Parking Functions, Labeled Trees and DCJ Sorting Scenarios

Aïda Ouangraoua\textsuperscript{1,2} and Anne Bergeron\textsuperscript{2}

\textsuperscript{1} Department of Mathematics, Simon Fraser University, Burnaby (BC), Canada
\texttt{aouangra@sfu.ca}
\textsuperscript{2} Lacim, Université du Québec à Montréal, Montréal (QC), Canada
\texttt{bergeron.anne@uqam.ca}

Abstract. In genome rearrangement theory, one of the elusive questions raised in recent years is the enumeration of rearrangement scenarios between two genomes. This problem is related to the uniform generation of rearrangement scenarios, and the derivation of tests of statistical significance of the properties of these scenarios. Here we give an exact formula for the number of double-cut-and-join (DCJ) rearrangement scenarios of co-tailed genomes. We also construct effective bijections between the set of scenarios that sort a cycle and well studied combinatorial objects such as parking functions and labeled trees.

1 Introduction

Sorting genomes can be succinctly described as finding sequences of rearrangement operations that transform a genome into another. The allowed rearrangement operations are fixed, and the sequences of operations, called sorting scenarios, are ideally of minimal length. Given two genomes, the number of different sorting scenarios between them is typically huge – we mean HUGE – and very few analytical tools are available to explore these sets.

In this paper, we give the first exact results on the enumeration and representation of sorting scenarios in terms of well-known combinatorial objects. We prove that sorting scenarios using DCJ operations on co-tailed genomes can be represented by parking functions and labeled trees. This surprising connection yields immediate results on the uniform generation of scenarios \textsuperscript{11,17}, promises tools for sampling processes \textsuperscript{6,12} and the development of statistical significant tests \textsuperscript{10,15}, and offers a wealth of alternate representations to explore the properties of rearrangement scenarios, such as commutation \textsuperscript{4,21}, structure conservation \textsuperscript{3,17}, breakpoint reuse \textsuperscript{13,16} or cycle length \textsuperscript{22}.

This research was initiated while we were trying to understand commuting operations in a general context. In the case of genomes consisting of single chromosomes, rearrangement operations are often modeled as inversions, which can be represented by intervals of the set \{1, 2, \ldots, n\}. Commutation properties are described by using overlap relations on the corresponding sets, and a major tool to understand sorting scenarios are overlap graphs, whose vertices represent single rearrangement operations, and whose edges model the interactions between

F.D. Ciccarelli and I. Miklós (Eds.): RECOMB-CG 2009, LNBI 5817, pp. 24–35, 2009.
© Springer-Verlag Berlin Heidelberg 2009
the operations. Unfortunately, overlap graphs do not upgrade easily to genomes with multiple chromosomes, see, for example, [13], where a generalization is given for a restricted set of operations.

We got significant insights when we switched our focus from single rearrangement operations to complete sorting scenarios. This apparently more complex formulation offers the possibility to capture complete scenarios of length $d$ as simple combinatorial objects, such as sequences of integer of length $d$, or trees with $d$ vertices. It also gives alternate representations of sorting scenarios, using non-crossing partitions, that facilitate the study of commuting operations and structure conservation.

In Section 3 we first show that sorting a cycle in the adjacency graph of two genomes with DCJ rearrangement operations is equivalent to refining non-crossing partitions. This observation, together with a result by Richard Stanley [19], gives the existence of bijections between sorting scenarios of a cycle and parking functions or labeled trees. Parking functions and labeled trees are some of a number of combinatorial objects enumerated by the formula $(n + 1)^{n-1}$ [8,18,20]. We give explicit bijections with parking functions and labeled trees in Sections 4 and 5. We conclude in Section 6 with remarks on the usefulness of these representations, on the algorithmic complexity of switching between representations, and on generalizations to genomes that are not necessarily co-tailed.

2 Preliminaries

Genomes are compared by identifying homologous segments along their DNA sequences, called blocks. These blocks can be relatively small, such as gene coding sequences, or very large fragments of chromosomes. The order and orientation of the blocks may vary in different genomes. Here we assume that the two genomes contain the same set of blocks and consist of either circular chromosomes, or co-tailed linear chromosomes. For example, consider the following two genomes, each consisting of two linear chromosomes:

Genome $A$: $(a - f - b - e - d) (-c - g)$

Genome $B$: $(a \ b \ c) (d \ e \ f \ g)$

The set of tails of a linear chromosome $(x_1 \ldots x_m)$ is $\{x_1, -x_m\}$, and two genomes are co-tailed if the union of their sets of tails are the same. This is the case for genomes $A$ and $B$ above, since the the union of their sets of tails is $\{a, -c, d, -g\}$.

An adjacency in a genome is a sequence of two consecutive blocks. For example, in the above genomes, $(e - d)$ is an adjacency of genome $A$, and $(a \ b)$ is an adjacency of genome $B$. Since a whole chromosome can be flipped, we always have $(x \ y) = (\ -y \ -x\ )$.

The adjacency graph of two genomes $A$ and $B$ is a graph whose vertices are the adjacencies of $A$ and $B$, and such that for each block $y$ there is an edge between adjacency $(y \ z)$ in genome $A$ and $(y \ z')$ in genome $B$, and an edge between $(x \ y)$ in genome $A$, and $(x' \ y)$ in genome $B$. See, for example, Figure 1.