Matemática Contemporânea, Vol. 44, 1–10
http://doi.org/10.21711/231766362015/rmc4412

Sorting Separable Permutations by Restricted Multi-break Rearrangements

Luís Felipe I. Cunha **Luis Antonio B. Kowada**
Celina M. H. de Figueiredo

Abstract

A multi-break rearrangement generalizes most of genome rearrangements, such as block-interchanges, transpositions and reversals. A k-break cuts k adjacencies over a permutation, and forms k new adjacencies by joining the extremities according to an arbitrary matching. Block-interchange distance is a polynomial problem, but the transposition and the reversal distances are both NP-hard problems. A FPT algorithm is known for the multi-break distance between two permutations. We propose the *restricted multi-break rearrangement* (*rmb*), where a restricted k-break cuts k adjacencies but forms k new adjacencies according to a fixed matching. By considering permutations graphs we are able to formulate a better way to represent the orders of a permutations. Cographs are P_4-free graphs, a subclass of permutation graphs. The permutations that characterize cographs are the separable permutations, exactly the permutations which do not contain particular patterns that yield P_4's. By using their cotree representation, we give an algorithm to sort by *rmb* the separable permutations.

2000 AMS Subject Classification: 68Q17, 68P10 and 05A05.

Key Words and Phrases: Algorithms, Genome rearrangements, Sorting permutations, Cographs

Supported by CAPES, CNPq, and FAPERJ.

1 Introduction

Genome rearrangements ask for the minimum number of mutational events over a genome required to transform a genome into another one. These problems are classical in bioinformatics [FLR+09]. In a mathematical model we assume a genome as a permutation of integers and we aim at finding the minimum number of operations to transform a given permutation into the identity permutation, named *rearrangement distance*, where several approaches regarding forms of operations are considered. In the present work we propose the *restricted multi-break rearrangement* operation, which generalizes the well-known block-interchange, transposition and reversal operations, and is a restricted form of the multi-break operation [AP08].

2 Sorting by rmb and k-rmb

For our purposes, a gene is represented by a unique integer and a chromosome with n genes is a *permutation* $\pi = [\boldsymbol{\pi_0}\, \pi_1\, \pi_2\, \ldots\, \pi_n \boldsymbol{\pi_{n+1}}]$, where $\boldsymbol{\pi_0} = 0$ and $\boldsymbol{\pi_{n+1}} = n+1$. The remaining elements form a bijection from the set $\{1, 2, \ldots, n\}$ onto itself, and the operations we consider will never act on $\boldsymbol{\pi_0}$ nor $\boldsymbol{\pi_{n+1}}$.

Definition 2.1. *The* restricted multi-break $rmb(a, b; c_1 \leftrightarrow d_1; c_2 \leftrightarrow d_2; \ldots ; c_k \leftrightarrow d_k)$*, where* $1 \leq a \leq c_1 \leq d_1 \leq c_2 \leq d_2 \leq \ldots \leq c_k \leq d_k \leq b \leq n$*, in* π*, reverses the interval of* π *defined by positions* a *and* b*, except for the* non-reversible blocks *defined by the pairs* (c_i, d_i) *for* $1 \leq i \leq k$*, thereby transforming* π *into the permutation* $\pi.rmb$ *illustrated in Figure 1:*

$$[\boldsymbol{\pi_0}\pi_1 \cdots \pi_{a-1} \boxed{\pi_b \cdots \pi_{d_k+1} \underbrace{\boxed{\pi_{c_k} \cdots \pi_{d_k}} \pi_{c_k-1} \cdots \pi_{d_1+1} \boxed{\pi_{c_1} \cdots \pi_{d_1}}}_{k \text{ non-reversible blocks}} \pi_{c_1-1} \cdots \pi_a} \pi_{b+1} \cdots \pi_n \boldsymbol{\pi_{n+1}}],$$

Figure 1: Permutation obtained from a restricted multi-break operation.

A sequence of m restricted multi-breaks *sorts* a permutation π (or *is a sorting sequence* for π) if $\pi\, rmb_1\, rmb_2 \cdots\ rmb_m\ =\ \iota$, where each rmb_i is a restricted multi-break and $\iota = [\mathbf{0}\,1\,2 \ldots n\ \boldsymbol{n{+}1}]$ is the *identity permutation*. The *k-rmb distance*, denoted $d_{krmb}(\pi)$ is the length of a minimum sequence of restricted multi-breaks that sorts π, where each restricted multi-break has at most k non-reversible blocks. When $k = n$ we call the *rmb distance* of π, denoted by $d_{rmb}(\pi)$, by the length of a shortest sorting sequence of restricted multi-breaks for π.

