Topological Crossover for the Permutation Representation

Alberto Moraglio and Riccardo Poli

Department of Computer Science, University of Essex,
Wivenhoe Park, Colchester, CO4 3SQ, UK
{amoragn, rpoli}@essex.ac.uk

Abstract. Topological crossovers are a class of representation-independent operators that are well-defined once a notion of distance over the solution space is defined. In this paper we explore how the topological framework applies to the permutation representation and in particular analyze the consequences of having more than one notion of distance available. Also, we study the interactions among distances and build a rational picture in which pre-existing recombination/crossover operators for permutation fit naturally. Lastly, we also analyze the application of topological crossover to TSP.

1 Introduction

The permutation representation (see [7] for an introduction) is one of the most-frequently used representations in evolutionary algorithms. Many combinatorial optimization problems, including TSP and scheduling problems, are naturally cast using permutations. The success of the permutation representation is due to its flexibility.

When applied to permutations, traditional crossover operators used for binary strings can produce invalid offspring. So, researchers have come up with a variety of operators specifically designed for permutations. They range from general-purpose operators working reasonably well on a wide spectrum of problems, such as the partially matched crossover (PMX), to specialized operators that work best on a specific class of problems [5], such as edge recombination crossover (ERX) for TSP.

Topological crossovers [9] are a class of *representation-independent* operators that are well-defined once a notion of distance over the solution set is defined. Simply stated, the offspring they produce are *between* their parents. This simple definition has surprising implications, including a powerful way to do crossover design for *any* representation and the potential for the development of a general theory of evolutionary algorithms encompassing *all* representations.

In previous work [9] we have shown how topological crossover generalizes the notion of crossover for binary strings. Differently from the binary string case, for which a single natural distance (the Hamming distance) is defined, permutations allow for various notions of distance that are all equally natural. In this paper, we explore how our topological framework applies to the permutation representation and in particular analyze the consequences of having more than one notion of distance available. We

study the interactions among distances and we build a rational picture in which pre-existing recombination/crossover operators fit naturally. As an important example, we also analyze in detail the application of topological crossover to TSP.

In section 2, we introduce the topological framework. Section 3 introduces various notions of distance for permutations. Section 4 focuses on distances based on permutation interpretation and discusses the difficulty with topological crossovers based on such distances. Section 5 introduces various edit distances and show that topological crossover is naturally suited to edit distances. Section 6 draws a parallel between "interpretation" distances and edit distances and suggest that topological crossovers defined over edit distances can be thought as topological crossovers defined over the corresponding interpretation distances. In section 7, we suggest that many pre-existing recombination operators for permutations are in fact topological crossovers under different edit distances. Section 8 gives an analysis of topological crossover for the TSP problem, discussing problems in its applicability and solutions. In section 9 we present our conclusions.

2 Topological Framework

A *configuration space C* is a pair (*G, Nhd*) where *G* is a set of syntactic configurations (syntactic objects or genotypes) and $Nhd : G \to 2^G$ is a syntactic neighborhood function which maps every configuration in *C* to the set of all its neighbor configurations in *C* which can be obtained by applying any unitary *edit move* from a pre-specified set. The neighborhood function must be symmetric ($y \in Nhd(x) \Leftrightarrow x \in Nhd(y)$, which is to say edit moves are reversible) and connected (any configuration can be transformed into any other in a finite number of moves). A configuration set may lead to more than one configuration space if multiple syntactic neighborhood functions are available. A configuration space *C*=(*G, Nhd*) is said to be a space endowed with a *neighborhood structure*. This is induced by the syntax of the configurations and the particular syntactic neighborhood function adopted. Such a neighborhood structure can be associated with an undirected *neighborhood graph W= (V, E)*, where *V* is the set of vertices representing configurations and *E* is the set of edges representing the relationship of neighborhood between configurations. Since the neighborhood structure is symmetric and connected, this space is also a *metric space* provided with a *distance function d* induced by the neighborhood function. Both *Nhd* and *d* identify univocally the structure of the space, so we can equivalently write *C*=(*G, Nhd*) or *C*=(*G, d*). Distances arising from graphs are known as *graphic distances.* A *fitness landscape F* is a pair *(C, f)* where *C*=(*G, d*) is a configuration space and $f : G \to R$ is a *fitness function* mapping a syntactic configuration to its *fitness value*. In [9] we have defined two *classes* of *representation-independent operators* using the notion of *distance* associated to the *landscape*: topological mutation and topological crossover. We give the main definitions and properties for topological crossover below since these are the starting point for the work on permutations reported in this paper. In a metric space (S, d) a line segment is the set of the form $[x; y] = \{ z \in S \mid d(x, z) + d(z, y) = d(x, y) \}$

