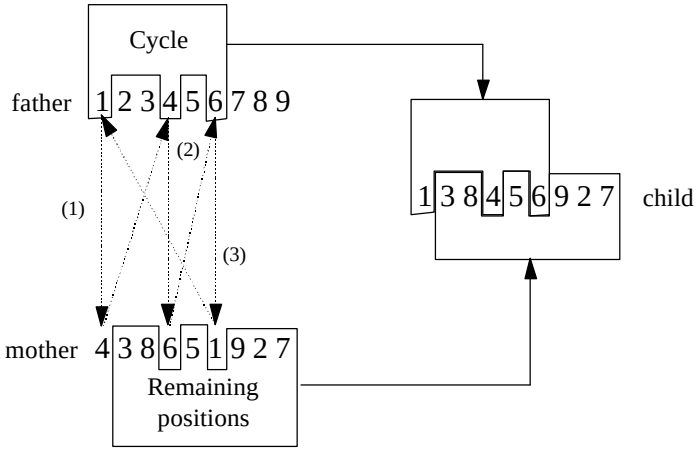


Figure 7. Example of CX. The numbers in brackets show the order in which elements are copied to the child.

This is a strictly transmitting operator for position formae, i.e., it does not introduce any implicit mutation. Since position formae are separable, this operator is also respectful with position formae. These are two positive features of this operator. On the other hand, a potential drawback of this operator is the positional bias introduced when the first cycle is always swapped.

4.5. Directed Edge Recombination (DER)

As its name suggests, the directed edge recombination operator works on the adjacency representation of the permutation. Its functioning is similar to the enhanced edge-recombination operator (Starkweather *et al.*, 1991) but considering only directed edges, i.e., an edge map is built using all edges in the parents and offspring are created taking edges from this list and giving preference to common edges.

This operator transmits as many adjacency formae as possible, trying to respect common edges. Both position formae and precedence formae (except those ones arising from immediate neighborhood) are ignored by this operator, whose use is justified by its good performance on the traveling salesperson problem and by the fact that flowshop problems can be translated into asymmetric TSPs (Stöppler and Bierwirth, 1992). However, the high fitness variance of adjacency formae does not support the use of this operator.

4.6. Empirical Results

To evaluate the performance of the previously defined operators, a collection of experiments has been done. First, the average transmission rates for different formae have been calculated. This has been done randomly generating one thousand pairs of 100-element permutations, and recombining them using the different operators. Subsequently, offspring have been checked to measure the maximum transmission rate from each parent. The results are shown in Table 1.

Table 1. Average transmission rates for different operators and representations. The results are given with respect to a 100-element problem. The size of transmitted precedence formae is calculated as in subsection 3.3.

	Order of transmitted formae			
Operator	Precedence	Adjacency	Position	Block
OX#1	82.60	48.18	34.59	33.10
OX#2	84.59	26.86	51.33	5.68
OX#3	89.21	48.07	35.82	34.34
PMX	84.96	37.67	48.01	34.85
CX	85.16	34.86	51.31	11.67
DER	71.63	41.03	2.08	1.49
PAR	84.18	13.71	5.06	1.52
PRR	86.71	5.05	2.49	0.88

Notice that OX#3 transmits higher-order block and precedence formae than both OX#1 and OX#2. The latter one seems to be more successful at transmitting position formae, but as shown later OX#3 becomes better than OX#2 as the search advances. It must also be noted that PMX and CX have similar transmission rates (although PMX transmits blocks better). This should be reflected in a comparable performance.

Next, the performance of these operators has been tested, using a genetic algorithm with the parameters shown in Table 2. The algorithm uses the *swap* mutation operator (Manderick *et al.*, 1991), whose functioning consists of selecting two positions at random and swapping their contents. For each operator and test problem, ten runs have been done. The results are shown in Table 3.