A *rmb* generalizes a *transposition* $\tau(a, d{+}1, b{+}1) = rmb(a, b; a \leftrightarrow d; d{+}1 \leftrightarrow b)$, a *block-interchange* $\beta(a, d_1, c_2{+}1, b)\ =\ rmb(a, b; a \leftrightarrow d_1; d_1{+}1 \leftrightarrow c_2; c_2{+}1 \leftrightarrow b)$, and a *reversal* $\rho(a, b) = rmb(a, b)$.

An *adjacency* (resp. a *reverse adjacency*) in a permutation π, is a pair (π_i, π_{i+1}) for $0 \leq i \leq n$ such that $\pi_{i+1} = \pi_i + 1$ (resp. $\pi_{i+1} = \pi_i - 1$). If it is neither an adjacency nor a reverse adjacency, i.e. $|\pi_i - \pi_{i+1}| \neq 1$, then (π_i, π_{i+1}) is called a *breakpoint*, and $b(\pi)$ is the number of breakpoints of π. The *reduced permutation*, denoted $gl(\pi)$, is obtained by replacing all consecutive adjacencies by a single element, so we have that $b(gl(\pi)) \leq b(\pi)$ and $d_{krmb}(\pi) \geq d_{krmb}(gl(\pi))$, but $d_{rmb}(\pi) = d_{rmb}(gl(\pi))$. Note that this equality also holds on the block-interchange and the transposition distance problems, but it does not hold on the reversal distance. Regarding k-rmb distance of a given permutation, we obtain the following bound.

Theorem 2.2. *The k-rmb distance of* π *satisfies* $d_{krmb}(\pi) \geq \frac{b(\pi)}{2(k+1)}$.

Proof. By applying a *rmb* in π with k non-reversible blocks, the number of breakpoints of $\pi \cdot rmb$ is $b(\pi \cdot rmb) \geq b(\pi) - (2 + 2k)$, since the best case of a $rmb(a, b; c_1 \leftrightarrow d_1; c_2 \leftrightarrow d_2; \ldots; c_k \leftrightarrow d_k)$ we are able to eliminate a pair of breakpoints in the external interval $(\pi_{a-1}\pi_a)$, $(\pi_b\pi_{b+1})$ and each pair of breakpoints of a non-reversible block $(\pi_{c_i-1}\pi_{c_i})$, $(\pi_{d_i}\pi_{d_i+1})$ for $i = 1, \ldots, k$. Thereby we can conclude the proof by induction on the number of breakpoints of π. ■

3 Sorting separable permutations by *rmb*'s

A graph is a *permutation graph* [Gol04] if and only if it has an intersection model consisting of straight line segments (one per vertex) between two parallel lines. Every permutation is associated to a permutation graph by the intersection model above. $PG(\pi) = (V, E)$ is the permutation graph of $\pi = [\mathbf{0}\pi_1 \cdots \pi_n \boldsymbol{n+1}]$, such that $V = \{1, 2, \cdots, n\}$, and $E = \{ij \mid i < j, \pi_i > \pi_j\}$. Note that the complementary graph $\overline{PG(\pi)} = PG(\sigma)$, where $\sigma = \pi\rho(1, n)$ is the reversal of π.

A *cograph* is a P_4-free graph, i.e. a particular permutation graph that can be constructed from isolated vertices by disjoint union and join operations. A permutation σ is a *pattern* of the permutation π if π contains a subsequence whose relative ordering matches σ. A permutation is such that the corresponding permutation graph is a cograph if and only if, it does not contain neither [2 4 1 3] nor [3 1 4 2] as a pattern, and these permutations are called *separable permutations* [BBL98]. The number of separable permutations of length n is given by the $(n-1)$-th Schröder number [Wes95].

Given a cograph we can represent disjoint union and join operations by its associated tree, named *cotree*. We construct a cotree by the following strategy: Given a cograph G, create a *leaf node* for each vertex of G. If G is disconnected, create the edges between the root of the cotree and other *internal nodes* (which are the connected components of G), and the root is labeled by 0, corresponding to the union of the connected components of G. If G is connected, consider its complement graph $\overline{G}$ (since $\overline{G}$ is a cograph and hence if G is connected, $\overline{G}$ is disconnected), create edges between the root of the cotree and other internal nodes (which are the connected components of $\overline{G}$), and the root is labeled by 1, corresponding to the join of the connected components of $\overline{G}$. After that we apply the above strategy to the new internal nodes until we get to the leaves of the cotree. We have that: the internal nodes on each root-to-leaf path on the cotree alternate labels 0 and 1; each internal node has at least two

children (except for the trivial graph); an internal node labeled by 0 that is parent of p leaves represents $p-1$ consecutive permutation adjacencies. Thus the reduced permutation is obtained by replacing all vertices that represent the consecutive permutation adjacencies by a single vertex and by eliminating their parent labeled as 0 if its contains only these leaves as children; when a reversal is applied, the cotree of $\overline{G}$ is obtained from the cotree of G by exchanging the 0/1 labels of the internal nodes and reflecting the tree.