where $x, y \in S$ are called extremes of the segment. A g-ary genetic operator OP takes g parents $p_1, p_2, \ldots p_g$ and produces one offspring c according to a given conditional probability distribution:

$$\Pr\{OP(p_1, p_2, \ldots p_g) = c\} = f_{OP}(c \mid p_1, p_2, \ldots p_g)$$

Definition: *The* image set *of a genetic operator OP is the set of all possible offspring produced by OP when the parents are* $p_1, p_2, \ldots p_g$ *with non-zero probability:*

$$\mathrm{Im}[OP(p_1, p_2, \ldots p_g)] = \{c \in S \mid f_{OP}(c \mid p_1, p_2, \ldots p_g) > 0\}$$

Definition: *A binary operator CX is a* topological crossover *if:*

$$\mathrm{Im}[CX(p_1, p_2)] \subseteq [p_1; p_2]$$

This simply means that in a topological crossover offspring lay *between* parents.

Definition: Topological uniform crossover *UX is a topological crossover where all z laying between parents x and y have the same probability of being the offspring:*

$$f_{UX}(z \mid x, y) = \Pr\{UX = z \mid P1 = x, P2 = y\} = \frac{\delta(z \in [x, y])}{|[x, y]|}$$

$$\mathrm{Im}[UX(x, y)] = \{z \in S \mid f_{UX}(z \mid x, y) > 0\} = [x, y]$$

where δ *is a function which returns 1 if the argument is true , 0 otherwise.*

Theorem: *The structure over the configuration space C can equivalently be defined by the set G of the syntactic configurations and one of the following objects: 1. the neighborhood function Nhd, 2. the neighborhood graph W= (V, E), 3. the graphic distance function d, 4. uniform topological crossover UX, 5. the set of all segments* $\boldsymbol{H}$.

Corollary: *Given a structure of the configuration search space in terms of neighborhood function or graphic distance function, UX is unique.*

Corollary: *Given a representation, there are as many UX operators as notions of graphic/syntactic distance for the representation.*

The following two properties apply to binary strings.

Theorem: *All mask-based crossover operators for binary strings are topological crossovers.*

Theorem: *The topological uniform crossover for the configuration space of binary strings endowed with Hamming distance is the traditional uniform crossover.*

3 Distances Between Permutations

Differently from binary strings where a single, natural definition of distance, the Hamming distance, is normally used, for permutations many notions of distance are equally natural. This situation is further complicated by the fact that such distances relate to each others in various ways with subtle dependencies. Further complication arises from the fact that permutations and circular permutations (and also permutations with repetitions to a lesser extent) are treated as if they were the same representation, which is incorrect. Indeed, although they are connected, they are different representations and they allow for different notions of distance. For a survey on metrics on permutations see [4].

Distances for permutations may have different origins:

a) Notions of distance arising from the *interpretation* of permutations: these measure the distance between the objects represented by two permutations.

b) Notions of distance directly connected with the *syntax*: these distances measure how two permutations differ in their syntax.

c) Notions of distance connected with the notion of mutation or *edit* distance: these distances measure the minimum number of moves necessary to transform a permutation into another by the application of a syntax modification operator.

These three types of distances are interdependent. For example, an edit distance is also a syntactic distance but a syntactic distance is not necessarily an edit distance (one can define a measure of syntactic similarity that is not defined on edit moves). Also, an edit distance is also a graphic distance (see section 2), but a graphic distance is not necessarily an edit distance[1]. In the following three sections, we study the connections between these 3 notions of distance and their suitability as bases for topological crossover.