Table 2. Parameters of the genetic algorithm

Population size	100
Evolution model	steady state w/o duplicates
Selection mechanism	linear ranking (η^+=2.0, η^-=0.0)
Probability of crossover	.9
Probability of mutation	1.0 / numberOfTasks
Mutation operator	swap
Total number of evaluations	100.000

Table 3. Average makespan. These results correspond to ten runs of the algorithm

Operator	**Problem Instance**						
	rec19 30×10	**rec25** 30×15	**rec31** 50×20	**rec37** 75×20	**r100** 100×20	**r125** 125×30	**r200** 200×50
OX #1	2127.8	2574.5	3146.6	5229.1	6649.7	9013.4	15352.6
OX #2	2123.1	2572.2	3151.1	5249.2	6644.2	9060.8	15399.7
OX #3	2119.3	2560.9	3127.7	5196.7	6583.1	8970.5	15238.2
PTR	2139.8	2570.9	3139.7	5260.1	6688.9	9085.7	15377.0
PAR	2149.5	2603.4	3180.7	5299.1	6744.2	9206.8	15588.0
PMX	2120.4	2567.6	3127.3	5185.2	6581.8	8924.7	15133.7
CX	2129.5	2572.2	3129.1	5202.5	6596.1	8936.4	15171.7
DER	2144.5	2593.3	3182.7	5315.0	6759.7	9182.1	15583.7

These results clearly agree with the predictions extracted from the analysis of the fitness variance of formae. First, notice that the directed edge recombination operator provides poor results due to the high variance of adjacency formae. Furthermore, it has a very high rate of implicit mutation in both position formae and precedence formae (Fig. 8-left).

Also, it can be seen that the third variant of order crossover is the best one. As hypothesized, the implicit mutation rate for both position and precedence formae is smaller as the algorithm evolves (see Fig. 8). However, it decreases faster in OX#3 than in OX#1 or OX#2. This fact, along with the block transmission it ensures, determines its good behavior. For similar reasons, PMX is the best operator. As OX#3, it transmits a block forma and has a very low implicit mutation rate for position and precedence formae.

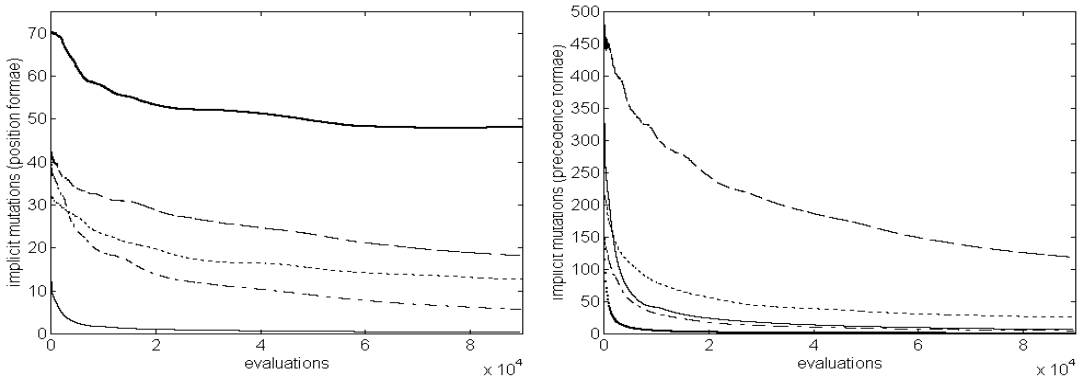


Figure 8. Implicit mutation in different operators (rec37 problem). The solid line corresponds to PMX, the dashed line to OX#1, the dotted line to OX#2, the dashdotted line to OX#3 and the thick line to DER (in the left graph) and CX (in the right graph).

On the other hand, the performance of CX is not as good as it should be. The reason can be found in the mentioned positional bias that takes place when the first cycle is always selected. This may cause a strong convergence in the first positions, causing the operator to be useless. This hypothesis is supported by the results that a new operator presented in the next section (RCX - Random Cycle Crossover) provides.

Finally, notice the bad results of PRR and, mainly, PAR. This fact admits two explanations. On the one hand, these operators have a very high rate of implicit mutation in position formae (see Fig. 9). On the other hand, precedence formae have a higher variance than position formae and therefore the algorithm can be more easily misled.

