Abstract Genome rearrangements are mutations that change the gene content of a genome or the arrangement of the genes on a genome. Several years of research on genome rearrangements have established different algorithmic approaches for solving some fundamental problems in comparative genomics based on gene order information. This review summarizes the literature on genome rearrangement analysis along two lines of research. The first line considers rearrangement models that are particularly well suited for a theoretical analysis. These models use rearrangement operations that cut chromosomes into fragments and then join the fragments into new chromosomes. The second line works with rearrangement models that reflect several biologically motivated constraints, e.g., the constraint that gene clusters have to be preserved. In this chapter, the border between algorithmically "easy" and "hard" rearrangement problems is sketched and a brief review is given on the available software tools for genome rearrangement analysis.

Key words: gene order analysis, genome rearrangements, cut and join, gene cluster

Genome Rearrangement Analysis:
Cut and Join Genome Rearrangements and Gene Cluster Preserving Approaches

Tom Hartmann, Martin Middendorf, Matthias Bernt

1 Introduction

During evolution the gene content as well as the arrangement of the genes within the genomes of species have been modified by various kinds of mutations. A genome consists of a set of chromosomes and each chromosome contains a set of genes. A chromosome represents a linear or circular string of DNA where the genes are located. Each gene of a chromosome has an orientation depending on which of the two strands of the DNA it is located. Mutations that change the arrangement of the genes on the chromosome are called rearrangement mutations (or rearrangements). Examples are inversions that reverse the order and orientation of a subsequence of genes on a chromosome and transpositions which move a subsequence of genes from one chromosome to another location on the same or another chromosome. Other examples are fissions which split a chromosome into two chromosomes and fusions which merge two chromosomes to form a single chromosome. In addition to such rearrangement mutations which do not change the gene content of a genome, there exist also rearrangements that change the gene content of a genome. However, the latter type of rearrangements are not considered in this chapter.

Gene order analysis aims to explain the differences of the gene arrangement (and gene content) between the genomes of extant species and to reconstruct their phylogenetic evolution. The most common approach for gene order analysis is maximum parsimony. This approach tries to explain the differences within a set of genomes by the minimum possible number of changes. For example, on might ask for a shortest sequence of (allowed) rearrangement mutations that transforms one given genome into another given genome. This problem is called the sorting problem. The corresponding problem that asks

Swarm Intelligence and Complex Systems Group, Institute of Computer Science, University Leipzig, Germany, e-mail: {thartmann,middendorf,bernt}@informatik.uni-leipzig.de

only for the smallest number of such mutations is called distance problem. Such distance information between genomes of species can be used, e.g., to reconstruct the phylogenetic relationships between the species, e.g., [110].

The exploration of the combinatorial and computational problems that occur in gene order analysis started with the pioneering works of Sankoff and Blanchette [97] and Watterson et al. [111]. Early breakthroughs have been obtained with respect to the sorting problem where polynomial time algorithms have been developed for the problem of sorting with inversions [71] and for sorting with inversions plus fission and fusion [70]. However, many other related problems have been shown to be NP-hard which makes it unlikely that they can be solved optimally in polynomial time. Examples are the problem of sorting with inversions for unsigned genomes (i.e., the orientations of the genes are ignored) [44], the inversion median problem (i.e., the problem to find a parsimonious common ancestral gene order for more than two given gene orders) [45], and the problem of sorting by transpositions [42]. Hence, one focus of the research in gene order analysis is to develop heuristics and approximation algorithms. Examples of such algorithms are the frequently used `MGR` algorithm for solving the inversion median problem [36] and approximation algorithms for the sorting by transpositions problem [54].

This chapter gives an overview on the different gene rearrangement problems and reviews the corresponding literature. We also give a brief review of the available software tools and web services for rearrangement analysis (see, e.g., Table 1). The chapter is organized along two lines of research in gene order analysis.

The first line of research tries to find models of rearrangement mutations which are well suited for a theoretical analysis. These models are based on two operations: i) to cut a chromosome into fragments and ii) to rejoin such fragments (into new chromosomes). The differences between the models are the number of cut points of an operation and the different ways how the fragments can be rejoined. Thereby, most of the (classical) rearrangement operations are directly covered and a few others, e.g., transpositions, can be emulated by a combination of such cut and join operations. One focus is to sketch the border between easy problems, i.e., problems that are polynomial time solvable and difficult problems, i.e., problems that are NP-hard.

The second line of research tries to identify biologically motivated constraints on the applicability of gene rearrangement mutations. One particular important type of constraints are conserved gene clusters, i.e., subsets of the genes that are observed in close proximity on all given genomes. Such gene clusters might form due to functional constraints or evolutionary inertia. The idea is to consider such conserved gene clusters in the reconstructions, i.e., genome rearrangement that break gene clusters are forbidden. Even if a rearrangement problem might become or remain NP-hard with such additional restrictions, e.g., the preserving inversion sorting problem [60], the search space becomes more structured such that many problem instances can be solved efficiently [17, 25, 27]. It is worth to mention that in addition to the

preservation of gene clusters, other biologically motivated constraints have been introduced, e.g., positional constraints on gene adjacencies motivated on chromosome conformation information [102, 108].

This chapter is organized as follows. In the next Section 2 a formal background on gene arrangements and rearrangement mutations is given. In addition, the basic problems of gene order analysis are introduced. An overview on the literature on cut and join models of rearrangement mutations is presented in Section 3. An overview on gene order analysis with conserved gene clusters is given in Section 5. The chapter ends with a conclusion in Section 6.

2 Preliminaries

A formal background on gene arrangements, rearrangement mutations, and basic problems of gene order analysis is given in this section.

2.1 Gene Orders

A *genome* is a set of linear or circular DNA sequences which are called *chromosomes*. Genomes are called *unichromosomal* if they consist of only a single chromosome and *multichromosomal* otherwise. If all chromosomes of a genome are linear (resp. circular) then the genome is called *linear* (resp. *circular*) and *mixed* otherwise.

A *gene* is a unit of genetic information that encodes a specific function. It is assumed here that each gene has a unique identifier and that it can be identified whether a gene occurs in a genome or not. If a gene occurs in a genome it can be located and it corresponds to an oriented sequence of DNA. In this chapter we consider only the case that genes within a genome do not intersect. If not stated otherwise, it is assumed that a gene occurs at most once in a genome. The set of genes of a genome α is denoted by $\mathcal{G}(\alpha)$. A gene g can be represented by its *extremities*: the *tail* g^t and the *head* g^h, i.e., the 5' and 3' end of the gene. A linear (resp. circular) chromosome can be represented by the linear (resp. circular) sequence of the extremities of its genes.

Given a genome α, an *adjacency* is an unordered pair $\{p, q\}$ of extremities p and q, $p \neq q$, of genes of $\mathcal{G}(\alpha)$ that are adjacent on a chromosome of α. The endpoints of linear chromosomes are called *telomeres*. A telomere is denoted by the pair $\{p, \circ\}$, where p is the extremity that is closest to the telomere and therefore p is not adjacent to the extremity of another gene. The set of all telomeres of a genome α is denoted by $\mathcal{T}(\alpha)$. The set of extremities $\mathcal{E}(\alpha)$ of a genome α consists of all extremities of the genes, i.e., $\mathcal{E}(\alpha) = \{g^h, g^t : g \in \mathcal{G}(\alpha)\}$. A genome α can be characterized by the set of

adjacencies of its genes and its telomeres, for simplicity let α denotes this set. Furthermore, $\mathcal{I}(\alpha)$ denoted the set of all adjacencies (without the telomeres) of a genome, i.e., $\mathcal{I}(\alpha) = \alpha \backslash \mathcal{T}(\alpha)$. Note that $\mathcal{I}(\alpha)$ uniquely determines the set of telomeres. For a set of genes $\mathcal{G}$, the set of all possible genomes is denoted by $\mathbb{G}(\mathcal{G})$ (or simply $\mathbb{G}$ if the context is clear).

A genome α can be represented by an undirected graph – the *genome graph* – which contains the extremities as vertices and an edge connects two vertices x and y if and only if either $\{x, y\}$ is an adjacency in α or x and y are head and tail of the same gene. *Linear* and *circular* chromosomes correspond to linear and circular connected components in the genome graph. Figure 1 illustrates an example of a genome graph.

A *weighted genome graph wGG* is a genome graph that has a nonnegative integer weight for each edge that connects two extremities x and y if and only if either $\{x, y\}$ is an adjacency or one of x, y is a telomere. The weight $w(\alpha)$ of a genome α is the sum of the weights of all edges of $wGG(\alpha)$.

Fig. 1 Genome graph of the mixed multichromosomal genome $\alpha = \{\{a^t, \circ\}, \{a^h, b^h\}, \{b^t, \circ\}, \{c^t, d^h\}, \{d^t, c^h\}\}$ consisting of the linear chromosome $\mathcal{C}_1 = \{\{a^t, \circ\}, \{a^h, b^h\}, \{b^t, \circ\}\}$ and the circular chromosome $\mathcal{C}_2 = \{\{c^t, d^h\}, \{d^t, c^h\}\}$. The set of telomeres of α is $\mathcal{T}(\alpha) = \{\{a^t, \circ\}, \{b^t, \circ\}\}$. Genome α can also be represented by $S_\alpha = \{(a\ -b), (-d\ -c)^\circ\}$, where $^\circ$ marks the chromosome as circular.

An alternative representation of a genome α is a set of signed strings S_α, where each string lists the genes of one chromosome as if read from one telomere to the other (or in the case of circular chromosomes starting from an arbitrary gene), see Figure 1. If the head of a gene x appears before its tail the gene in the string gets a negative sign (represented by $-x$), otherwise a positive sign is assigned to the gene (a positive sign is omitted). The sign represents the (relative) strandedness of the genes. Both strings of the two reading directions of a linear chromosome and also all strings generated from different start points in circular chromosomes are considered to be equivalent. Note, that for some applications this assumption might not be useful, e.g. if parts chromosomes with known orientation. For unichromosomal genomes with n genes this representation is equivalent to a *signed permutation* of length n. Such a permutation π is denoted by $(\pi(1)\ \pi(2) \dots \pi(n))$, where $\pi(i)$ denotes the i-th element. The identity permutation $(1\ 2 \dots\ n)$ is denoted by ι. For differentiation the string corresponding to a circular chromosome is enclosed between $()^\circ$. In the following the two representations of unichromosomal genomes are used interchangeably, where π and σ denote genomes as permutations, and α and β denote genomes as sets of adjacencies and telom-

eres. The considered genomes are always assumed to consist of the same set of genes throughout this chapter.

Often in nature a subset of genes is found in close proximity in the genomes of many different species. Such a set of genes is called gene cluster [69]. A simple formal model for gene clusters are *common intervals* [72]. An *interval* I of a permutation π of length n is a subset of elements of π which form a consecutive segment, i.e., there exists a pair (i, j), with $1 \leq i \leq j \leq n$, such that $\{|\pi(x)| : i \leq x \leq j\} = I$. For a set of signed permutations Π a common interval is a subset of the elements of the permutations that is an interval in each permutation in Π. Singleton sets $\{i\}$ with $i \in [1 : n]$ as well as $\{1, \ldots, n\}$ are called *trivial common intervals*. The set of all common intervals of a set of permutations Π is denoted by $C(\Pi)$. A set of permutations Σ is *consistent* with the common intervals of a set of permutations Π if $C(\Pi) = C(\Pi \cup \Sigma)$.

2.2 Rearrangement Model

Different types of rearrangement mutations are of interest for rearrangement analysis. See Figure 2 for a selection of such rearrangement mutations. The set of possible rearrangement mutations that is of interest is called a rearrangement model.

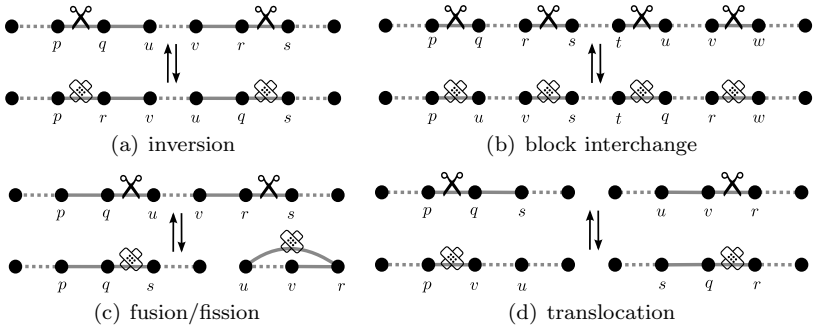


Fig. 2 Rearrangement events visualized on the genome graph. (a) An inversion reverses the order of the genes between two cut points (a cut is indicated by a scissor, a join is indicated by a plaster) and inverses the orientation of the affected genes. (b) A block interchange swaps two intervals. If the two intervals are adjacent, i.e., the middle cut points coincide, it is a transposition. (c) A fission cuts out an interval and creates a circular chromosome. A fusion inverts this operation. (d) A translocation swaps the ends of two chromosomes.