In principle, given *any* notion of distance over the solution set, the corresponding topological crossover and topological mutation operators are well-defined. What is a good distance then? A good distance is one that (i) makes a given landscape (i.e. a solution set with its fitness function plus a notion of distance) easy to search for the specific search operator employed[2] and that (ii) allows such operator to be implemented efficiently. The first point connects with *landscape design*, the second with *distance duality*. Here we concentrate on the latter.

Topological crossover and mutation are well-defined for any notion of distance (whether based on syntax or not). So, operators are well-defined independently from the underlying representation. In practice, however, the genetic operators have to be implemented. If they are not based on a notion of edit distance that links them tightly to the solution representation, they become difficult or even *impossible to implement efficiently.* To understand the reasons for this we need to stress the *geometric nature* of these operators and of the *geometric interpretation* of the landscape coming with them, and we need to introduce the notion of *distance duality*. The notion of edit distance arising from the syntax of configurations has a natural dual interpretation:

a) seen in the configuration space, it is a measure of *similarity* (or dissimilarity) between two syntactic objects

b) seen in the neighborhood graph, it is the *length of the shortest path* connecting two vertices and, therefore, it is a measure of *spatial remoteness* between points when interpreting such a structure in a *geometric sense.*

For each representation and edit move definitions, this duality manifests itself in a different way. In the case of permutations the duality implies that picking elements in the segment (shortest path) is equivalent to picking elements on a minimal sorting

[1] It can be argued that any graphic distance is an edit distance under an arbitrary set of edit operations. However we consider "simple" edit distances based on a small set of moves with a compact syntactic definition.

[2] A rule of thumb to pick a good distance as a base for geometric crossover is to choose a distance that is meaningful for the interpretation of the solution representation (phenotype).

trajectory from one parent permutation to the other. This connection between the geometric notion of "belonging to a segment" and its syntactic dual of "being on a minimal sorting trajectory" is ultimately what allows topological crossover to be *actually implemented* in an efficient way. So, even if a topological crossover is representation-independent, when dealing with its implementation the specific representation used makes indeed the difference between an *efficient* operator and an *inefficient* one.

4 Permutation Interpretations and Related Distances

Permutations can be used to represent solutions to different types of problems for which different relations among the elements in the permutation are relevant. There are three major interpretations of a permutation [1]. For example, in TSP permutations represent tours and the relevant information is the *adjacency relation* among the elements of a permutation. In resource scheduling problems permutations represent priority lists and the relevant information in this case is the *relative order* of the elements of a permutation. In other problems, the important characteristic is the *absolute position* of the elements in the permutation.

Let us consider the permutation (C D E B F A). If the adjacency is important then the fact that the elements D and E are adjacent is relevant as well as the fact that the elements C and B are not adjacent. If the important aspect is the relative order then what is relevant is the fact that D precedes E and that C precedes B. If the absolute order is important then the relevant point is that C is in position 1, D in position 2, etc. For each interpretation of permutation, it is possible to write a binary matrix that represents the actual relation among elements in the permutation. So, we can have a relative order matrix, an absolute position matrix and an adjacency matrix. For example, for the permutation (C D E B F A) we have the following matrices:

relative order matrix:

	A	B	C	D	E	F
A	0	0	0	0	0	0
B	1	0	0	0	0	1
C	1	1	0	1	1	1
D	1	1	0	0	1	1
E	1	1	0	0	0	1
F	1	0	0	0	0	0

absolute position matrix:

	A	B	C	D	E	F
Pos1	0	0	1	0	0	0
Pos2	0	0	0	1	0	0
Pos3	0	0	0	0	1	0
Pos4	0	1	0	0	0	0
Pos5	0	0	0	0	0	1
Pos6	1	0	0	0	0	0

adjacency matrix:

	A	B	C	D	E	F
A	0	0	0	0	0	1
B	0	0	0	0	1	1
C	0	0	0	1	0	0
D	0	0	1	0	1	0
E	0	1	0	1	0	0
F	1	1	0	0	0	0

It is possible to define three distance functions for permutations based in the corresponding relative order matrices, absolute position matrices and adjacency matrices. The distance between two permutations is then the Hamming distance between their corresponding matrices in the three interpretations. We refer to these distances as *relative order distance* (ROD), *absolute position distance* (APD) and *adjacency distance* (AD).