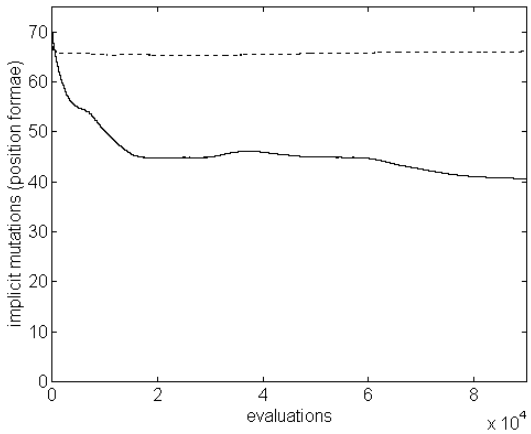


Figure 9. Implicit mutation for position-formae in precedence-based operators (rec37 problem). The solid line corresponds to PRR and the dotted line to PAR.

5. New Recombination Operators

The empirical results obtained in the previous sections show that the most effective operators work on position- and block-based representations. This sections tries to confirm this conclusion. For that purpose, four new operators are designed and their effectiveness is studied. Two of these operators (RCX and UCX) work on position formae. The other two ones (BX and UBX) manipulate block formae.

5.1. Random Cycle Crossover (RCX) and Uniform Cycle Crossover (UCX)

As mentioned before, one of the drawbacks of standard CX is that it always selects the cycle that contains the first element of a parent. This admits a straightforward solution: to select a cycle that contains a random position. Using this simple modification, it is possible to distribute the interchange of formae across all positions of the sequence. The so-obtained operator will be called Random Cycle Crossover (RCX).

On the other hand, it was stated in section 3.2.3 that position formae are separable, i.e., it is possible to simultaneously respect and assort position formae. Such an operator can be defined on the basis of cycle interchange. This operator will be called Uniform Cycle Crossover and can be seen as a kind of uniform crossover for position formae. Its functioning is as follows: first, all cycles are identified. Subsequently, a test is done to decide from which parent each cycle should be taken. The amount of information that is copied from a single parent is controllable by means of a change in the distribution of the test results. Fig. 10 shows an example of RCX and UCX.