Along the lines of [21] a *rearrangement model* $\mathcal{M}$ is defined as a binary relation over the set of all possible genomes, i.e., $\mathcal{M} \subseteq \mathbb{G} \times \mathbb{G}$. A pair of genomes $\mathcal{R} = (\alpha, \alpha')$ is a *rearrangement event* in $\mathcal{M}$ if and only if $\mathcal{R} \in \mathcal{M}$.

The event $\mathcal{R}$ can be considered as a rearrangement that transforms α into α', we write $\alpha' = \alpha \circ \mathcal{R}$. Given two genomes α and α' and a (rearrangement) model $\mathcal{M}$, a *scenario* is a sequence $\mathcal{R}_1, \ldots, \mathcal{R}_k$ of rearrangement events in $\mathcal{M}$ with $\alpha' = \alpha \circ \mathcal{R}_1 \circ \ldots \circ \mathcal{R}_k$. The set of all possible scenarios for two given genomes α and α' is denoted by $\mathcal{S}_{\mathcal{M}}(\alpha, \alpha')$. The most famous rearrangement model for unichromosomal genomes is the inversion – or reversal – model ($\mathcal{INV}$). This model consists of all inversion operations and was studied first in [97, 111].

For each rearrangement model $\mathcal{M}$ a corresponding preserving variant, denoted by $\mathcal{M}^p$, can be defined as the set of rearrangement operations of $\mathcal{M}$ which do not destroy any common interval of the two given gene orders in any of its intermediate gene orders. Formally, an event $\rho = (\pi, \pi')$ of a model $\mathcal{M}$ is called *preserving* with respect to a set of permutations Π if $C(\Pi) = C(\Pi \cup \{\pi, \pi'\})$, i.e., π as well as π' are consistent with Π.

2.3 Genome Rearrangement Analysis

There exist several approaches to extract phylogenetic information from gene arrangement data, see [57] for a comprehensive overview. One example is the computation of rearrangement distances between genomes. Another example is the joint reconstruction of a phylogenetic tree for a set of genomes and the corresponding rearrangement events along its edges. In the following the central combinatorial problems that need to be solved for such approaches are formally defined.

Given a rearrangement model $\mathcal{M}$ and two genomes α and α' the *distance problem* for $\mathcal{M}$ is to find the minimum number $d_{\mathcal{M}}(\alpha, \alpha')$ of events from $\mathcal{M}$ to transform α into α'. That is, $d_{\mathcal{M}}(\alpha, \alpha') = \min_{S \in \mathcal{S}_{\mathcal{M}}(\alpha, \alpha')} |S|$. To find a corresponding reconstruction, i.e., a corresponding scenario that transforms one genome into the other is called *sorting problem*. The name is motivated by the fact that, by relabeling the elements of α and α', we can always assume that α' is the identity permutation ι. For a rearrangement model $\mathcal{M}$ that includes events which involve a constant number of cuts it holds that for every $\alpha \in \mathbb{G}$ the set $\{(\alpha, \alpha') \in \mathcal{M} : \alpha' \in \mathbb{G}\}$ has polynomial size. This are in particular all models involving a subset of the rearrangement operations that are shown in Figure 2. For such models it holds that the sorting problem can be solved in polynomial time if a polynomial time algorithm solves the distance problem, e.g., [21, 71]. Due to the close connection between the sorting problem and the distance problem this chapter is focused on the distance problem. However, the sorting problem can sometimes be solved faster than by testing all possible rearrangement events, e.g., the sorting problem for the $\mathcal{INV}$ model can be solved in sub-quadratic time [104]. It should be mentioned that the distance problem was studied for the first time in [97] for the case of the $\mathcal{INV}$ model. In [71] a polynomial time algorithm

for the sorting problem of signed permutations was given for the $\mathcal{INV}$ model. Examples of software tools which can solve this problem are `SORT`[2], `UniMoG`, and `baobabLUNA`. In addition, the software tool `revDis` can be used to solve the distance problem under $\mathcal{INV}$. See Table 1 for a summary.

Two central problems of gene order analysis for more than two genomes are the small and the large parsimony problem for gene orders. The *small parsimony problem* for a rearrangement model $\mathcal{M}$ and a given binary tree T where every leaf is associated to a genome asks for one ancestral genome α_i for every inner vertex i of T such that the sum of the distances between the two genomes of an edge of the tree is minimized, i.e., $\sum_{(u,v)\in E(T)} d_{\mathcal{M}}(\alpha_u, \alpha_v)$ has to be minimized, where $E(T)$ is the set of edges of T. For the *large parsimony problem* (also known as multiple genome rearrangement problem) the tree itself is sought in addition to the ancestral genomes at its inner vertices. The small and the large parsimony problem for gene orders were both studied, e.g., in [98].

A special case of the parsimony problems for rearrangement model $\mathcal{M}$ is the *median problem* which asks for a gene order α that minimizes the sum of the distances to given genomes $\alpha_1, \ldots, \alpha_k$, i.e., to find a α with $\min_{\alpha\in\mathbb{G}} \sum_{i=1}^{k} d_{\mathcal{M}}(\alpha, \alpha_i)$. Mostly, the case of $k = 3$ is considered. The solution of this case is often used in algorithms for solving larger multiple genome rearrangement problems [36, 86, 115]. With a few notable exceptions [56, 88, 105] the small parsimony problem has been proven to be NP-hard for most rearrangement models. For example, the median problem with $k = 3$ is NP-hard for inversions [44] and for transpositions [12]. However, the software tools `median` and `GRAPPA` provide solver for both of these problems, see Table 1.

Genome rearrangement problems that are defined for preserving rearrangement models are called *preserving genome rearrangement problems*. For a discussion of further rearrangement models and problems see [59]. In the following sections several basic algorithms for rearrangement models with and without the preserving property are studied.

3 Cut and Join Genome Rearrangement Models

This section reviews results for unconstrained rearrangement analysis, i.e., without additional constraints, e.g., to preserve gene cluster. The focus is on the so-called *cut and/or join* models. In particular, these are the single-cut or join model, the single-cut and join model, the double-cut and join model, and the multi-cut and join model. Since many of the results are based on particular graph structures these are introduced first in the following subsection.

Table 1 List of resources and web services which are thematically related; sorting problem (SP), distance problem (DP), median problem (MP), small parsimony problem (SPP), large parsimony problem (LPP), transpositions ($\mathcal{TR}$), inverse transpositions ($\mathcal{ITR}$), model including inversions, translocations, fusions, and fissions ($\mathcal{HP}$), duplications and deletions ($\mathcal{INDEL}$), block-interchanges ($\mathcal{BI}$), restricted $\mathcal{DCJ}$ events ($\mathcal{RDCJ}$), tandem-duplications ($\mathcal{TD}$), duplications ($\mathcal{D}$), weighted inversions ($\mathcal{WINV}$).

Resource	Reference	Features/Note
Web services		
CEGeD	[22, 23]	DP under $\mathcal{INV}$, $\mathcal{TR}$, and $\mathcal{DCJ}$
CREx	[29, 30]	SP under $\mathcal{INV}^p \cup \mathcal{TR}^p \cup \mathcal{ITR}^p$ and TDRL
GEvolutionS	[81]	simulates scenarios under $\mathcal{TR} \cup \mathcal{DCJ} \cup \mathcal{HP}$
GRIMM[a]	[106, 107]	SP under $\mathcal{HP}$ and $\mathcal{INV}$
MGR[a]	[13, 36]	LPP under $\mathcal{INV}$ and $\mathcal{HP}$
MGRA	[4, 5]	SPP and LPP under $\mathcal{DCJ}$
MLGO	[74, 75]	SPP and LPP under $\mathcal{DCJ} \cup \mathcal{INDEL}$
Roci	[101, 112]	SPP including $\mathcal{INV} \cup \mathcal{TR} \cup \mathcal{ITR}$ while preserving conserved intervals
SORT[2]	[76, 77]	SP under $\mathcal{INV}$, $\mathcal{BI}$, $\mathcal{HP} \cup \mathcal{BI}$, and $\mathcal{INV} \cup \mathcal{BI}$; infers phylogenetic trees based on distances
SBBI[a]	[49, 85]	SP under $\mathcal{BI}$
UniMoG[a]	[73]	DP and SP under $\mathcal{RDCJ}$, $\mathcal{INV}$, $\mathcal{TR}$, and $\mathcal{HP}$
Software for download		
baobabLUNA	[38, 39]	DP and SP under $\mathcal{INV}$; builds breakpoint graphs
dcjdDist	[10, 11]	phylogenetic reconstruction under $\mathcal{INV} \cup \mathcal{BI} \cup \mathcal{TD} \cup \mathcal{D}$
DING	[33, 32]	DP under $\mathcal{DCJ} \cup \mathcal{INDEL}$
EMRAE	[116, 117]	LPP for synteny blocks under $\mathcal{INV} \cup \mathcal{HP} \cup \mathcal{TR}$
GENESIS	[67, 68]	SP under $\mathcal{WINV}$, $\mathcal{WINV} \cup \mathcal{TR}$, $\mathcal{WINV} \cup \mathcal{HP}$, and $\mathcal{WINV} \cup \mathcal{HP} \cup \mathcal{TR}$
GRAPPA	[9, 86]	LPP under $\mathcal{INV}$
GREDU	[99, 100]	DP under $\mathcal{DCJ} \cup \mathcal{INDEL}$
Mauve	[50, 51]	multiple alignment tool; DP of synteny blocks under $\mathcal{DCJ}$
median	[14, 15][b]	MP under $\mathcal{INV}$, $\mathcal{TR}$, and $\mathcal{WINV} \cup \mathcal{TR}$
minswrt	[14, 15][b]	DP under $\mathcal{WINV} \cup \mathcal{TR}$
MSOAR	[64, 65]	calculates orthologous genes of genomic sequences involving genome rearrangements ($\mathcal{HP} \cup \mathcal{INDEL}$)
PATHGROUPS	[118, 119]	SPP under $\mathcal{DCJ}$
phylo	[14, 15]	phylogenetic reconstruction under $\mathcal{WINV} \cup \mathcal{TR}$
revDis	[14, 15][b]	DP under $\mathcal{INV}$
SPP-DJC	[53]	DP under $\mathcal{DCJ} \cup \mathcal{INDEL}$
weightedbb	[14, 15][b]	branch and bound algorithm for DP under $\mathcal{WINV} \cup \mathcal{TR}$

[a] Also available for download and local use.
[b] And included references.

3.1 Genome Rearrangement Graphs

The *breakpoint graph* for signed genomes is a key element of efficient algorithms for the calculation of the distance between genomes under the $\mathcal{INV}$ model [71]. For a source genome α and a target genome α' the breakpoint graph $BP(\alpha, \alpha') = G(V, E_b \cup E_g)$ is defined as follows (see also Figure 3). The vertices correspond to the extremities of the genes, i.e., $V = \mathcal{E}(\alpha)$. *Black edges* connect adjacent extremities of the source genome, i.e., $E_b = \alpha$ and *gray edges* connect adjacent extremities of the target genome, i.e., $E_g = \alpha'$.

For linear genomes two auxiliary vertices are added that are assumed to be adjacent to the first, respectively, the last extremity of the two genomes. Since each vertex has degree two the breakpoint graph can be partitioned into cycles, each of them with alternating black and gray edges. If both genomes are equal the breakpoint graph consists of a partition into color-alternating cycles of length two. Such cycles are called *one-cycles*. An extension of the breakpoint graph for more than two genomes has been introduced in [45] to solve the median problem for $\mathcal{INV}$. The software tool `baobabLUNA` provides a function which builds the breakpoint graph for two given genomes, see Table 1.

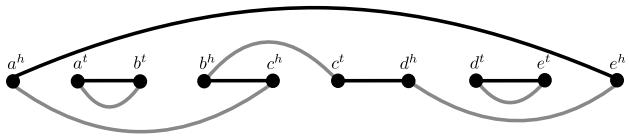


Fig. 3 Breakpoint graph for the circular genomes $S_\alpha = \{(-a\ b\ -c\ -d\ e)^\circ\}$ and $S_{\alpha'} = \{(a\ -c\ -b)^\circ, (-d\ e)^\circ\}$. Adjacencies of α (resp. α') are represented by black (resp. gray) edges. Note that the edges of $BP(\alpha, \alpha')$ form a partition into alternating cycles and adjacencies that are common to both genomes form one-cycles.

