Bioinformatics
doi.10.1093/bioinformatics/xxxxxx
Advance Access Publication Date: Day Month Year
Manuscript Category

OXFORD

Subject Section

Sorting cancer karyotypes using double-cut-and-joins, duplications and deletions

Ron Zeira and Ron Shamir [1,*]

[1] Blavatnik School of Computer Science, Tel Aviv university, Tel Aviv, 6997801, Israel

* To whom correspondence should be addressed.

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Abstract

Motivation: Problems of genome rearrangement are central in both evolution and cancer research. Most genome rearrangement models assume that the genome contains a single copy of each gene and the only changes in the genome are structural, i.e., reordering of segments. In contrast, tumor genomes also undergo numerical changes such as deletions and duplications, and thus the number of copies of genes varies. Dealing with unequal gene content is a very challenging task, addressed by few algorithms to date. More realistic models are needed to help trace genome evolution during tumorigenesis.
Results: Here we present a model for the evolution of genomes with multiple gene copies using the operation types double-cut-and-joins, duplications and deletions. The events supported by the model are *reversals*, *translocations*, *tandem duplications*, *segmental deletions*, and *chromosomal amplifications* and *deletions*, covering most types of structural and numerical changes observed in tumor samples. Our goal is to find a series of operations of minimum length that transform one karyotype into the other. We show that the problem is NP-hard and give an integer linear programming formulation that solves the problem exactly under some mild assumptions. We test our method on simulated genomes and on ovarian cancer genomes. Our study advances the state of the art in two ways: It allows a broader set of operations than extant models, thus being more realistic, and it is the first study attempting to reconstruct the full sequence of structural and numerical events during cancer evolution.
Availability: Code and data are available in https://github.com/Shamir-Lab/Sorting-Cancer-Karyotypes
Contact: ronzeira@post.tau.ac.il, rshamir@tau.ac.il
Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

During cancer, the tumor genome rapidly accumulates somatic mutations. While some mutations are small, affecting one or a few bases, other are large-scale events. Here we focus on the latter. They include inversions, chromosomal translocations, tandem duplications, segmental deletions, and whole chromosome amplifications or losses (Vogelstein *et al.*, 2013). Some cancer types are predominately characterized by these types of mutations (Ciriello *et al.*, 2013). Understanding these changes can assist in predicting disease progression and the outcome of medical interventions (Fielding, 2010). For instance, early translocations and tandem duplications in ovarian cancer were shown to contribute to drug sensitivity and clonal expansion (Ng *et al.*, 2012).

1.1 Aberration types and cancer genome data

The *copy number* (*CN*) of a genomic segment is the number of copies of the segment a genome contains. In a healthy diploid genome, each segment has $CN = 2$. In a *segmental deletion*, a segment of the DNA is deleted resulting in a genome with one less copy of the segment. In *chromosomal deletion*, an entire chromosome is deleted. In a *tandem duplication*, a segment of a chromosome is duplicated and inserted right after the original one. A *chromosomal duplication* (or *amplification*) creates an additional copy of an entire chromosome. Overall, deletions and duplications can change both the structure and the CN of the genome.

Other aberrations change only the structure of the genome but not its CNs. In an *inversion* (or *reversal*), a segment of a chromosome is reversed relative to its original orientation. In a *translocation*, two chromosomes exchange ends segments.

Given the germline genome G and the tumor genome T, a *breakpoint* is a position between two bases that are consecutive in G but not in T. Inversions and translocations introduce two breakpoints, segmental deletions and tandem duplications introduce one breakpoint, and chromosomal duplications/deletions introduce no breakpoints.

The primary source of data for cancer genome analysis today is deep sequencing. It allows inference of CN changes based on read depth (Ding *et al.*, 2014), and facilitates inferring breakpoints in the genome, detecting structural variants and identifying rearrangements (Korbel *et al.*, 2007). If the mapped locations of the two ends of a paired-end read do not match the read length, the read is called *discordant* and suggests a breakpoint in the genome. The location and orientation of such discordant reads can help detect the type of event (Abo *et al.*, 2014). Accurate reconstruction of the numerical and structural variations from deep sequencing data remains a challenge, and a myriad of computational methods have been devised for this task (Ding *et al.*, 2014).