Given a separable permutation and its corresponding cotree, we present Algorithm 1, which returns an upper bound for the *rmb* distance. A *rmb* operation can be viewed as a reversal applied over a reduced permutation, where each block of consecutive permutation adjacencies is considered a non-reversible block on a *rmb* operation.

Algorithm 1 Sorting permutation π with cotree $T(\pi)$ and height $h(T(\pi))$.

$\pi \leftarrow gl(\pi)$; $d \leftarrow 0$; $T(\pi) =$ cotree of $PG(\pi)$.
while $h(T(\pi)) \geq 1$ **do**
  Apply reversal ρ in the leaves of $T(\pi)$ (corresponding to a *rmb* in the permutation before reducing); $d \leftarrow d+1$; $\pi \leftarrow gl(\pi.\rho)$.
end while
return d.

Figure 2 illustrates an example of how to sort the permutation $\pi =$ [**0** 3 2 1 10 11 6 5 4 7 8 9 **12**]. Algorithm 1 first obtains the reduced permutation. Now, Algorithm 1 performs a while loop: a reversal is applied, followed by a reduction where every 0 node that only contains leaves as children is removed. Given a cotree of $gl(\pi)$, for every internal node u, and $N(u)$ the set of its neighbors, we define the *layer of* u, named l_u by the following iterative construction:

1. $i = 1$;
2. if layer of each node is given, then stop. Otherwise at step i obtain $T(\pi)$;
3. give layer i to each node that has only leaves as children;

4. return to Item 2 with respect to $T(\pi)$ relabeled without nodes with layers already given.

The layer l_u, corresponds to the l_u-th step the while loop is performed when u is eliminated by reduction in Algorithm 1. Thereby, we define the *layer of* π, named $l(\pi)$, by the maximum layer among the set I of all internal nodes of the cotree of π: $l(\pi) = \max_{u \in I} l_u$. Note that for a given permutation π, we can determine $l(\pi)$ in polynomial time in the cotree.

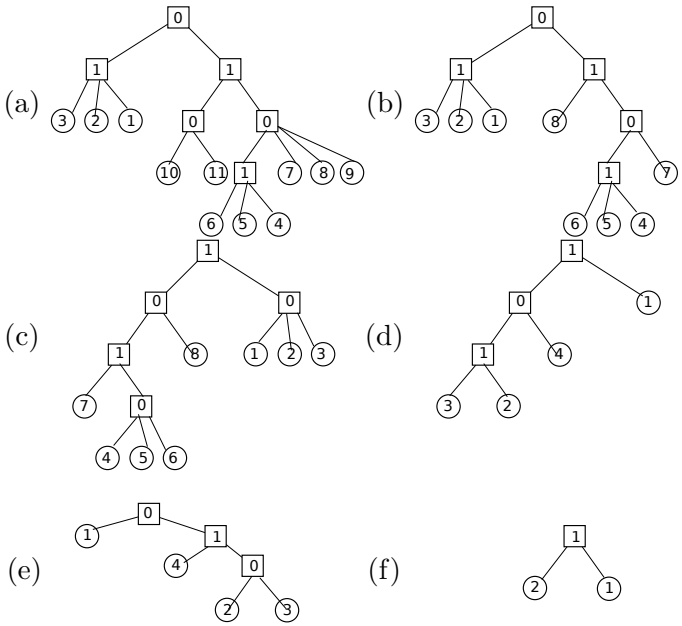

Figure 2: Steps to sort $\pi = [\mathbf{0}\,3\,2\,1\,10\,11\,6\,5\,4\,7\,8\,9\,\mathbf{12}]$. Each reversal applied over a reduced permutation corresponds to a *rmb* applied over a original permutation.

Lemma 3.1. *Given a separable permutation π, Algorithm 1 returns $l(\pi)$ operations to sort π.*

Theorem 3.2. *Given a separable permutation π, we have $d_{rmb}(\pi) \leq l(\pi)$.*

4 Improving the upper bound by *rmb*'s

In this work we have developed an algorithm to sort the class of separable permutations with respect to *rmb*, which generalizes three well-known rearrangement operations. Some questions arise: Is it possible to extend in some sense our strategy for any permutation? Is it possible to develop another strategy to achieve the exact distance for the separable permutations?