The *adjacency graph* $AG(\alpha, \alpha')$ is the line graph of the breakpoint graph of two genomes α and α' [23], see Figure 4. Hence, the vertices of the adjacency graph are the adjacencies and telomeres of both genomes and edges connect two adjacencies if and only if they share one extremity, i.e., the edge multi set is given by $\{\{u, v\} : u \in \alpha, v \in \alpha', u \cap v \neq \emptyset\}$. As for the breakpoint graph, shared adjacencies form cycles of the length two. But in contrast to the breakpoint graph, these cycles consists of color alternating vertices instead of color alternating edges. Since this is only a cosmetic difference, both cycles of length two are referred as one-cycles. The degree of a vertex is one if it is a telomere and two otherwise. Since the adjacency graph is bipartite every cycle has an even length. Hence, the adjacency graph consists of cycles, *even paths* which connect telomeres of the same genome, and *odd paths* which connect telomeres of different genomes. An even path which starts and ends at an adjacency that belongs to the same genome α is called α*-path*.

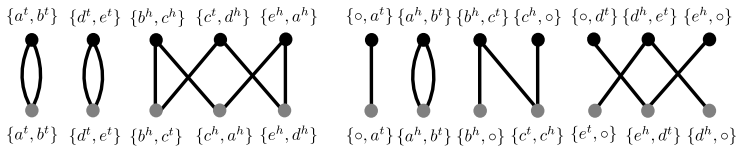


Fig. 4 (a) Adjacency graph for the circular genomes of Figure 3. The adjacency graph contains three cycles. To visualize that $AG(\alpha, \alpha')$ is the line graph of $BP(\alpha, \alpha')$ gray (resp. black) vertices of $AG(\alpha, \alpha')$ correspond to gray (resp. black) edges of $BP(\alpha, \alpha')$ and the edges of $AG(\alpha, \alpha')$ correspond to the vertices of $BP(\alpha, \alpha')$. (b) Adjacency graph for mixed genomes $S_\beta = \{(a\ b\ c), (d\ e)\}$ and $S_{\beta'} = \{(-b\ -a), (c)^\circ, (e\ d)\}$. $AG(\beta, \beta')$ consists of one cycle, two odd paths, and two even paths.

The breakpoint graph and the adjacency graph can both be constructed in linear time and require linear space [23]. The adjacency graph has the potential advantage that the genes of the two genomes are separated whereas they are tangled in the breakpoint graph which complicates the interpretation [63].

3.2 Single-Cut or Join Model

The *single-cut or join* model $\mathcal{SCOJ}$ [56] includes two types of rearrangements (see Figure 5(a)): A *cut* breaks an adjacency which creates two telomeres and a *join* forms a new adjacency by connecting two telomeres. Formally, a pair of genomes (α, α') is in $\mathcal{SCOJ}$ if and only if there exist different extremities p and q such that either $\alpha \cup \{\{p, \circ\}, \{\circ, q\}\} = \alpha' \cup \{p, q\}$ (cut) or $\alpha \cup \{p, q\} = \alpha' \cup \{\{p, \circ\}, \{\circ, q\}\}$ (join) holds. The single-cut or join model is a simplistic approximation of genome rearrangement evolution since all genome rearrangements can be represented by compositions of cuts and joins. However, this model has the advantage to allow for an efficient solution of some core problems of genome rearrangement analysis as explained in the following. More realistic models are described in the next sections.

Polynomial Time Solvable Problems

For the $\mathcal{SCOJ}$ model efficient algorithms are known for the distance problem, the sorting problem, the median problem, and the small parsimony problem [56].

Under $\mathcal{SCOJ}$ a parsimonious scenario for two genomes α and α' is obtained by cutting all adjacencies which are in α but not in α' and joining all extremities which are in α' but not in α. Hence, $d_{\mathcal{SCOJ}}(\alpha, \alpha') = |\mathcal{I}(\alpha) \triangle \mathcal{I}(\alpha')|$ where $\triangle$ denotes the symmetric difference, i.e., for two sets X and Y it holds $X \triangle Y := (X \setminus Y) \cup (Y \setminus X)$.

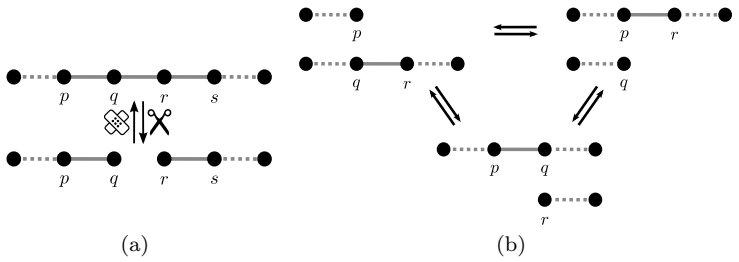


Fig. 5 Genome graphs representing rearrangement events. Dots indicate the existence of additional edges. (a) Events of the $\mathcal{SCOJ}$ model: cut event and join event. (b) Cut-and-join event of the $\mathcal{SCAJ}$ model. Together, both figures display the events of the $\mathcal{SCAJ}$ model.

This also solves the sorting problem under $\mathcal{SCOJ}$ in linear time. For details see [55]. The distance $d_{\mathcal{SCOJ}}(\alpha, \alpha')$ can also be interpreted in terms of the adjacency graph $AG(\alpha, \alpha')$. It holds that $d_{\mathcal{SCOJ}}(\alpha, \alpha') = 2N - 2C_2 - P$, where N is the number of genes, C_2 is the number of one-cycles, and P is the number of paths in $AG(\alpha, \alpha')$. This is because $|\mathcal{I}(\alpha)| = N - t_\alpha/2$, where t_α is the number of telomeres of α (analogously this holds for α'), $|\mathcal{I}(\alpha) \cap \mathcal{I}(\alpha')| = C_2$ and the fact that a path always connects two telomeres.

The median problem under $\mathcal{SCOJ}$ can also be solved in linear time [56]. The corresponding algorithm includes each adjacency into the median if it is an adjacency in the majority of the given genomes. That is, for input genomes $\alpha_1, \ldots, \alpha_k$ the median α consists of the adjacencies $\{p, q\} \in \bigcup_{i=1}^{k} \mathcal{I}(\alpha_i)$ with $|\{i \in [1 : k] : \{p, q\} \in \mathcal{I}(\alpha_i)\}| \geq \frac{k}{2}$. Note that the median for $\mathcal{SCOJ}$ is unique for an odd number of genomes whereas for an even number of genomes the inclusion or exclusion of adjacencies that are present in half of the genomes gives a parsimonious solution. For example, for $\alpha_1 = \{\{a^t, \circ\}, \{a^h, b^t\}, \{b^h, c^t\}, \{c^h, \circ\}\}$, $\alpha_2 = \{\{c^t, \circ\}, \{c^h, a^t\}, \{a^h, b^t\}, \{b^h, \circ\}\}$, and $\alpha_3 = \{\{a^t, \circ\}, \{a^h, c^t\}, \{b^h, c^h\}, \{b^t, \circ\}\}$ the median is $\mathcal{I}(\alpha) = \{\{a^h, b^t\}\}$. Since $\mathcal{T}(\alpha)$ is uniquely determined by $\mathcal{I}(\alpha)$, it follows that α is exactly the set $\{\{a^t, \circ\}, \{a^h, b^t\}, \{b^h, \circ\}, \{c^t, \circ\}, \{c^h, \circ\}\}$ with $\sum_{i=1}^{3} d_{\mathcal{SCOJ}}(\alpha, \alpha_i) = 1 + 1 + 3 = 5$. For more details and variants of the problem see [56].

The small parsimony problem can also be solved in polynomial time [56]. The idea is to encode the presence/absence of the adjacencies as binary characters. The ancestral adjacencies can be reconstructed with a variant of Fitch's algorithm [61] which resolves conflicting adjacencies.

NP-Hard Problems

The large parsimony problem for $\mathcal{SCOJ}$ is NP-hard [56]. This result was proven by reduction from the NP-hard Steiner tree problem in $\{1, 0\}^N$ where

binary characters are used to encode the presence or the absence of an adjacency [62].

Note that the $\mathcal{SCOJ}$ distance is effectively identical to the breakpoint distance [56], i.e., the number of adjacencies that are not common to both genomes. Hence, the polynomial time result for the breakpoint median problem for multichromosomal genomes [105] is consistent with the linear time result for the median problem under $\mathcal{SCOJ}$. The breakpoint median problem is NP-hard for unichromosomal genomes [92].

3.3 Single-Cut and Join Model

The *single-cut and join* model [21], denoted by $\mathcal{SCAJ}$, is an extension of the $\mathcal{SCOJ}$ model, i.e., $\mathcal{SCAJ} \supset \mathcal{SCOJ}$. Additionally to the cut and join operations of the $\mathcal{SCOJ}$ model the $\mathcal{SCAJ}$ model allows for a *cut-and-join* operation which cuts an adjacency and connects two (potentially different) telomeres. A pair of genomes (α, α') is an event in $\mathcal{SCAJ}$ if and only if $(\alpha, \alpha') \in \mathcal{SCOJ}$ or there exist different extremities p, q, and r such that $\alpha \cup \{\{p, r\}, \{q, \circ\}\} = \alpha' \cup \{\{p, q\}, \{r, \circ\}\}$ (cut-and-join).

The cut-and-join event realizes the following new rearrangement events (see Figure 5(b)): i) a linear chromosome can be transformed into a circular chromosome and a linear chromosome (fission), ii) a starting sequence (prefix) or an ending sequence (suffix) of a linear chromosome is either inverted (affix inversion) or joined to another linear chromosome, and iii) a circular chromosome can be cut and joined with a linear chromosome yielding a linear chromosome (fusion).

Polynomial Time Solvable Problems

The sorting problem and the distance problem under $\mathcal{SCAJ}$ can be solved in linear time [21]. The distance of two genomes α and α' under $\mathcal{SCAJ}$ can be computed by $d_{\mathcal{SCAJ}}(\alpha, \alpha') = N - I/2 - C_2 + C_{\geq 3}$, where N is the number of genes, C_2 is the number of one-cycles, $C_{\geq 3}$ is the number of cycles of length at least three, and I is the number of odd paths in the adjacency graph $AG(\alpha, \alpha')$. The sorting algorithm as given in [21] is sketched in the following. For two genomes α and α' with $\alpha \neq \alpha'$ it holds that $AG(\alpha, \alpha')$ contains at least one path of length greater than one or a cycle of length greater than two (since otherwise both genomes would be equal). For each path and cycle one of the following events is used to reduce the distance between both genomes by one (see Figure 6): a) a one-cycle is detached from an α-path of length greater than two (cut-and-join), b) an α-path of length two is replaced by a one-cycle (join), c) an α'-path is split into two odd paths (cut), or d) a cycle of length greater than two is replaced by an even path (cut). The algorithm

terminates after $d_{\mathcal{SCAJ}}(\alpha, \alpha')$ steps, because each of these operations reduces the distance by one. See [21] for a detailed analysis and a formal proof.

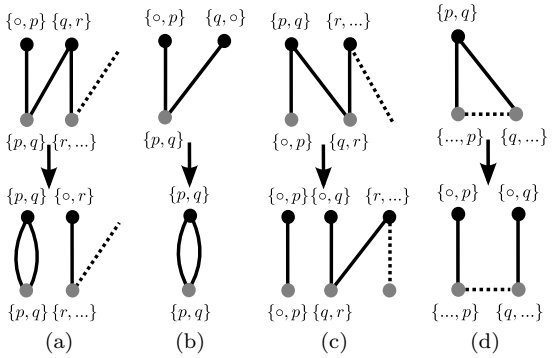


Fig. 6 Cases $a) - d)$ of the algorithm for sorting α into α' under $\mathcal{SCAJ}$ shown on the adjacency graph. Top) components of $AG(\alpha, \alpha')$ and bottom) resulting components of $AG(\alpha \circ \rho, \alpha')$, where ρ is either a cut, a join, or a cut-and-join with $(\alpha, \alpha \circ \rho) \in \mathcal{SCAJ}$ and $d_{\mathcal{SCAJ}}(\alpha, \alpha') > d_{\mathcal{SCAJ}}(\alpha \circ \rho, \alpha')$.

Open Problems

The complexity of other rearrangement problems and the distance problem where only linear chromosomes are allowed remain open.