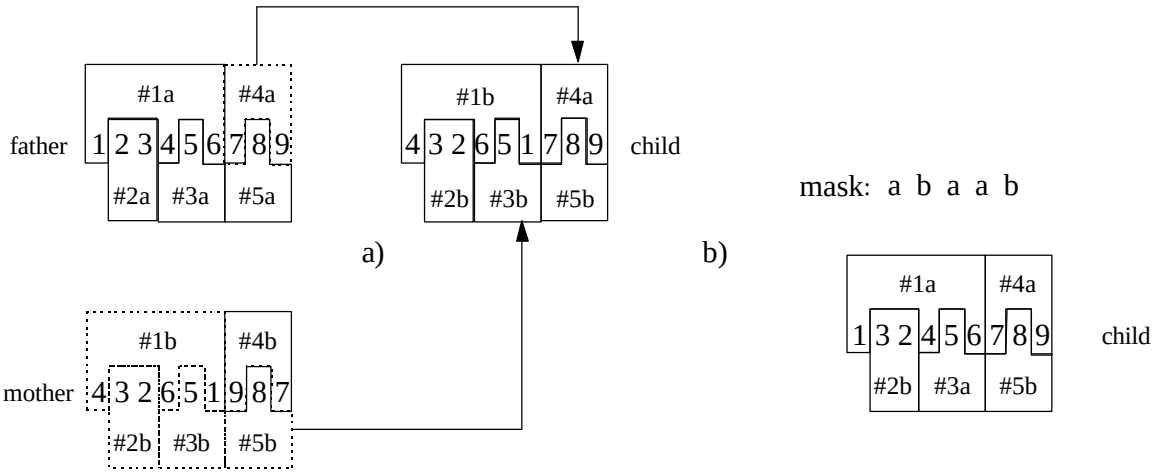


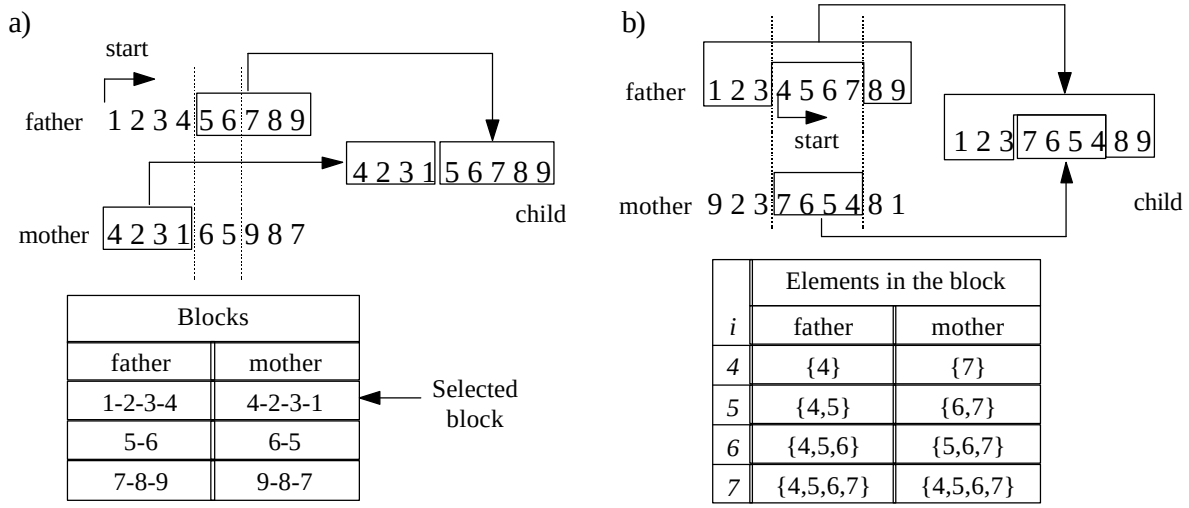
Figure 10. Examples of RCX (left) and UCX (right). RCX selects a random position and copies the cycle containing that position. UCX generates a mask to decide from which parent each cycle is copied.

5.2. Block Crossover (BX) and Uniform Block Crossover (UBX)

These operators are analogous to RCX and UCX respectively, working with block formae. Starting from a given position, a block is found consecutively adding elements until the elements in the block are a permutation of the corresponding elements in the other parent. BX works selecting a random block in one parent and copying it in the offspring. The remaining positions are taken from the other parent. There exist two possibilities to select such a random block:

- Start from the first element of the sequence, identify all blocks and randomly select one (BX#1).
- Select a random position and extract the block from that position, considering the sequence as a ring if necessary (BX#2).

Fig. 11 shows an example of the application of these two versions of the operator.



Blocks	
father	mother
1-2-3-4	4-2-3-1
5-6	6-5
7-8-9	9-8-7

Selected block

	Elements in the block	
i	father	mother
4	{4}	{7}
5	{4,5}	{6,7}
6	{4,5,6}	{5,6,7}
7	{4,5,6,7}	{4,5,6,7}

Figure 11. Examples of BX#1 (left) and BX#2 (right). BX#1 selects a random block, while BX#2 extracts a block from a random position.

As to UBX, it first identifies all blocks (starting from the beginning) and generates a random mask that determines from which parent each block is copied (Fig 12).

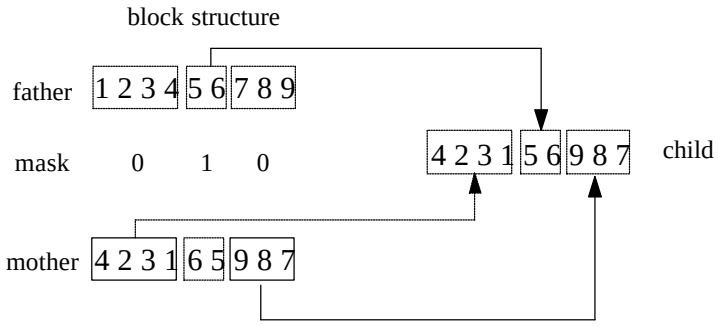


Figure 12. Example of Uniform Block Crossover

Notice that both BX and UBX are respectful with simple position formae. Moreover, they strictly transmit precedence formae too, i.e., they introduce no implicit mutation either in position or in precedence. In fact, only an exogenous adjacency relation is included for each change from 0 to 1 (and vice versa) in the mask.

5.3. Empirical Results

Experiments with these operators have been done using the same parameter settings and problem instances as in section 4.6. The results are shown in Table 4. PMX has been included as a reference.