For example, the segments between two permutations under ROD include all the *well-formed* permutations (not any arbitrary string) where the relative order relation of the parent permutations is transmitted perfectly.

In principle, topological operators can be defined using these notions of distance. So we can define *rigorously* relative order topological crossover (ROX) and mutation (ROM), absolute position topological crossover (APX) and mutation (APM), and adjacency topological crossover (AX) and mutation (MX). However, ROD, APD and AD are not straightforwardly connected with edit distances for permutations and therefore ROX, APX and AX may result difficult, if not impossible, to implement exactly in an efficient way.

Each interpretation distance is also connected with a notion of *syntactic* distance between permutations that is not necessarily an *edit* distance. ROD is connected with the *offset distance* that sums for each element the number of positions away is in the two permutations. APD is connected with the *Hamming distance* for permutations that, analogously to the Hamming distance for strings, is the number of mismatches at corresponding positions when the two permutations are aligned. AD is connected with the *breakpoint distance* that counts the occurrences of two elements being adjacent in a permutation *and* non-adjacent in the other. Topological operators in principle can be defined using these notions of syntactic distance but then and again, since these are non-edit distances, the corresponding genetic operators are hard to implement.

5 Classical Mutations, Edit Distances and Topological Crossover

Various mutation operators are defined for permutations. The most common are [1]:

Inversion or 2-change (block-reversal): This operator selects two points along the string then reverses the segment between the points. It is particularly well-suited for the TSP and for all the problems that naturally admit a permutation representation in which adjacency among elements plays an important role.

Insert and block-transposition: This operator selects one element and inserts it at some other position in the permutation. There is a non-reversible variant: one selects two elements and then moves the second element before the first. These operators are used with scheduling problems in which relative order of elements is important.

Swap and adjacent swap (two-element swap): The swap operator selects two elements and swaps their positions. The adjacent swap swaps two contiguous elements.

Scramble: This operator selects a sub-list and randomly reorders the elements while leaving the other elements in the permutation in the same absolute position.

Each notion of mutation is connected with a notion of *edit distance* for permutations. Therefore, we can talk of *reversal distance*, *transposition distance*, *swap distance*, *adjacent swap distance*, *scramble distance* and so on. Notice that there are a number of variations for each of these distances which result from imposing constraints on the edit move.

For each notion of edit distance there is a corresponding notion of topological crossover. So we can define many possible crossovers for permutations, each induced from a corresponding mutation. Since these are crossovers based on similar, but not identical, neighborhood structures, they will tend to have similar behaviors. So, what are the *important topological crossovers* then? We propose an initial answer to this question in the next section. However, not all topological crossovers based on edit

distances have efficient implementations. Indeed, constraints on edit moves transform the complexity of crossover from polynomial to NP-hard [13].

Because of the distance duality, a point on a segment between two permutations, under a given edit distance, is on a minimal sorting trajectory connecting the two permutations. This allows to actually implementing such crossovers by *sorting algorithms*. Some edit distances give rise to a crossover that can be implemented exactly and efficiently. Other edit distances give rise to crossovers that are possible to implement efficiently (in polynomial time) only in an approximated way. Quite interestingly, bubble sort and insertion sort fit the definition of topological crossover for, respectively, the adjacent swap distance and the swap distance. So, *ordinary sorting algorithms can actually be used as crossovers*!

6 Relations Between Edit Distances and Interpretation Distances

Depending on the interpretation of the permutation, the same mutation operator can be seen as a small change or a major change. For example, the inversion operator does a minimal change when one thinks of a permutation in terms of adjacency, but a major change when the same permutation is seen as a priority list (relative order).

A single mutation should represent a minimal change [10] [11]. According to this principle, there are three mutation operators that do a different minimal change in a permutation, one for each interpretation. When the permutation is thought as an adjacency relation then the minimal mutation operator is the inversion operator: while reversing the order of a sub-list, only two adjacency links (edges) are changed. When the permutation represents a relative order the minimal mutation operator is the adjacent swap operator that affects only the relative order of a pair of elements. When the absolute position of elements in the permutation is relevant, the minimal mutation operator is the swap operator that changes the absolute positions of only two elements.