3.4 Double-Cut and Join Model

The $\mathcal{DCJ}$ model acts on mixed genomes. This is the case, e.g., for genomes that contain plasmids in addition to the linear chromosomes [46, 94, 109] or in tumor genomes [95]. In the $\mathcal{DCJ}$ model an event cuts two adjacencies and forms two new adjacencies by joining four telomeres, see Figure 7. More formally, a pair of genomes (α, α') is an event in $\mathcal{DCJ}$ if and only if $(\alpha, \alpha') \in \mathcal{SCAJ}$ or there exist $p, q, r, s \in \mathcal{E}(\alpha)$ such that $\alpha \cup \{\{p, s\}, \{r, q\}\} = \alpha' \cup \{\{p, q\}, \{r, s\}\}$ (double-cut and join). Translocations, fusions, fissions, and inversions are represented directly in $\mathcal{DCJ}$. The operations block-interchange and transposition are included indirectly via two $\mathcal{DCJ}$ operations: the excision of a circular intermediate and the reinsertion at a different position. This can be implemented naturally by assigning a weight of two to block-interchanges and transpositions and a weight of one to translocations, fusions, fissions, and inversions [114]. The occurrence of such implicit transpositions, i.e., two consecutive $\mathcal{DCJ}$ rearrangements that are

equivalent to a transposition, in parsimonious $\mathcal{DCJ}$ scenarios was estimated to 17% of the rearrangement mutations for mammalian genomes [79].

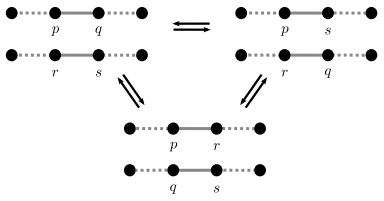


Fig. 7 Application of double-cut and join operation to adjacencies visualized on the genome graph. Note that the events of the submodel $\mathcal{SCAJ}$ which are illustrated in Figure 5 are also events of $\mathcal{DCJ}$.

Polynomial Time Solvable Problems

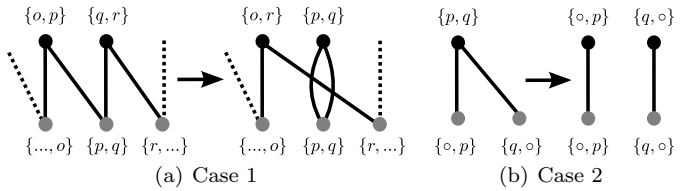
The distance problem under $\mathcal{DCJ}$ can be computed in linear time using the breakpoint graph [114]. The $\mathcal{DCJ}$ distance for two genomes α, α' can be computed by $d_{\mathcal{DCJ}}(\alpha, \alpha') = b - c$, where b is the number of breakpoints and c is the number of cycles in $BP(\alpha, \alpha')$. The web service `CEGeD` can be used to solve the distance problem under $\mathcal{DCJ}$, see Table 1. In order to apply the breakpoint graph for multichromosomal genomes so called capping, i.e., the addition of auxiliary end elements, and the concatenation of linear chromosomes is needed [114].

By using the adjacency graph a linear time algorithm was presented which has the advantage that also multichromosomal genomes are covered naturally [23]. With respect to the adjacency graph the $\mathcal{DCJ}$ distance is computed as $d_{\mathcal{DCJ}}(\alpha, \alpha') = N - (C + I/2)$ for two genomes α, α' with N genes where C is the number of cycles and I is the number of odd paths in $AG(\alpha, \alpha')$. The equivalence of the two distance formulas follows from the following facts: i) the number of breakpoints b equals $N + t_\alpha/2 + E_{\alpha'} = N + t_{\alpha'}/2 + E_\alpha$, where t_β is the number of telomeres of a genome β and E_β is the number of β-paths, ii) $t_\alpha/2 + t_{\alpha'}/2 = I + E$ and $E = E_\alpha + E_{\alpha'}$, and iii) $c = C + I + E$.

In the following a sketch of the proof of correctness of the $\mathcal{DCJ}$ distance formulas is given, for details see [23]. An alternative derivation is presented in [24]. The adjacency graph of two equal genomes consists of one-cycles (C_2) - one such cycle for each adjacency - and odd paths of length one (I_1) - one such path for each telomere. Hence sorting is equivalent to choosing $\mathcal{DCJ}$ events which increase the number of C_2 or I_1. Since a $\mathcal{DCJ}$ operation can: i) not modify the number of cycles and odd paths simultaneously, ii) increase the number of cycles at most by one, and iii) increase the number of odd paths at most by two it follows that $d_{\mathcal{DCJ}}(\alpha, \alpha') \geq N - (C + I/2)$ holds. The tightness

of this bound was shown by giving a greedy algorithm that always attains the bound. In each step the greedy algorithm can apply one of the following two cases, see Figure 8. Case 1: In the presence of the adjacencies $\{p,q\} \in \alpha'$ and $\{o,p\},\{q,r\} \in \alpha$ the $\mathcal{DCJ}$ event is $(\alpha, \{\alpha \setminus \{\{o,p\},\{q,r\}\}\} \cup \{\{p,q\},\{o,r\}\})$. This creates a new one-cycle and reduces the size of a path by two. Case 2: For telomeres $\{p,\circ\},\{q,\circ\} \in \alpha'$ and an adjacency $\{p,q\} \in \alpha$ the $\mathcal{DCJ}$ event is $(\alpha, \{\alpha \setminus \{p,q\}\} \cup \{\{p,\circ\},\{q,\circ\}\})$. This replaces an even path of length two by two odd paths of length one. Observe that the second event uses only one cut.

In [23] a linearization of a circular genome is modeled by an additional cut of a vestigial adjacency $\{\circ,\circ\}$ and two joins of the resulting telomeres to telomere adjacencies, see [63] for a detailed description.



(a) Case 1 (b) Case 2

Fig. 8 The two cases of the $\mathcal{DCJ}$ sorting algorithm [23] shown for the adjacency graph.

NP-Hard Problems

Many experimental studies do not determine the strandedness of the genes [48]. In such cases genomes are commonly represented by *unsigned permutations*. This motivates the *sorting problem for unsigned genomes (SUG)* under $\mathcal{DCJ}$ which was studied for the first time in [48]. The NP-hardness of this problem for unichromosomal unsigned genomes was shown by a reduction from the NP-hard *breakpoint graph decomposition (BGD)* problem [80]. The BGD problem is to find a decomposition of a breakpoint graph into a maximum number of edge-disjoint alternating cycles. Note that in contrast to the case of signed genomes in the breakpoint graph for unsigned genomes each gene corresponds to a single vertex. An approximation algorithm for solving the SUG problem for multichromosomal mixed genomes was presented in [48]. The algorithm is based on a modification of the approximation algorithm of [83] for the BGD problem.

Recall that similar to $\mathcal{DCJ}$ also for $\mathcal{INV}$ the distance problem for signed genomes is polynomial time solvable [71] whereas it is NP-hard for unsigned genomes [44]. This suggests that the complexity of the distance problem for rearrangements that cut and join two adjacencies is rather dependent on

the availability of gene strandedness information than on the set of allowed rearrangement events.

The median problem under $\mathcal{DCJ}$ was shown to be NP-hard by reduction of the BGD problem [105]. Note that also this result holds analogously for the $\mathcal{INV}$ model [45]. Since the median problem is NP-hard the small and the large parsimony problem are NP-hard too. For the median problem exact algorithms [113, 115] and also heuristics [1, 82] have been presented in the literature. Based on the heuristic median solver an algorithm for the small phylogeny problem has been developed in [1]. Examples of software tools which can be used to solve the small parsimony problem are `MGRA` and `PATHGROUPS` under $\mathcal{DCJ}$ and `SPP-DJC` under $\mathcal{DCJ} \cup \mathcal{INDEL}$, see Table 1.

Applications

In [21] the distance problems under $\mathcal{SCAJ}$ and $\mathcal{DCJ}$ have been calculated for 281 synteny blocks of the human-mouse comparison from [93] and for 1359 synteny blocks of the genomes of chimp, rhesus monkey, mouse, rat, and dog from [84] with respect to the human genome. The results show that the obtained distances under $\mathcal{SCAJ}$ are much larger than the distances under $\mathcal{DCJ}$. The authors stress that the goal of the $\mathcal{SCAJ}$ model is rather an indicator for evaluating results under the $\mathcal{DCJ}$ model than being a realistic evolutionary model. Based on their results, the authors suggest that single-cut operations are more important in the mouse-human evolution than in the chimp-human evolution. The pairwise $\mathcal{DCJ}$ distance between synteny blocks from genomes of *Shigella boydii*, *Shigella dysenteriae*, *Shigella flexneri*, and *Escherichia coli* have been calculated in [63]. The generation of the synteny blocks and the calculation of the $\mathcal{DCJ}$ distances have been made by `Mauve`, see Table 1 for more information.

The algorithm for the small phylogeny problem which has been developed in [1] was used to analyze 13 chloroplast DNAs of genomes of *Campanulaceae* and the mammalian dataset from [87] (including the presented phylogenetic tree) which consists of the genomes of human, rat, mouse, cat, dog, pig, and cow. The results show that the reconstructed ancestral genomes contain between 1 and 5 circular chromosomes that occur in immediate ancestors of the seven mammalian species. The total number of used $\mathcal{DCJ}$ operations is 486 and the total number of restricted $\mathcal{DCJ}$ operations, i.e., $\mathcal{DCJ}$ operations that do not produce any circular chromosomes, is 487. This result strengthens the conjecture that there is no biological evidence of circular chromosomes in the nuclear genomes of higher eukaryotes [41].

Many software tools are based on the $\mathcal{DCJ}$ model, see Table 1 for a overview. For instance, `GEvolutionS` can be used to simulate evolutionary genome rearrangement scenarios and `MLGO` and `GREDU` can be used for the case that the input genomes do not have the same gene content.

3.5 Multi-Cut and Join Model

Rearrangement operations like inversions, fusions, fissions, and translocations can easily be modeled by cutting a genome at two positions before applying rejoin operations. However, rearrangement operations that create three breakpoints, e.g., transpositions and inverse transpositions, i.e., transpositions were in addition the orientation of the transposed genes is changed, can be modeled only indirectly when at most two cuts are allowed, e.g., by using a pair of $\mathcal{DCJ}$ operations or three $\mathcal{SCAJ}$ operations. Therefore, a generalized rearrangement model has been suggested [3] which cuts a genome at $k \in \mathbb{N}$ positions followed by rejoining the resulting fragments into new order. The resulting model is called *multi-cut and join* model ($\mathcal{MCJ}$) and for a specific $k \in \mathbb{N}$ we refer it as *k-cut and join* model ($\mathcal{KCJ}$) [1]. Formally, a pair of genomes (α, α') is an event in $\mathcal{KCJ}$ if and only if $(\alpha, \alpha') \in (\mathcal{K}-1)\mathcal{CJ}$ or there exists a set of k adjacencies $K \subset \alpha$ such that $\alpha' \cup K = \alpha \cup X$, where X is a subset of $\{\{x, y\} : x, y \in \mathcal{E}(K), x \neq y\} \cup \{\{p, \circ\} : p \in \mathcal{E}(K)\}$ such that $\mathcal{E}(X) \cup \{p : \{p, \circ\} \in \mathcal{T}(X)\} = \mathcal{E}(K) \cup \{p : \{p, \circ\} \in \mathcal{T}(K)\}$, $|X| = k$, $X \cap K = \emptyset$, $1\mathcal{CJ} = \mathcal{SCAJ}$, and $2\mathcal{CJ} = \mathcal{DCJ}$. Note that for all $i \in [1 : k-1]$ this definition implies that a $(\mathcal{K}-i)\mathcal{CJ}$ event is a particular event of $\mathcal{KCJ}$ which can be understood as a k-cut and join event which leaves exactly i of the k adjacencies unchanged.

Polynomial Time Solvable Problems

The distance problem under $\mathcal{KCJ}$ can be solved in polynomial time by computing $d_{\mathcal{KCJ}}(\alpha, \alpha') = \lceil (N - C_s(\alpha, \alpha'))/(k-1) \rceil$ for genomes α and α', where $k \geq 2$ is the number of cuts and $C_s(\alpha, \alpha')$ is the size of the maximum partition of the cycles in $BP(\alpha, \alpha')$ where the total number of black edges in each subset of cycles is equal to $1 \bmod (k-1)$ [3]. For instance, $C_2(\alpha, \alpha')$ equals the number of cycles in $BP(\alpha, \alpha')$ (since for any number c of cycles $(c \equiv 1) \bmod 1$) and $C_3(\alpha, \alpha')$ gives the number of odd cycles in $BP(\alpha, \alpha')$. Hence for $k = 2$, the formula equals the $\mathcal{DCJ}$ distance formula presented in [114]. Further, two algorithms for computing the distance formula between two genomes – a dynamic programming algorithm which is practical for small values of k and a linear time algorithm with exponential (in k) time preliminary computations – were presented in [3].