Regarding the former question, a possible strategy to guarantee an upper bound by *rmb* for a non separable permutation could be transforming it into a separable permutation eliminating every P_4 by a *rmb* operation, and after that applying Algorithm 1 to sort the new permutation. But, as Figure 3 illustrates, a *rmb* to eliminate a P_4 may create a new P_4. Although it happens in general, we can ask for classes of permutations that do not create a new P_4. Furthermore, for which permutations the strategy to transform them into separable permutations and afterwards apply Algorithm 1 leads us to an optimal sorting sequence?

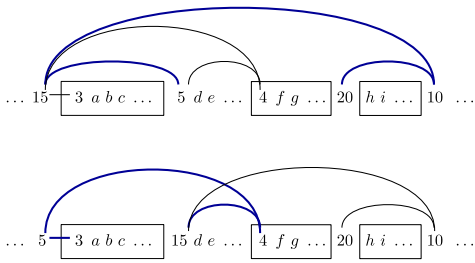

Figure 3: Eliminating the P_4 corresponding to 15 5 20 10, by the *rmb* that inverts between 15 and 5 keeping the non-reversible block elements between 3 and the one just before 5, we create the P_4 corresponding to 5 3 15 4.

Regarding the later question, we consider another graph, let $Ad(\pi) = (V, A)$ denote the *adjacency directed graph* of π, defined by: $V = \{\pi_1, \cdots, \pi_n\}$, and the directed set of arcs $A = \{(i-1, i) : i = 2, \cdots, n\}$. Note that for

any permutation of length n the adjacency graph is a path of length n. Keeping the vertices in the same order as the elements corresponding to the permutation, we can partition the arc set into four kinds of arcs: i) *left straights*, L_s, an arc (π_i, π_{i-1}) whenever $\pi_{i-1} = \pi_i + 1$, ii) *left curves*, L_c, an arc (π_i, π_j) whenever $j < i - 1$ and $\pi_j = \pi_i + 1$, iii) *right straights*, R_s, an arc (π_{i-1}, π_i) whenever $\pi_i = \pi_{i-1} + 1$ and iv) *right curves*, R_c, an arc (π_i, π_j) whenever $j > i + 1$ and $\pi_j = \pi_i + 1$.

The identity permutation is the unique permutation where all arcs are right straights, therefore our goal is to transform any path configuration into the path where all arcs are right straights.

Proposition 4.1. *If any π such that $R_c = \emptyset$, then $d_{rmb}(\pi) \leq 1$.*

Proof. Since π has no right curve arc, we can succeed a *rmb* by transforming each consecutive left straight arcs into consecutive right straight arcs by inverting such interval; each consecutive right straights is either not considered in the *rmb* if those elements are the firsts or the lasts of the permutation, or is considered as a non-reversible block; and each left curve arc (a, b) we transform into a right straight by inverting from b until a considering the non-reversible blocks the corresponding intervals of right straights between a and b. Note that such elements between b and a yield consecutive right straight arcs, therefore such *rmb* does not change the arcs that are already right straights. ■

Proposition 4.1 suggests us to transform a permutation in each step into another one by reducing the maximum number of right arcs. So, one more question arises: how can we maximize the number of right arcs considering shuffling arcs of different kinds?

References

[AP08] Max A. Alekseyev and Pavel A. Pevzner, *Multi-break rearrangements and chromosomal evolution*, Theoret. Comput. Sci. **395** (2008), no. 2-3, 193–202. MR 2424507

[BBL98] Prosenjit Bose, Jonathan F. Buss, and Anna Lubiw, *Pattern matching for permutations*, Inform. Process. Lett. **65** (1998), no. 5, 277–283. MR 1620935

[FLR+09] Guillaume Fertin, Anthony Labarre, Irena Rusu, Éric Tannier, and Stéphane Vialette, *Combinatorics of genome rearrangements*, Computational Molecular Biology, MIT Press, Cambridge, MA, 2009. MR 2518996

[Gol04] Martin Charles Golumbic, *Algorithmic graph theory and perfect graphs*, second ed., Annals of Discrete Mathematics, vol. 57, Elsevier Science B.V., Amsterdam, 2004, With a foreword by Claude Berge. MR 2063679

[Wes95] Julian West, *Generating trees and the Catalan and Schröder numbers*, Discrete Math. **146** (1995), no. 1-3, 247–262. MR 1360119

Luís Felipe I. Cunha
Universidade Federal do Rio de Janeiro
Brasil
lfignacio@cos.ufrj.br

Celina M. H. de Figueiredo
Universidade Federal do Rio de Janeiro
Brasil
celina@cos.ufrj.br

Luis Antonio B. Kowada
Universidade Federal Fluminense
Brasil
luis@vm.uff.br