A neighborhood function (see section 2) induced by the syntax must be symmetric and connected. The inversion operator is symmetric (re-reversing the same sub-list produces the original permutation) and connected (by repeated reversions it is possible to reach any permutation from any other permutation). Also the adjacent swap operator is symmetric and connected (bubble sort based on adjacent swap is able to sort any permutation of elements). The same holds for the swap operator.

The adjacent swap operator can be seen as a special case of swap as well as a two-element sub-list inversion operator. Its neighborhood is the intersection of the other two operators' neighborhoods.

7 Existing Crossovers and Permutation Interpretations

There are a number of crossover operators defined for permutation (for a good overview, see [1][2]). Most of them were devised with a *specific interpretation* of the permutation in mind. This is reflected in their names. So, for example, Davis's *order crossover* emphasizes the fact that a permutation is seen as a relative order, *cycle*

crossover preserves absolute positions, and *edge recombination crossover* focuses on the adjacency relation of the elements in the permutation.

Some crossovers achieve their goals of transmitting a specific relationship among elements from the parents to the children perfectly (*perfect crossovers*), others achieve their goals only approximately (*imperfect crossovers*). For example cycle crossover transmits perfectly the common positional information of parents to children and so is a perfect crossover; both Davis's order crossover and edge recombination are imperfect crossovers in that they are not able to transmit perfectly, respectively, the common relative order of the parents and the adjacency relation. However, the common relative order is much easier to transmit perfectly than the adjacency relation. Indeed, another crossover, the merge crossover, perfectly transmits the relative order of parents to children.

Some crossover operator is deliberately designed to be a trade-off, transmitting part of the relative order, part of the absolute position and part of the adjacency relation present in the parent permutations to the offspring permutations (*hybrid crossovers*). This is indeed possible since the three relations have subtle interdependencies. One of such crossover operators is the partially matched crossover. Hybrid crossovers have the advantage to work reasonably well independently from the specific interpretation of the permutation. However when hybrid crossovers are compared with perfect crossovers for a specific interpretation of the permutation on a problem in which this interpretation is relevant, the hybrid ones perform much worse than the perfect ones.

8 Topological Crossover for TSP

Edge recombination is an operator expressly designed for TSP. It considers a solution as a tour of cities and, therefore, rather than being defined for permutations is defined over *circular permutations*. In its various improvements its stated objective is to greedily recombine parent tours in order to transmit as much as possible the adjacency relation, introducing in the offspring tours the minimum number of "foreign" edges not present in either parent [1].

As in the linear case, also for circular permutations it is possible to write an adjacency matrix. Again, the segment between the parent circular permutations (under Hamming distance for the adjacency relation matrix) contains all the feasible offspring circular permutations that perfectly respect the adjacency relation of their parents. The topological crossover for circular permutations under this notion of distance is well-defined and actually achieves what *edge recombination can only aspire to*.

However, the notion of distance based on adjacency matrix is not an edit distance and, therefore, the corresponding crossover operator is hard or impossible to implement efficiently. In the case of circular permutations, the block-reversal move is the notion of edit distance closest to the adjacency matrix distance. In a single application to a tour, this does the minimal change to the adjacency relation among elements in the permutation. This move is the well-known *2-change* move, and it is the basis for successful local search algorithms for TSP [6]. Figure 1 shows the possible offspring (the segment) between two circular (parent) permutations under topological crossover.

Analogously to the linear case, the circular permutations in the segment under reversal distance are those laying in a minimal sorting trajectory from a parent circular permutation to the other. Sorting circular permutations by reversals is NP-hard [12]. So, *the topological crossover under this notion of distance cannot be implemented efficiently*.

Sorting circular permutations by reversals is tightly connected with the problem of sorting linear permutations by reversals. So all the algorithms developed for the latter task can be used with minor modifications also for the former [12]. Sorting linear permutations by reversals is NP-hard too [3]. However a number of approximation algorithms (running in polynomial time) exist to solve this problem within a bounded error from the optimum [8]. This allows implementing efficiently approximate crossovers whose image set is a super-set of that of the exact crossover.