Table 4. Average makespan (10 runs).

Operator	**Problem Instance**						
	rec19 30×10	**rec25** 30×15	**rec31** 50×20	**rec37** 75×20	**r100** 100×20	**r125** 125×30	**r200** 200×50
RCX	2128.8	2571.2	3126.1	5188.8	6595.1	8948.1	15159.7
UCX	2127.1	2577.3	3125.4	5197.0	6567.7	8923.4	15137.8
BX #1	2115.9	2573.0	3128.7	5184.6	6571.6	8900.5	15183.6
BX #2	2120.4	2561.6	3121.6	5190.0	6605.2	8932.8	15166.8
UBX	2113.6	2554.7	3134.2	5196.2	6594.8	8906.4	15157.2
PMX	2120.4	2567.6	3127.3	5185.2	6581.8	8924.7	15133.7

As it can be seen, these operators are competitive with PMX and, in several cases, provide better results. This confirms the correlation between the manipulation of position and block formae and good performance. Notice also that RCX performs better on average than the standard CX, thus supporting the hypothesis presented in section 4.6.

6. Conclusions and Future Work

This work has studied four different representations of permutations with application to flowshop scheduling problems. The fitness variance of the formae each representation induces has been used as a metric to assess the quality of these representations with respect to a target measure (makespan). The obtained results show that the salient features of a given scheduling for that measure are more dependent on the absolute positions of the tasks than on their relative orderings or adjacency relations. Moreover, there exists a strong linkage between contiguous positions, as the lowest variance of block formae shows.

Subsequently, the classical operators for permutation recombination have been analyzed in terms of the formae they manipulate and the implicit mutation they introduce. The empirical tests that have been carried out are consistent with that analysis and with the results regarding

fitness variance. Thus, position-based operators as PMX exhibit the best performance. On the other hand, operators exclusively acting on adjacency (DER) or precedence (union and intersection) provide the worst results. The reason is the higher fitness variance of the formae they manipulate (which contributes to mislead the algorithm) and the high rates of implicit mutation for position formae they induce.

Four new operators have been defined. One of them (RCX) is a simple modification of CX that provides better results than the latter. The other three operators have been designed to combine simple position formae (UCX) and block formae (BX and UBX). The results they provide are encouraging. They are not only competitive with PMX but even better in some of the test problems.

Notice that the analysis presented in this work has been oriented to a predefined objective (minimizing makespan). This implies that the obtained ranking of representations and operators is not necessarily generalizable to different fitness functions. In fact, there might exist a fitness function whose solutions were better represented in terms of precedence, adjacency or even by another feature not considered here. This could be determined using an analogous analysis of fitness variance and forma manipulation. In this sense, the methodology seems to be generalizable.

Some comments must also be done on the analysis of preexisting operators. It is usually the case that some operators are "hybrid" in the sense that they simultaneously manipulate different representations. Performance prediction is then harder since it may be difficult to determine the dominant representation. Notice that this dominant representation may even change as the search advances due to factors such as the implicit mutation. A deeper analysis is required in these situations. This constitutes a line of future work.

References

Antonisse, J. (1989). A New Interpretation of Schema Notation that Overturns the Binary Encoding. In Schaffer, J.D. (ed.), *Proceedings of the Third International Conference on Genetic Algorithms* (pp. 86-91), San Mateo, CA: Morgan-Kaufmann

Beasley, J.E. (1990). OR-Library: Distributing Test Problems by Electronic Mail. *Journal of Operational Research Society*, 41(11): 1069-1072

Bellman, R.E. & Dreyfus, S.E. (1962). *Applied Dynamic Programming*, Princeton University Press

Davis, L. (1985). Applying Adaptive Algorithms to Epistatic Domains. In *Proceedings of the International Joint Conference on Artificial Intelligence* (pp. 162-164), Morgan Kaufmann, Los Angeles CA.

Davis, L. (1991). *Handbook of Genetic Algorithms*, New York:Van Nostrand Reinhold Computer Library

Fox, B.R. & McMahon M.B. (1991). Genetic Operators for Sequencing Problems. In Rawlins G.J.E. (ed.), *Foundations of Genetic Algorithms 1* (pp. 284-300). San Mateo CA: Morgan-Kaufmann.