1.2 Genome rearrangement models

Over the past two decades, many genome rearrangement models were studied. The classical model seeks a shortest sequence of inversions and translocations that transform one genome into another (Hannenhalli and Pevzner, 1995b,a). Such a sequence is called a *sorting scenario*. Later, a simpler model based on *double-cut-and-join* (*DCJ*) was proposed. In a DCJ, the genome is cut in two locations and the four loose ends are reconnected as two pairs. This model can represent both inversions and translocations (Yancopoulos *et al.*, 2005; Bergeron *et al.*, 2006). Feijão and Meidanis (2011) provided a simpler model called *single-cut-or-join* (*SCoJ*), in which every operation either cuts the genome or joins two loose ends. These DCJ and SCoJ models assumed a single copy of each genomic segment and no operation that alters copy number. Extant models with multiple segment copies often result in NP-hard problems (Tannier *et al.*, 2009; Shao and Lin, 2012). Some rearrangement models assume that a breakpoint cannot be used twice in a sorting scenario (Pevzner and Tesler, 2003).

Several models have addressed multiple copies along with other operation types. Some allow insertions or deletions of genomic segments along with DCJs, but only for non-duplicated segments (da Silva *et al.*, 2012). For sorting multiple copy genomes using DCJs only, both an exact integer linear program (ILP) and an approximation have been given (Shao *et al.*, 2015; Shao and Lin, 2012). Bader (2010) provided a heuristic for sorting by DCJs, duplications, and deletions. Shao and Moret (2015) devised an ILP for sorting genomes with multiple copies via DCJs and certain type of segmental duplications. Zeira and Shamir (2017) gave a linear algorithm for sorting with SCoJs and chromosomal duplications on genomes with at most two copies. Ozery-Flato and Shamir (2009) studied a model with certain duplications, deletions and SCoJs, and provided a 3-approximation algorithm that performed well on cancer genomes.

Several models attempted to introduce CN-modifying operations. Chowdhury *et al.* (2014) defined an edit distance between CN profiles obtained from FISH, where the edit operations are amplification or deletion of single genes, single chromosomes, or the whole genome. However, these methods are tailored to FISH data with a limited number of genes. Schwarz *et al.* (2014) introduced a model that allows amplifications and deletions of contiguous segments. A linear time algorithm for this edit distance was later given (Zeira *et al.*, 2017). However, all these models consider only CN modifications but not structural rearrangements.

1.3 Graph models for tumor rearrangements

Graph theory contributed remarkably to the area of genomic rearrangements. Breakpoint graphs are widely used for representation and analysis of rearranged genomes in evolution (Bafna and Pevzner, 1993; Hannenhalli and Pevzner, 1995b) and in cancer genomes (Raphael *et al.*, 2003). Greenman *et al.* (2012) created models that expanded the breakpoint graph and they used them in order to infer some order over tumor mutations.

Oesper *et al.* (2012) further expanded the breakpoint graph with a structure called the *interval adjacency graph*, which represents breakpoints, discordant reads and CN information. Their method, called *PREGO*, uses the number of reads supporting each edge to resolve the CN of genomic segments and identify discordant adjacencies in the tumor genome. Decomposition of this graph into a set of paths corresponds to a set of chromosomes. PREGO was shown to efficiently identify complex rearrangement in ovarian cancer sequencing data. Eitan and Shamir (2017) expanded this model and tested it in extensive simulations and on real cancer data.

1.4 Our contribution

We propose here a model for the structural and numerical changes that a genome with multiple segmental copies undergoes. The allowed operations are DCJs, tandem duplications, segmental deletions and whole chromosome duplications and deletions. This model encompasses many of the common aberrations in cancer, and does not preclude breakpoint reuse. However, we restrict both duplications and deletions to simple paths that include at most one copy of each segment. Similarly to Oesper *et al.* (2012), genomes are represented by the copy number of each segment and the adjacencies between them. Our goal is to find a shortest series of operations that transform one genome into the other, e.g., a normal genome to an observed tumor genome. Unlike most models, we focus here on finding the actual sequence of events. We show that the problem is NP-hard, give an ILP formulation for solving this problem and apply it on simulated and ovarian tumor data. The algorithm is able to resolve the sequence of events for tumors of average complexity.

The study advances the state of the art in genome rearrangement analysis in cancer in two ways: It allows a broader set of operations than extant models, thus being more realistic, and it is also the first model attempting to reconstruct the full sequence of structural and numerical events during cancer evolution.

2 Methods

In this section we present our model and formulate an ILP to solve it.