Open Problems

The distance problem under $\mathcal{KCJ}$ for linear multichromosomal genomes was studied in [2] resulting in lower bounds for the rearrangement distance. To the

[1] In [3] these rearrangements are called *multi-break* rearrangements.

best of our knowledge no additional polynomial time algorithms have been published for the median problem and the small/large parsimony problem.

4 Rearrangement Models with Intergenic Regions

This section reviews results that incorporate the sizes of intergenic regions into the models for rearrangement analysis. In recent years researchers have argued that the sizes of intergenic regions can be relevant information for rearrangement analysis in order to improve phylogenetic distance estimation.

To incorporate the sizes of the intergenic regions into the rearrangement models, weighted genome graphs can be defined where each edge $\{p, q\}$ has a weight if p, q are extremities of neighboring genes or $q = \circ$ and $\{p, \circ\}$ is a telomere. The weight of an edge equals the size of the corresponding intergenic region, i.e., the intergenic region between the two neighboring genes, respectively the size of the intergenic region between the gene and the telomere. The weight $w(\alpha)$ of a genome α is then the sum of the weights of all edges of the genome graph of α, i.e., the total size of all intergenic regions of α. Then, weighted versions of the genome rearrangement problems can be defined. A weighted single-cut or join model is a single-cut or join model where the weight of the adjacency that is broken by a cut equals the sum of the weights of the two telomeres that are created and the sum of the weights of the adjacency that is formed by a join equals the sum of the weights of the two telomeres that are connected. Weighted versions of single-cut and join models or multi-cut and join models are defined analogously.

The computational complexity and approximation-algorithms are known for various weighted rearrangement problems that consider the size of intergenic regions. Sorting with weighted reversals for unsigned permutations [40] and for signed permutations [89] is NP-hard. The former problem has a 4-approximation algorithm [40] and the latter problem has a 2-approximation algorithm [89]. Sorting with weighted reversals and weighted transpositions problem is NP-hard for unsigned and signed permutations [91]. The former problem has a 4-approximation algorithm [52] and the latter problem has a 3-approximation algorithm [91]. Sorting with weighted transpositions for unsigned permutations is NP-hard [90] (Note, that already the unweighted version of this problem is NP-hard [42]) and has 3.5-approximation algorithms [90]. The Block-Interchange is a rearrangement event that swaps the position of two segments (not necessarily consecutive) of the genome. The sorting by intergenic block-interchange problem has a 2-approximation algorithm [52] and its complexity is unknown.

A weighted version of the DCJ distance problem was studied in [58] where a weighted DCJ operation was defined as a DCJ operation which changes the weights of the edges such that the sum of the weights of the two cutted edges equals the weights of the two joined edges and the weights of all other edges

do not change. It was shown that the wDCJ distance problem is strongly NP-hard and has a 4/3-approximation algorithm.

There exist mutations that change the size of intergenic regions but do not change the order of the genes. To consider such mutations within the rearrangement models an i-indel operation is defined as an operation that changes the size of an intergenic region. The weight of an i-indel operation is the corresponding amount of size change. The sorting problem for the wDCJ model with i-indel operations (i.e., to find a shortest wDCJ scenario with minimum total weight of the i-indel operations) can be solved in time $O(n \log n)$ [43]. Several approximation algorithms have been designed for the problem of sorting unsigned permutations with i-indels in [7]: i) 4-approximation algorithm for sorting with i-indels and weighted reversals has a, ii) a 4.5-approximation algorithm for sorting with i-indels and weighted transpositions, and iii) a 6-approximation algorithm for sorting with i-indels, weighted reversals, and weighted transpositions. For signed permutations the latter problem has a 3-approximation algorithm [6].

Since also the gene order rearrangement operations might change the size of the intergenic regions corresponding extended gene order rearrangement operation have been studied. Transpositions that might remove a part of an intergenic region (i.e., a part of the genome that does not contain a gene) to include it into another intergenic region has been considered in [91, 52]. It was shown that for such generalized transpositions improved approximation algorithms exist for the following cases: i) a 2.5-approximation algorithm for sorting unsigned permutation with (generalized) weighted transpositions [91] and ii) sorting with (generalized) weighted transpositions for signed permutations and unsigned permutations has a 2.5-approximation algorithm [91], respectively 3-approximation algorithm [52].

5 Preserving Genome Rearrangement Models

This section discusses the sorting problem, the median problem, and the small parsimony problem under $\mathcal{INV}^p$. Some results for related preserving rearrangement models are discussed at the end of the section.

The sorting problem under $\mathcal{INV}^p$ has been introduced in [60] with the objective to produce biologically more relevant results. Unfortunately, the sorting problem under $\mathcal{INV}^p$ is NP-hard [60]. But, it is fixed-parameter tractable [37] and there exist linear run time algorithms for many relevant instances [17, 20], and algorithms with a polynomial average run time [37]. Algorithms for the median problem and the small parsimony problem under $\mathcal{INV}^p$ that have a polynomial run time for many relevant data sets have been proposed in [26, 28], see Section 5.2.

5.1 Strong Interval Tree

The strong interval tree (SIT)[2] is the central data structure for efficient preserving rearrangement analysis. The main reason is that the SIT can be computed in linear time and represents the common intervals (which can be quadratic in number) in linear space [17].

Let Π be a set of k genomes that are represented by signed permutations and λ be a consistent reference permutation (typically $\lambda \in \Pi$). A common interval $I \in C(\Pi)$ is *strong* if every other common interval in $C(\Pi)$ is either included in I or it includes I. By definition the strong intervals of Π form a hierarchy which is captured by the *strong interval tree* (SIT). The SIT of Π with respect to λ, denoted by $T^{\lambda}(\Pi)$, is an ordered tree where each vertex represents a strong common interval of Π and the edges represent the minimum inclusion relation. The order of the children of a vertex in the SIT are without loss of generality given by their order in the reference permutation λ. For every inner vertex I of a SIT with children $I_1, \ldots, I_l$ that are ordered according to the reference permutation λ the *quotient permutation* associated with I for $\pi \in \Pi$ is the permutation π_I^{λ} which satisfies that $\pi_I^{\lambda}(i)$ precedes $\pi_I^{\lambda}(j)$ if and only if I_i is to left of I_j in π, for $i \neq j$. A quotient permutation is called *linear increasing* if it is equal to ι, *linear decreasing* if it is $(l\ (l-1) \ldots 1)$, and *prime* otherwise. A vertex I of a SIT is called *linear* if all k quotient permutations are either linear increasing or linear decreasing. Otherwise it is called *prime*. It holds that a vertex of a SIT is linear if and only if every union of consecutive children is a common interval, whereas for a prime vertex only the union of all children is a common interval [17].

A so called *k-sign* $\{+1, -1\}^k$ is assigned to the vertices to represent the orientation of linear quotient permutations and the orientation of the elements in the case of leaf vertices. For a vertex I the i-th element of a k-sign, denoted by $s(i)$, is determined based on the following rules: i) A leaf I has $s(i) = +1$ if the corresponding element has the same sign in π_i as in λ and $s(i) = -1$ otherwise. ii) A linear inner vertex I has $s(i) = +1$ if π_{iI}^{λ} is the identity permutation and -1 otherwise. iii) A prime vertex with linear parent inherits the $k - sign$ of the parent. A SIT is *unambiguous* if every prime vertex has a linear parent and *ambiguous* otherwise. SITs without prime vertices are called *definite*. In the *signed quotient permutation* each element is assigned the sign of the corresponding child node, i.e., $\pi_{iI}^{\lambda}(j)$ is assigned the i-th component of the k-sign of the j-th child. For a linear vertex I of the SIT a sign $s_I(i)$ contains the same information as the quotient permutation π_{iI}^{λ}, i.e., there is a bijective relation between signs of linear vertices of definite SITs and consistent permutations. Note that these rules uniquely determine the signs of every vertex of unambiguous and definite SITs, but the signs of a prime root vertex and prime vertices with prime parent in ambiguous trees

[2] Note that strong interval trees are similar to PQ-trees [35].

are not determined. An example illustrating a SIT for a set of permutations is shown in Figure 9.

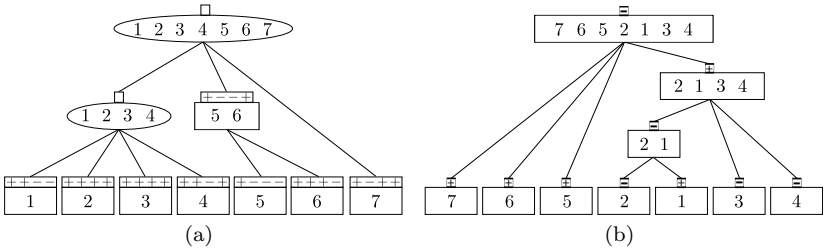


Fig. 9 (a) Strong common interval tree of $\Pi = \{(1 \dots 7), (-7\ 1\ 3\ 2\ 4\ 6\ -5), (6\ -5\ -4\ 3\ -1\ 2\ 7), (-5\ -6\ 7\ 3\ 4\ 2\ -1)\}$ where the trivial common intervals, $\{5, 6\}$, and $\{1, 2, 3, 4\}$ are the strong common intervals of Π. Prime vertices and linear vertices are represented by ellipses and rectangles, respectively. The root vertex and vertex $\{1, 2, 3, 4\}$ are prime and vertex $\{5, 6\}$ is linear increasing with respect to $(1 \dots 7)$. (b) Definite SIT $T^{\iota}(\Sigma)$ for $\Sigma = \{(7\ 6\ 5\ -2\ 1\ -3\ -4), \iota\}$. Note that every vertex is linear since every union of consecutive children of each vertex is a common interval of Σ. The signs of the vertices are shown at the top of the rectangles. The signs with respect to ι are omitted since it is $+$ for all vertices. Vertex $\{2, 1\}$ is linear decreasing since its quotient permutation is (2 1) and the quotient permutation (1 2 3) of vertex $\{2, 1, 3, 4\}$ implies that it is linear increasing. A scenario sorting $\pi_1 = (7\ 6\ 5\ -2\ 1\ -3\ -4)$ to $(-7\ -6\ -5\ -4\ -3\ -2\ -1)$ is obtained by applying inversions $\{7\}$, $\{6\}$, $\{5\}$, $\{4\}$, $\{3\}$, $\{1\}$, $\{1, 2\}$, and $\{1, 2, 3, 4\}$ to π_1. Recall that ι and $(-7\ -6\ -5\ -4\ -3\ -2\ -1)$ are considered to be equal.

5.2 Algorithms for the Preserving Inversion Model

Algorithms for the sorting problem under $\mathcal{INV}^p$ for signed unichromosomal linear genomes and a partition of the set of problem instances into instances, that can be solved in linear time, polynomial time, or exponential time have been presented in [17]. More precisely, problem instances with a definite SIT can be solved in linear time, those with an unambiguous SIT can be solved in sub-quadratic time, and instances with an ambiguous SIT have a worst case exponential run time.

In the following we will identify inversions by the interval that they reverse. Consider a signed permutation π that is to be transformed into ι. Note that arbitrary pairs of signed permutations can be reduced to this case by renaming the elements of the permutations. Let $T^{\iota}(\Pi)$ be the strong interval tree of $\Pi = \{\pi, \iota\}$. Instead of the 2-sign at each prime vertex with linear parent and each linear vertex a 1-sign (or just sign for short) is stored at the vertices. The sign is equal to the 2-sign component that corresponds to π. This is because the sign of the target permutation ι is $+$ for all vertices with

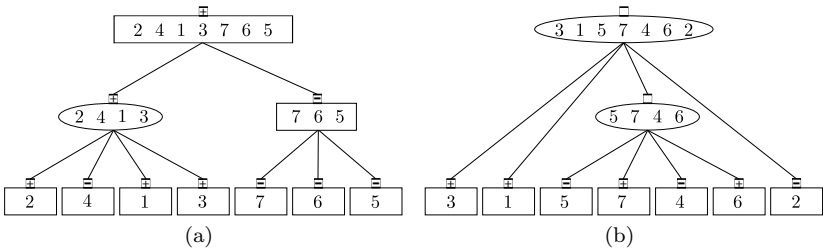


Fig. 10 (a): Unambiguous SIT $T^{\iota}(\Pi)$ for $\Pi = \{\pi = (2\ -4\ 1\ 3\ -7\ -6\ -5), \iota\}$. Vertex $\{2\ 4\ 1\ 3\}$ is prime since its quotient permutation is $(2\ 4\ 1\ 3)$. Its parent vertex is linear increasing which results in an unambiguous SIT. A parsimonious scenario sorting $(2\ -4\ 1\ 3)$ to ι is given by the inversions $\{1,3,4\}$, $\{2,3\}$, $\{3\}$, and $\{1,2,3\}$. Applying the scenario to π yields $\pi' = (1\ 2\ 3\ 4\ -7\ -6\ -5)$ which has a definite SIT $T^{\iota}(\pi', \iota)$. For π' the inversion $\{5,6,7\}$ is a parsimonious scenario. Both scenarios together are a parsimonious scenario for the unambiguous SIT $T^{\iota}(\Pi)$. (b): Example of an ambiguous SIT. Here, the signs of prime vertices of $T^{\iota}(\{(3\ 1\ -5\ 7\ -4\ 6\ -2), \iota\})$ are not uniquely determined. A parsimonious scenario is obtained by a negative sign for the root vertex and a positive sign for its child prime vertex.

a sign. Therefore, we can call a vertex linear increasing (resp. decreasing) if the quotient permutation is linear increasing (resp. decreasing).