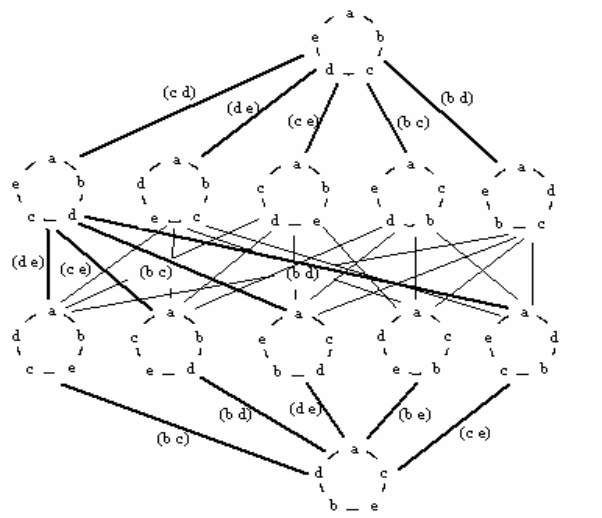

Fig. 1. Example of topological crossover between two circular permutations

9 Conclusions

Topological crossover and mutation are well-defined once one has a notion of distance over the solution set. The permutation representation allows for three notions of non-edit distances connected with the permutation interpretations. Three topological crossovers based on the permutation interpretations are therefore well-defined. However, such topological crossovers are hard or even impossible to implement efficiently, in that they are based on non-edit distances.

Most of the pre-existing crossover operators for permutations are designed around interpretations. We have shown that they fit, some exactly and some imperfectly, the

topological crossover definitions connected with permutations interpretations. The permutation representation also allows for a number of edit distances connected with various notions of mutation. Each notion of edit distance induces a notion of topological crossover. Because of the distance duality, under a given edit distance a point on a segment between two permutations is on a minimal sorting trajectory connecting the two permutations. This allows implementing such crossovers using *sorting algorithms*. Some edit distances give rise to crossovers that can be implemented exactly and efficiently. Other edit distances give rise to crossovers can be implemented efficiently (in polynomial time) only in an approximated way.

The three topological crossovers induced by permutation interpretations are tightly connected with three topological crossovers based on edit distances. The connection relies on the principle of "minimal change". In future research we will investigate this connection in greater depth.

Circular permutations are tightly connected to traditional permutations but they do not coincide. We have shown how to apply topological crossover to TSP that is naturally defined over circular permutations. In future work we will implement topological crossover for TSP.

References

[1] Bäck T, Fogel DB, Michalewicz Z (2000) Evolutionary Computation 1: Basic Algorithms and Operators. IOP Publishing, Bristol, UKT.

[2] Bäck T, Fogel D and Michalewicz Z (1997) Handbook of evolutionary computation. IOP Publishing and Oxford University Press, New York

[3] Caprara A (1997) Sorting by reversals is difficult, Proc. of the 1st Ann. International Conference on Computational Molecular Biology (RECOMB 97), pages 75-83, ACM, New York.

[4] Deza M and Huang T (1998) Metrics on permutations, a survey, J. Combinatorics, Information and System Sciences 23, pages 173-185.

[5] Fox B R and McMahon M B (1991) Genetic Operators for Sequencing Problems In G J E Rawlins, editor, Foundations of Genetic Algorithms, pages 284-300. Morgan Kaufmann.

[6] Glover F and Kochenberger G (2002) Handbook of Metaheuristics. Kluwer Academic.

[7] Goldberg D E (1989) Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading, MA.

[8] Kececioglu J and Sankoff D (1995) Exact and approximate algorithms for sorting by reversals, with applications to genome rearrangement. Algorithmica, 13, pages 180-210.

[9] Moraglio A and Poli R (2004) Topological interpretation of crossover. Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2004), pages 1377-1388.

[10] Radcliffe N J (1992) Nonlinear genetic representations. In R. Manner and B. Manderick, editors, Proceedings of the 2 nd Conference on Parallel Problems Solving from Nature, pages 259-268. Morgan Kaufmann.

[11] Radcliffe N J (1994) The Algebra of Genetic Algorithms, Annals of Maths and Artificial Intelligence 10, pages 339-384.

[12] Solomon A, Sutcliffe P, Lister R (2003) Sorting Circular Permutations by Reversals. WADS 2003, pages 319-328

[13] Vergara J P C (1997) Sorting by Bounded Permutations, Virginia Polytechnic Institute, PhD thesis.