Garey, M. & Johnson, D. (1979). *Computers and intractability: A guide to the theory of NP-Completeness*, San Francisco: Freeman and Co.

Goldberg, D.E. & Lingle Jr. R. (1985). Alleles, loci and the traveling salesman problem. In Grefenstette, J.J. (ed.), *Proceedings of an International Conference on Genetic Algorithms*, Hillsdale: Lawrence Erlbaum Associates

Hart, W.E. & Belew, R.K. (1991). Optimizing an arbitrary function is hard for the genetic algorithm. In Belew, R.K. & Booker, L.B. (eds.), *Proceedings of the Fourth International Conference on Genetic Algorithms* (pp. 190-195), San Mateo CA: Morgan-Kaufmann

Hillier, F.S. & Lieberman, G.J. (1967). *Introduction to Operations Research*. San Francisco CA: Holden-Day

Holland, J.H. (1975). *Adaptation in Natural and Artificial Systems*. Ann Arbor:University of Michigan Press

Lawler, E.L. & Wood, D.E. (1966). Branch and Bounds Methods: A survey. *Operations Research* 14: 699-719

Manderick, B., de Weger, M. & Spiessens, P. (1991), The Genetic Algorithm and the Structure of the fitness Landscape. In Belew, R.K. & Booker, L.B. (eds.), *Proceedings of the Fourth International Conference on Genetic Algorithms* (pp. 143-150). San Mateo CA: Morgan-Kaufmann

Michalewicz, Z. (1992). *Genetic Algorithms + Data Structures = Evolution Programs*. Berlin-Heidelberg: Springer-Verlag

Oliver, I.M., Smith, D.J. & Holland, J.R.C. (1987). A Study of Permutation Crossover Operators on the Traveling Salesperson Problem. In Grefenstette, J.J.(ed.), *Proceedings of the Second International Conference on Genetic Algorithms and their Applications* (pp. 224-230). Hillsdale: Lawrence Erlbaum Associates

Radcliffe, N.J. (1991). Equivalence Class Analysis of Genetic Algorithms. *Complex Systems* 5: 183-205

Radcliffe, N.J. (1992). Non-Linear Genetic Representations. In Männer, R. & Manderick, B. (eds.), *Parallel Problem Solving From Nature 2,* Elsevier Science Publishers, Amsterdam, pp. 259-268

Radcliffe, N.J. (1994). The Algebra of Genetic Algorithms. *Annals of Mathematics and Artificial Intelligence* 10: 339-384

Radcliffe, N.J. & Surry, P.D. (1994). Fitness Variance of Formae and Performance Prediction. In Whitley, D. & Vose, M.D. (ed.), *Foundations of Genetic Algorithms 3*. Morgan-Kaufmann

Reeves, C.R. (1993) *Modern Heuristic Techniques for Combinatorial Problems*. Oxford: Blackwell Scientific Publications

Starkweather, T., McDaniel, S., Mathias, K., Whitley, D. & Whitley, C. (1991). A comparison of Genetic Sequencing Operators. In Belew, R.K. & Booker, L.B. (eds.), *Proceedings of the Fourth International Conference on Genetic Algorithms* (pp. 69-76). San Mateo CA:Morgan-Kaufmann

Stöppler, S. & Bierwirth, C. (1992). The Application of a Parallel Genetic Algorithm to the $n/m/P/C_{max}$ Flowshop Problem. In Fandel, G., Gulledge, Th. & Jones, A. (eds.), *New Directions for Operations Research in Manufacturing* (pp. 161-179). Springer-Verlag

Syswerda, G. (1991). Schedule Optimization using Genetic Algorithms. In Davis, L. (ed.), *Handbook of Genetic Algorithms*, New York:Van Nostrand-Reinhold Computer Library, pp. 332-349

Vose, M.D. (1991). Generalizing the notion of schema in genetic algorithms. *Artificial Intelligence* 50:385-396

Wolpert, D.H. & Macready, W.G. (1995). No Free Lunch Theorems for Search, Sante Fe Institute Technical Report SFI-TR-95-02-010