The key for solving the sorting problem under $\mathcal{INV}^p$ is that an inversion is preserving if and only if it is a vertex of the SIT or a union of children of a prime vertex [17]. Thus, linear vertices can only be reversed as a whole whereas the children of prime vertices can be freely rearranged. With respect to the SIT the problem of sorting by preserving inversions is to apply rearrangements such that all quotient permutations are transformed into the identity permutation, i.e., the vertices become linear increasing. Hence, if a the sign of vertex I of $T^{\iota}(\pi)$ is different to the sign of its parent vertex, it holds that the inversion defined by I is always part of any optimal sorting scenario [17] (Lemma 2), i.e., the inversions that correspond to vertices with a sign different to the one of their parent define a parsimonious scenario [17] (Theorem 2). This can easily be implemented to run in linear time, see Figure 9(b) for an example. Since the inversions in parsimonious preserving scenarios for a problem instance with a definite strong interval tree commute the set of all parsimonious scenarios can be obtained easily.

For a problem instance with unambiguous SIT a method to transform the signed quotient permutation of prime vertices into the identity permutation is needed. Since the children of a prime vertex can be freely rearranged a solution is to reconstruct an unconstrained parsimonious inversion scenario for the signed quotient permutation. That is, for every prime vertex I a parsimonious scenario from π_I^{λ} to ι is computed if I has a positive sign. If it has a negative sign a scenario from π_I^{λ} to $(-n\ \ldots\ -1)$ is computed. Applying the corresponding inversions to π results in a definite SIT which can be processed

as explained beforehand. The run time of the algorithm for unambiguous SITs is dominated by solving the sorting problem for prime vertices which is done by the algorithm presented in [103]. The algorithm from [103] solves the sorting problem under $\mathcal{INV}$ for signed permutations in sub-quadratic run time $O(n\sqrt{n\log(n)})$. An example for sorting an unambiguous SIT under $\mathcal{INV}^p$ is given in Figure 10(a).

For ambiguous SITs the signs of prime vertices with prime parent are unknown, see Figure 10(b). By trying every combination of sign assignments and applying the algorithm for unambiguous SITs an exact solution can be obtained. An improvement is possible by applying dynamic programming separately for every component of connected prime vertices. Assume that $u \in \mathbb{N}$ unsigned prime vertices exist, in this case the described algorithm for ambiguous SITs (which uses the algorithm from [103]) has an exponential run time of $O(2^u n\sqrt{n\log(n)})$ in the worst case.

In the following the algorithm of [26] for the small parsimony problem under $\mathcal{INV}^p$ for sets of genomes with a definite SIT is presented. The algorithm runs in polynomial time. The basis of the approach is an extension of the results of [19] which was first used in an approach for solving the preserving inversion median problem [28]. The idea of the algorithm is to extract binary characters from the SIT which are then processed by Fitch's algorithm [61] for maximum parsimony. Figure 11 illustrates the algorithm which is described below.

Again, the foundation of the algorithm is the insight that an inversion is preserving with respect to a set of permutations Π if and only if it is either a vertex of $T(\Pi)$ or a union of consecutive child vertices of a prime vertex. This result from [26] is a generalization of a result of [17]. Hence, for a linear vertex preserving inversions can only change its order. Therefore, for each linear vertex it needs to be determined in which of the permutations it has to be inverted or not. For a set of permutations $\Pi = \{\pi_1, \ldots, \pi_k\}$ the Parity Lemma [17] and its reformulation for SITs with k-signs [26] states that a vertex I has to be inverted in permutation π_i in a parsimonious preserving scenario if and only if its sign $s_I(i)$ differs from the sign $s_J(i)$ of its parent vertex J. Hence, the essential information is the difference of the k-signs of the vertices and their respective parents. To capture these differences every vertex of the SIT $T^\lambda(\Pi)$ is assigned to a binary character representing the information on the equality or inequality of its k-signs and the k-signs of its parent. Formally, consider a non-root vertex I and its parent vertex J of $T^\lambda(\Pi)$ with k-signs s_I and s_J. The binary *character state assignment* of I is a k-tuple c_I such that $c_I(i) = s_I(i) \cdot s_J(i)$ for $i \in [1:k]$. The character state assignment for the root vertex is defined by comparing its k-sign with $\{+1\}^k$. Note that a character state assignment for every vertex of $T^\lambda(\Pi)$ uniquely determines Π since π_i is determined by multiplying $c_I(i)$ and parents sign $s_J(i)$ from top to bottom starting at root vertex with $+1$. Hence, each assignment of character states to the nodes of the SIT uniquely determines a consistent permutation, see Figure 11(a) for an example.

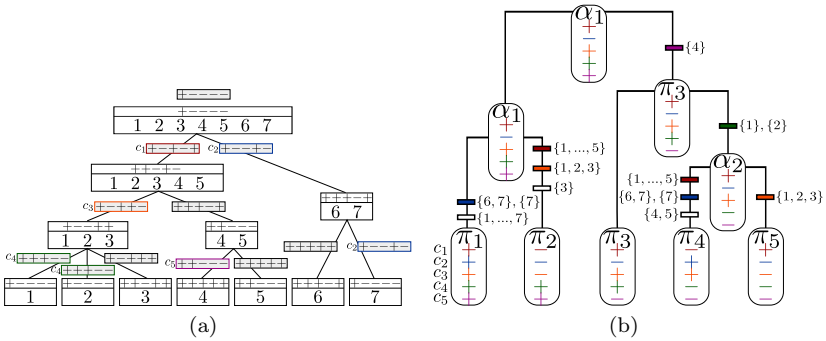


Fig. 11 (a): SIT $T^{\iota}(\Pi)$ with k-signs for $\pi_1 = \iota$, $\pi_2 = (6\ -7\ 3\ -2\ -1\ 4\ 5)$, $\pi_3 = (6\ -7\ -5\ 4\ -3\ -2\ -1)$, $\pi_4 = (-7\ -6\ -1\ -2\ 3\ -5\ 4)$, and $\pi_5 = (6\ -7\ -5\ 4\ -1\ -2\ 3)$. The character state of a vertex is placed on the edge to its parent. Character state assignments of vertex $\{1\}$ is $(+\,+\,+\,-\,-)$ because the k-signs of vertex $\{1\}$ and vertex $\{1,2,3\}$ are equal in the first three positions and unequal in positions four and five. SIT $T^{\iota}(\Pi)$ shows five unique, nontrivial characters which are denoted by $c_1, \ldots, c_5$. The permutation π_2 is uniquely determined by character states and k-signs since the signs of all vertices in pre-order are $-1, +1, -1, -1, -1, +1, +1, +1, +1, +1, +1, -1$. (b): Optimal solution of the small parsimony problem under $\mathcal{INV}^p$ which is obtained by Fitch's algorithm [61] for T and nontrivial, unique characters $c_1, \ldots, c_5$. Ancestral permutations $\alpha_1 = (6\ -7\ -5\ -4\ -3\ -2\ -1)$ and $\alpha_2 = (6\ -7\ -5\ 4\ -3\ 2\ 1)$ are determined by preserving inversions represented on the edges. Root permutation α_1 is constructed from π_3 by reverting element $\{4\}$. This information is obtained since the character state assignment of both vertices differs only in one sign of c_5 which corresponds to vertex $\{4\}$ in the SIT.

Due to the bijective relation between consistent permutations, character state assignments, and preserving inversions, the small parsimony problem for a set of permutations Π with definite SIT and phylogenetic tree T can be solved exactly and efficiently by solving the small parsimony problem for the character states for the vertices of $T^{\lambda}(\Pi)$. Consider a set of permutations Π with a definite SIT $T^{\lambda}(\Pi)$, the character state assignments of its vertices, and a given tree T. Solving the small parsimony problem for T and the binary characters defined for the vertices yields a reconstruction of ancestral character states of T. Given the ancestral character states of a vertex of T, the signs for the vertices of the SIT and the corresponding permutations are constructed by the following procedure. Starting with the identity permutation all strong common intervals need to be inverted for which the sign differs from the one of its parent. See Figure 11(b) for an example. This algorithm runs in $O(kn)$ time for a set Π of k permutations of length n if $T^{\lambda}(\Pi)$ is definite. Because of the one-to-one relation between the characters defined by the SIT and parsimonious inversions the large parsimony problem for binary characters under $\mathcal{INV}^p$ is NP-hard as well – even for sets of permutations with definite SIT [26].

In [28] an exact algorithm – called TCIP – for the median problem under $\mathcal{INV}^p$ was presented. TCIP uses the bijective relations between consistent permutations, character state assignments, and preserving inversions which yields a linear run time for definite SITs. For ambiguous SITs different unconstrained versions of the median problem have to be solved for the quotient permutations of prime vertices considering different k-sign assignments which significantly increases the run time of TCIP. However, it has been shown empirically that median problems of random gene orders as well as organellar gene orders often have a definite SIT and then TCIP has a good performance.

Application

In [26] the small phylogeny problem has been analyzed under $\mathcal{INV}^p$ for 4 unique γ-*Proteobacteria* gene orders from [16] and for a set of *Burkholderia* gene orders. Pairwise scenarios under $\mathcal{INV}^p$ between 16 synteny blocks of the X Chromosome of the human, mouse, and rat genomes from [66] have been constructed in [17]. The results show that the SIT for the rat and mouse comparison is definite and that a parsimonious preserving scenario contains 11 inversion. The comparison between human and rat (resp. mouse) shows an unambiguous SIT which results into a preserving scenario with 13 inversions (resp. 12 inversions).

5.3 Related Preserving Problems

Phylogenetic reconstructions should consider all relevant rearrangement operations. Mitochondrial gene orders, for instance, evolve by inversions, (inverse) transpositions, and tandem-duplication-random-loss (TDRL) events (i.e., a tandem duplication of a continuous sequence of genes followed by the loss of one copy of each duplicated gene [47]). In particular TDRLs are important rearrangement events in mitochondrial genomes [34, 78, 96]. The algorithm CREx which solves the preserving sorting problem heuristically under these four rearrangement operations has been presented in [30]. The basic principle of CREx is the detection of patterns in the SIT that determine corresponding rearrangement events: i) An inversion corresponds to a vertex with a different sign as its parent, ii) a transposition is represented by a vertex that has two children and its sign differs from the one of its children and its parent, iii) an inverse transposition corresponds to a vertex whose sign is different to its parents signs and the signs of all but the first (or last) children, and iv) a TDRL corresponds to a prime vertex whose quotient permutation has only positive or only negative elements. CREx uses a step wise approach which identifies (inverse) transposition patterns and inversion patterns in linear vertices of SITs. In the second step prime vertices are heuristically solved by combining

TDRL and inversion events. For a detailed description of CREx the reader is referred to [25].

The sorting problem under $\mathcal{DCJ}^p$ was studied in [18, 19]. In this study a version of the preserving property was applied which allows to cut out circular intermediates consisting of a subset of common intervals. Polynomial time algorithms for multichromosomal mixed genomes with SIT consisting of prime vertices and linear vertices with two children only, e.g., ambiguous SITs, were given. The sorting problem under $\mathcal{DCJ}^p$ turns out to be NP-hard for definite and unambiguous SITs [19]. The problem is already difficult for linear vertices with three children, because it is impossible to decide for a parsimonious sorting direction. However, Bérard et al. presented a fixed parameter polynomial time algorithm which uses a specific pattern of common intervals as a parameter to solve the sorting problem for definite SITs [19]. The average-time complexity of their algorithm is still open. Interestingly, the results of [19] for the sorting problem imply the first case where the models $\mathcal{INV}^p$ and $\mathcal{DCJ}^p$ differ in complexity.

Application

CREx has been used in [31] to study the phylogeny of 185 Metazoa mitochondiral gene orders. In particular, the gene orders of Arthropod species and the Chordata species have been studied. A comprehensive analysis of the results is given in [25]. Furthermore, CREx has been used in several publications for the analysis of mitochondrial gene orders. A recent example is [8], where the mitochondrial genome of *Aglaiogyrodactylus forficulatus* was compared with the genomes of other monogenoidean flatworm species.

6 Conclusion

In this chapter we have given an overview on genome rearrangement analysis. The focus was on the following two approaches. The first approach is the development of models for rearrangement mutations that are well suited for a theoretical analysis. In particular, models that use the two operations cut, i.e., to split a chromosome into fragments, and join, i.e., to merge fragments of chromosomes to new chromosomes. The second approach is to identify biologically motivated constraints on the applicability of gene rearrangement mutations. In particular, constraints that model conserved gene clusters have been considered. Several open problems for genome rearrangement analysis have been mentioned.

References

1. Adam, Z., Sankoff, D.: The ABCs of MGR with DCJ. Evol Bioinform Online **4**, 69–74 (2008)
2. Alekseyev, M.A.: Multi-break rearrangements and breakpoint re-uses: from circular to linear genomes. J Comput Biol **15**(8), 1117–1131 (2008)
3. Alekseyev, M.A., Pevzner, P.A.: Multi-break rearrangements and chromosomal evolution. Theor Comput Sci **395**(2), 193–202 (2008)
4. Alekseyev, M.A., Pevzner, P.A.: Breakpoint graphs and ancestral genome reconstructions. Genome Res **19**(5), 943–957 (2009)
5. Alekseyev, M.A., Pevzner, P.A.: `MGRA`. `http://mgra.cblab.org/` (2009)
6. Alexandrino, A.O., Brito, K.L., Oliveira, A.R., Dias, U., Dias, Z.: Reversal distance on genomes with different gene content and intergenic regions information. In: International Conference on Algorithms for Computational Biology, pp. 121–133. Springer (2021)
7. Alexandrino, A.O., Oliveira, A.R., Dias, U., Dias, Z.: Incorporating intergenic regions into reversal and transposition distances with indels. Journal of Bioinformatics and Computational Biology **19**(06), 2140,011 (2021)
8. Bachmann, L., Fromm, B., Patella de Azambuja, L., Boeger, W.A.: The mitochondrial genome of the egg-laying flatworm aglaiogyrodactylus forficulatus (platyhelminthes: Monogenoidea). Parasit Vectors **9**(1), 1–8 (2016)
9. Bader, D.A., Moret, B.M., Warnow, T., Wyman, S.K., Yan, M., Tang, J., Siepel, A.C., Caprara, A.: `GRAPPA`. `https://www.cs.unm.edu/~moret/GRAPPA/` (2004)
10. Bader, M.: `dcjdDist`. `http://www.uni-ulm.de/in/theo/m/alumni/bader/` (2009)
11. Bader, M.: Sorting by reversals, block interchanges, tandem duplications, and deletions. BMC Bioinformatics **10**(Suppl 1), S9 (2009)
12. Bader, M.: The transposition median problem is NP-complete. Theor Comput Sci **412**(12-14), 1099–1110 (2011)
13. Bader, M., Abouelhoda, M.I., Ohlebusch, E.: `MGR`. `http://grimm.ucsd.edu/MGR/` (2002)
14. Bader, M., Abouelhoda, M.I., Ohlebusch, E.: A fast algorithm for the multiple genome rearrangement problem with weighted reversals and transpositions. BMC Bioinformatics **9**(1), 1–13 (2008)
15. Bader, M., Abouelhoda, M.I., Ohlebusch, E.: `phylo`. `http://www.uni-ulm.de/in/theo/m/alumni/bader/` (2008)
16. Belda, E., Moya, A., Silva, F.J.: Genome rearrangement distances and gene order phylogeny in γ-proteobacteria. Mol Biol Evol **22**(6), 1456–1467 (2005)
17. Bérard, S., Bergeron, A., Chauve, C., Paul, C.: Perfect sorting by reversals is not always difficult. IEEE/ACM Trans Comput Biol Bioinform **4**(1), 4–16 (2007)
18. Bérard, S., Chateau, A., Chauve, C., Paul, C., Tannier, E.: Perfect DCJ rearrangement. In: Proc. RECOMB Int'l Workshop Comparative Genomics (RCG '08), *LNCS*, vol. 5267, pp. 158–169 (2008)
19. Bérard, S., Chateau, A., Chauve, C., Paul, C., Tannier, E.: Computation of perfect DCJ rearrangement scenarios with linear and circular chromosomes. J Comput Biol **16**(10), 1287–1309 (2009)
20. Bérard, S., Chauve, C., Paul, C.: A more efficient algorithm for perfect sorting by reversals. Inf Process Lett **106**(3), 90–95 (2008)
21. Bergeron, A., Medvedev, P., Stoye, J.: Rearrangement models and single-cut operations. J Comput Biol **17**(9), 1213–1225 (2010)
22. Bergeron, A., Mixtacki, J., Stoye, J.: `CEGeD`. `http://bibiserv.techfak.uni-bielefeld.de/ceged` (2006)
23. Bergeron, A., Mixtacki, J., Stoye, J.: A unifying view of genome rearrangements. In: Proc. 6th Int'l Workshop Algorithms in Bioinformatics (WABI '06), *LNCS*, vol. 4175, pp. 163–173 (2006)

24. Bergeron, A., Stoye, J.: The Genesis of the DCJ Formula, *Computational Biology*, vol. 19, pp. 63–81. Springer (2013)
25. Bernt, M.: Gene order rearrangement methods for the reconstruction of phylogeny. Ph.D. thesis, University Leipzig (2009)
26. Bernt, M., Chao, K.M., Kao, J.W., Middendorf, M., Tannier, E.: Preserving inversion phylogeny reconstruction. In: Proc. 12th Int'l Workshop Algorithms in Bioinformatics (WABI '12), *LNCS*, vol. 7534, pp. 1–13 (2012)
27. Bernt, M., Merkle, D., Middendorf, M.: A fast and exact algorithm for the perfect reversal median problem. In: Proc. 3rd Int'l Symp. Bioinformatics Research and Applications (ISBRA '07), *LNCS*, vol. 4463, pp. 305–316 (2007)
28. Bernt, M., Merkle, D., Middendorf, M.: Solving the preserving reversal median problem. IEEE/ACM Trans Comput Biol Bioinform **5**(3), 332–347 (2008)
29. Bernt, M., Merkle, D., Ramsch, K., Fritzsch, G., Perseke, M., Bernhard, D., Schlegel, M., Stadler, P.F., Middendorf, M.: `CREx`. `http://pacosy.informatik.uni-leipzig.de/crex` (2007)
30. Bernt, M., Merkle, D., Ramsch, K., Fritzsch, G., Perseke, M., Bernhard, D., Schlegel, M., Stadler, P.F., Middendorf, M.: CREx: inferring genomic rearrangements based on common intervals. Bioinformatics **23**(21), 2957–2958 (2007)
31. Bernt, M., Middendorf, M.: A method for computing an inventory of metazoan mitochondrial gene order rearrangements. BMC Bioinformatics **12**(9), 1 (2011)
32. Bohnenkämper, L., Braga, M.D., Doerr, D., Stoye, J.: Computing the rearrangement distance of natural genomes. Journal of Computational Biology **28**(4), 410–431 (2021). DOI 10.1089/cmb.2020.0434
33. Bohnenkämper, L., Braga, M.D., Doerr, D., Stoye, J.: `DING`. `https://gitlab.ub.uni-bielefeld.de/gi/ding` (2021)
34. Boore, J.L.: The duplication/random loss model for gene rearrangement exemplified by mitochondrial genomes of deuterostome animals, pp. 133–147. Springer (2000)
35. Booth, K.S., Lueker, G.S.: Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. J Comput Syst Sci **13**(3), 335–379 (1976)
36. Bourque, G., Pevzner, P.A.: Genome-scale evolution: Reconstructing gene orders in the ancestral species. Genome Res **12**(1), 26–36 (2002)
37. Bouvel, M., Chauve, C., Mishna, M., Rossin, D.: Average-case analysis of perfect sorting by reversals. Discrete Math Algorithms and Appl **3**(3), 369–392 (2011)
38. Braga, M.D.: `baobabLUNA`. `http://doua.prabi.fr/software/luna#perm` (2008)
39. Braga, M.D.: baobabluna: the solution space of sorting by reversals. Bioinformatics **25**(14), 1833–1835 (2009)
40. Brito, K.L., Jean, G., Fertin, G., Oliveira, A.R., Dias, U., Dias, Z.: Sorting by genome rearrangements on both gene order and intergenic sizes. Journal of Computational Biology **27**(2), 156–174 (2020)
41. Brown, T.A.: Genomes. Garland science (2006)
42. Bulteau, L., Fertin, G., Rusu, I.: Sorting by transpositions is difficult. SIAM J Discrete Math **26**(3), 1148–1180 (2012)
43. Bulteau, L., Fertin, G., Tannier, E.: Genome rearrangements with indels in intergenes restrict the scenario space. BMC bioinformatics **17**(14), 225–231 (2016)
44. Caprara, A.: Sorting by reversals is difficult. In: Proc. 1th Ann. Int'l Conf. Computational Molecular Biology (RECOMB '97), pp. 75–83 (1997)
45. Caprara, A.: The reversal median problem. INFORMS J Comput **15**(1), 93–113 (2003)
46. Casjens, S., Palmer, N., van Vugt, R., Huang, W.M., Stevenson, B., Rosa, P., Lathigra, R., Sutton, G., Peterson, J., Dodson, R.J., Haft, D., Hickey, E., Gwinn, M., White, O., Fraser, C.M.: A bacterial genome in flux: the twelve linear and nine circular extrachromosomal DNAs in an infectious isolate of the Lyme disease spirochete *Borrelia burgdorferi*. Mol Microbiol **35**(3), 490–516 (2000)

47. Chaudhuri, K., Chen, K., Mihaescu, R., Rao, S.: On the tandem duplication-random loss model of genome rearrangement. In: Proc. 17th Ann. ACM-SIAM Symp. Discrete Algorithm (SODA '06), pp. 564–570 (2006)
48. Chen, X.: On sorting permutations by double-cut-and-joins. In: Proc. 16th Ann. Int'l Computing and Combinatorics Conf. (COCOON '10), *LNCS*, vol. 6196, pp. 439–448 (2010)
49. Christie, D.A.: Sorting permutations by block-interchanges. Inf Process Lett **60**(4), 165–169 (1996)
50. Darling, A.C., Mau, B., Blattner, F.R., Perna, N.T.: Mauve: multiple alignment of conserved genomic sequence with rearrangements. Genome Res **14**(7), 1394–1403 (2004)
51. Darling, A.C., Mau, B., Blattner, F.R., Perna, N.T.: `Mauve`. `http://darlinglab.org/mauve/mauve.html` (2015)
52. Dias, U., Oliveira, A.R., Brito, K.L., Dias, Z.: Block-interchange distance considering intergenic regions. In: Brazilian Symposium on Bioinformatics, pp. 58–69. Springer (2019)
53. Doerr, D., Chauve, C.: `SPP-DJC`. `https://github.com/danydoerr/spp_dcj` (2021)
54. Elias, I., Hartman, T.: A 1.375-approximation algorithm for sorting by transpositions. IEEE/ACM Trans Comput Biol Bioinform **3**(4), 369–379 (2006)
55. Feijão, P., Meidanis, J.: SCJ: a variant of breakpoint distance for which sorting, genome median and genome halving problems are easy. In: Proc. 9th Int'l Workshop Algorithms in Bioinformatics (WABI '09), *LNCS*, vol. 5724, pp. 85–96 (2009)
56. Feijão, P., Meidanis, J.: SCJ: a breakpoint-like distance that simplifies several rearrangement problems. IEEE/ACM Trans Comput Biol Bioinform **8**(5), 1318–1329 (2011)
57. Felsenstein, J., Felenstein, J.: Inferring phylogenies, vol. 2. Sinauer Associates Sunderland (2004)
58. Fertin, G., Jean, G., Tannier, E.: Algorithms for computing the double cut and join distance on both gene order and intergenic sizes. Algorithms for Molecular Biology **12**(1), 1–11 (2017)
59. Fertin, G., Labarre, A., Rusu, I., Tannier, E., Vialette, S.: Combinatorics of Genome Rearrangements, 1st edn. The MIT Press (2009)
60. Figeac, M., Varré, J.S.: Sorting by reversals with common intervals. In: Proc. 4th Int'l Workshop Algorithms in Bioinformatics (WABI '04), *LNCS*, vol. 3240, pp. 26–37 (2004)
61. Fitch, W.M.: Toward defining the course of evolution: minimum change for a specific tree topology. Syst Biol **20**(4), 406–416 (1971)
62. Foulds, L.R., Graham, R.L.: The steiner problem in phylogeny is NP-complete. Adv Appl Math **3**(1), 43–49 (1982)
63. Friedberg, R., Darling, A.E., Yancopoulos, S.: Genome Rearrangement by the Double Cut and Join Operation, *Methods in Molecular Biology*, vol. 452, pp. 385–416. Humana Press (2008)
64. Fu, Z., Chen, X., Vacic, V., Nan, P., Zhong, Y., Jiang, T.: MSOAR: a high-throughput ortholog assignment system based on genome rearrangement. J Comput Biol **14**(9), 1160–1175 (2007)
65. Fu, Z., Chen, X., Vacic, V., Nan, P., Zhong, Y., Jiang, T.: `MSOAR`. `http://msoar.cs.ucr.edu/index.php` (2009)
66. Gibbs, R.A., Weinstock, G.M., Metzker, M.L., Muzny, D.M., Sodergren, E.J., Scherer, S., Scott, G., Steffen, D., Worley, K.C., Burch, P.E., et al.: Genome sequence of the Brown Norway rat yields insights into mammalian evolution. Nature **428**(6982), 493–521 (2004)
67. Gog, S., Bader, M., Ohlebusch, E.: Genesis: genome evolution scenarios. Bioinformatics **24**(5), 711–712 (2008)
68. Gog, S., Bader, M., Ohlebusch, E.: `GENESIS`. `http://www.uni-ulm.de/en/in/institute-of-theoretical-computer-science/m/alumni/bader/` (2009)

69. Graham, G.J.: Tandem genes and clustered genes. J Theor Biol **175**(1), 71 – 87 (1995)
70. Hannenhalli, S., Pevzner, P.: Transforming men into mice (polynomial algorithm for genomic distance problem). In: Proc. 36th Ann. Symp. Foundations of Computer Science (FOCS '95), pp. 581–592 (1995)
71. Hannenhalli, S., Pevzner, P.A.: Transforming cabbage into turnip: Polynomial algorithm for sorting signed permutations by reversals. J ACM **46**(1), 1–27 (1999)
72. Heber, S., Stoye, J.: Finding all common intervals of k permutations. In: Proc. 12th Ann. Symp. Combinatorial Pattern Matching (CPM '01), *LNCS*, vol. 2089, pp. 207–218 (2001)
73. Hilker, R., Sickinger, C., Friesen, R., Mixtacki, J., Stoye, J.: `UniMoG`. `http://bibiserv.techfak.uni-bielefeld.de/dcj` (2005)
74. Hu, F., Lin, Y., Tang, J.: `MLGO`. `http://www.geneorder.org/server.php` (2014)
75. Hu, F., Lin, Y., Tang, J.: MLGO: phylogeny reconstruction and ancestral inference from gene-order data. BMC Bioinformatics **15**(1), 1–6 (2014)
76. Huang, Y.L., Huang, C.C., Tang, C.Y., Lu, C.L.: `SoRT`^{2}. `http://genome.cs.nthu.edu.tw/SORT2/` (2009)
77. Huang, Y.L., Lu, C.L.: Sorting by reversals, generalized transpositions, and translocations using permutation groups. J Comput Biol **17**(5), 685–705 (2010)
78. Inoue, J.G., Miya, M., Tsukamoto, K., Nishida, M.: Evolution of the deep-sea gulper eel mitochondrial genomes: large-scale gene rearrangements originated within the eels. Mol Biol Evol **20**(11), 1917–1924 (2003)
79. Jiang, S., Alekseyev, M.A.: Implicit transpositions in shortest DCJ scenarios. In: Proc. 2nd Int'l Conf. Algorithms for Computational Biology (AlCoB '15), *LNCS*, vol. 9199, pp. 13–24 (2015)
80. Kececioglu, J., Sankoff, D.: Exact and approximation algorithms for sorting by reversals, with application to genome rearrangement. Algorithmica **13**(1-2), 180–210 (1995)
81. Krell, P.: `GEvolutionS`. `http://bibiserv.techfak.uni-bielefeld.de/gevolutions` (2014)
82. Lenne, R., Solnon, C., Stützle, T., Tannier, E., Birattari, M.: Reactive stochastic local search algorithms for the genomic median problem. In: Proc. 8th European Conf. Evolutionary Computation in Combinatorial Optimisation (EvoCOP '08), *LNCS*, vol. 4972, pp. 266–276 (2008)
83. Lin, Y., Moret, B.M.: Estimating true evolutionary distances under the DCJ model. Bioinformatics **24**(13), 114–122 (2008)
84. Ma, J., Zhang, L., Suh, B.B., Raney, B.J., Burhans, R.C., Kent, W.J., Blanchette, M., Haussler, D., Miller, W.: Reconstructing contiguous regions of an ancestral genome. Genome Res **16**(12), 1557–1565 (2006)
85. Martin, M.: `SBBI`. `http://bibiserv.techfak.uni-bielefeld.de/sbbi` (2007)
86. Moret, B.M.E., Wang, L.S., Warnow, T., Wyman, S.K.: New approaches for reconstructing phylogenies from gene order data. Bioinformatics **17**(9), 165–173 (2001)
87. Murphy, W.J., Larkin, D.M., der Wind, A.E.v., Bourque, G., Tesler, G., Auvil, L., Beever, J.E., Chowdhary, B.P., Galibert, F., Gatzke, L., Hitte, C., Meyers, S.N., Milan, D., Ostrander, E.A., Pape, G., Parker, H.G., Raudsepp, T., Rogatcheva, M.B., Schook, L.B., Skow, L.C., Welge, M., Womack, J.E., O'Brien, S.J., Pevzner, P.A., Lewin, H.A.: Dynamics of mammalian chromosome evolution inferred from multispecies comparative maps. Science **309**(5734), 613–617 (2005)
88. Ohlebusch, E., Abouelhoda, M., Hockel, K.: A linear time algorithm for the inversion median problem in circular bacterial genomes. J Discrete Algorithms **5**(4), 637–646 (2007)
89. Oliveira, A.R., Jean, G., Fertin, G., Brito, K.L., Bulteau, L., Dias, U., Dias, Z.: Sorting signed permutations by intergenic reversals. IEEE/ACM Transactions on Computational Biology and Bioinformatics **18**(6), 2870–2876 (2020)

90. Oliveira, A.R., Jean, G., Fertin, G., Brito, K.L., Dias, U., Dias, Z.: A 3.5-approximation algorithm for sorting by intergenic transpositions. In: International Conference on Algorithms for Computational Biology, pp. 16–28. Springer (2020)
91. Oliveira, A.R., Jean, G., Fertin, G., Brito, K.L., Dias, U., Dias, Z.: Sorting permutations by intergenic operations. IEEE/ACM Transactions on Computational Biology and Bioinformatics **18**(6), 2080–2093 (2021)
92. Pe'er, I., Shamir, R.: The median problems for breakpoints are NP-complete. Elec Colloq Comput Complexity **5**(71) (1998)
93. Pevzner, P., Tesler, G.: Transforming men into mice: The nadeau-taylor chromosomal breakage model revisited. In: Proc. 7th Ann. Int'l Conf. Computational Molecular Biology (RECOMB '03), pp. 247–256 (2003)
94. Qiu, W.G., Schutzer, S.E., Bruno, J.F., Attie, O., Xu, Y., Dunn, J.J., Fraser, C.M., Casjens, S.R., Luft, B.J.: Genetic exchange and plasmid transfers in Borrelia burgdorferi sensu stricto revealed by three-way genome comparisons and multilocus sequence typing. Proc Natl Acad Sci USA **101**(39), 14,150–14,155 (2004)
95. Raphael, B.J., Pevzner, P.A.: Reconstructing tumor amplisomes. Bioinformatics **20**(Suppl 1), 265–273 (2004)
96. San Mauro, D., Gower, D.J., Zardoya, R., Wilkinson, M.: A hotspot of gene order rearrangement by tandem duplication and random loss in the vertebrate mitochondrial genome. Mol Biol Evol **23**(1), 227–234 (2006)
97. Sankoff, D.: Edit distance for genome comparison based on non-local operations. In: Proc. 3rd Ann. Symp. Combinatorial Pattern Matching (CPM '92), *LNCS*, vol. 644, pp. 121–135 (1992)
98. Sankoff, D., Blanchette, M.: Multiple genome rearrangement and breakpoint phylogeny. J Comput Biol **5**(3), 555–570 (1998)
99. Shao, M.: `GREDU`. `https://github.com/shaomingfu/gredu` (2015)
100. Shao, M., Lin, Y., Moret, B.: An exact algorithm to compute the DCJ distance for genomes with duplicate genes. In: Proc. 18th Ann. Int'l Conf. Computational Molecular Biology (RECOMB '14), *LNCS*, vol. 8394, pp. 280–292 (2014)
101. Stoye, J., Wittler, R.: A unified approach for reconstructing ancient gene clusters. IEEE/ACM Trans Comput Biol Bioinform **6**(3), 387–400 (2009)
102. Swenson, K.M., Simonaitis, P., Blanchette, M.: Models and algorithms for genome rearrangement with positional constraints. Algorithm Mol Biol **11**(1), 1–10 (2016)
103. Tannier, E., Bergeron, A., Sagot, M.F.: Advances on sorting by reversals. Discrete Appl Math **155**(6), 881–888 (2007)
104. Tannier, E., Sagot, M.F.: Sorting by reversals in subquadratic time. In: Proc. 15th Ann. Symp. Combinatorial Pattern Matching (CPM '04), *LNCS*, vol. 3109, pp. 1–13 (2004)
105. Tannier, E., Zheng, C., Sankoff, D.: Multichromosomal median and halving problems under different genomic distances. BMC Bioinformatics **10**(1), 1–15 (2009)
106. Tesler, G.: GRIMM: genome rearrangements web server. Bioinformatics **18**(3), 492–493 (2002)
107. Tesler, G., Yu, Y., Pevzner, P.: `GRIMM`. `http://grimm.ucsd.edu/GRIMM/` (2002)
108. Véron, A.S., Lemaitre, C., Gautier, C., Lacroix, V., Sagot, M.F.: Close 3D proximity of evolutionary breakpoints argues for the notion of spatial synteny. BMC Genomics **12**(1), 1–13 (2011)
109. Volff, J.N., Altenbuchner, J.: A new beginning with new ends: linearisation of circular chromosomes during bacterial evolution. FEMS Microbiol Lett **186**(2), 143–150 (2000)
110. Wang, L.S., Warnow, T., Moret, B.M.E., Jansen, R.K., Raubeson, L.A.: Distance-based genome rearrangement phylogeny. J Mol Evol **63**(4), 473–483 (2006)
111. Watterson, G., Ewens, W., Hall, T., Morgan, A.: The chromosome inversion problem. J Theor Biol **99**(1), 1–7 (1982)
112. Wittler, R.: `Roci`. `http://bibiserv.techfak.uni-bielefeld.de/roci` (2004)

113. Xu, A.W., Sankoff, D.: Decompositions of multiple breakpoint graphs and rapid exact solutions to the median problem. In: Proc. 8th Int'l Workshop Algorithms in Bioinformatics (WABI '08), *LNCS*, vol. 5251, pp. 25–37 (2008)
114. Yancopoulos, S., Attie, O., Friedberg, R.: Efficient sorting of genomic permutations by translocation, inversion and block interchange. Bioinformatics **21**(16), 3340–3346 (2005)
115. Zhang, M., Arndt, W., Tang, J.: An exact solver for the DCJ median problem. In: Proc. Pacific Symp. on Biocomputing (PSB '09), pp. 138–149 (2009)
116. Zhao, H., Bourque, G.: `EMRAE`. `http://www.gis.a-star.edu.sg/~bourque/software.html` (2009)
117. Zhao, H., Bourque, G.: Recovering genome rearrangements in the mammalian phylogeny. Genome Res **19**(5), 934–942 (2009)
118. Zheng, C., Sankoff, D.: On the pathgroups approach to rapid small phylogeny. BMC Bioinformatics **12**(1), 1–9 (2011)
119. Zheng, C., Sankoff, D.: `Pathgroups`. `http://albuquerque.bioinformatics.uottawa.ca/lab/software.html` (2011)