2.1 Notation

A genome contains a set $\mathcal{G} = \mathcal{G}^* \cup \mathcal{T}_\mathcal{G}$ of entities. $\mathcal{G}^*$ is a set of n *genes*, denoted $1, \ldots, n$. Each gene $g \in \mathcal{G}$ has two *extremities*, a *head* g^h and a *tail* g^t. W.l.o.g., denote $g^t = 2g$ and $g^h = 2g + 1$ for every $g \in \mathcal{G}$. $\mathcal{T}_\mathcal{G}$ is a set of special genes called *telomeres*. Each telomere has only one extremity. Telomeres come in pairs distinguished as the *left telomere* and the *right telomere*. A left telomere has only a head and a right telomere has only a tail. The left and right telomeres correspond to the start and end of chromosomes in the real genome. The genes in $\mathcal{G}^*$ are also called *internal* genes.

Denote by $\mathcal{T}$ the set of extremities corresponding to telomeres, and by $\mathcal{E}^* = \{g^t, g^h | g \in \mathcal{G}^*\}$ the set of extremities of internal genes. The set of all extremities is denoted by $\mathcal{E} = \mathcal{E}^* \cup \mathcal{T}$. Throughout, for an extremity e we shall denote by $g(e)$ the gene it belongs to.

A *karyotype* is represented by a pair $K = (cn, adj)$. $cn : \mathcal{G} \to \mathbb{N}$ is a gene *copy number profile* and $adj : \mathcal{E} \times \mathcal{E} \to \mathbb{N}$ is an *adjacency copy number matrix*, such that $\forall e \in \mathcal{E}, cn(g(e)) = \sum_{e' \in \mathcal{E}} adj(e, e')$. Notice that adj is symmetric, and different copies of a gene or adjacency are indistinguishable.

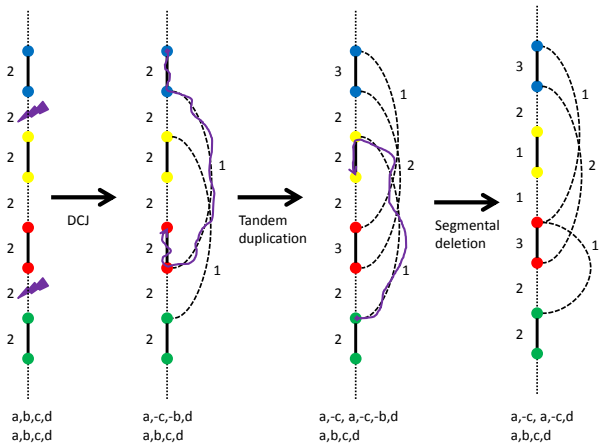

Fig. 1. The effect of model operations on the karyotype graph. Solid edges: interval edges; dotted edges: adjacencies; numbers on edges: CN; lightning signs: breakpoints. The scribbled lines show the path affected by the operation. The sequences of genes corresponding to possible chromosomes are shown below each graph, with each chromosome in a separate line. The genes a, b, c, d correspond to the graph segments from top to bottom of colors blue, yellow, red and green, respectively.

The *karyotype graph* of K is a weighted undirected graph $G = (\mathcal{E}, E, W)$ akin to the interval adjacency graph (Oesper *et al.*, 2012) (see Figure 1). The edge set $E = E_I \cup E_A$ consists of *interval edges* E_I and *adjacency edges* E_A. Interval edges $E_i = \{(g^t, g^h) | g \in \mathcal{G}^*, cn(g) > 0\}$ correspond to genes and adjacency edges $E_i = \{(u, v) | u, v \in \mathcal{E}, adj(u, v) > 0\}$ correspond to adjacencies in the karyotype. The weight $W : E \to \mathbb{N}$ is defined as the CN of the edge, i.e.,

$$W(u,v) = \begin{cases} cn(g) & \text{if } (u,v) = (g^t, g^h) \in E_I \\ adj(u,v) & \text{if } (u,v) \in E_A \end{cases}$$

Removing a copy of an existing edge (u, v) from G results in a new graph in which the CN of (u, v) is lower by 1 and the edge is deleted from the graph if its new CN is zero. Similarly, *adding a copy* of an edge (u, v) to G results in a new graph $G' = (\mathcal{E}, E', W')$ in which the CN of the edge (u,v) increases by 1 if it exists in G, or adding the new edge (u, v) with $CN = 1$.

An *alternating path* is a simple path in G in which odd edges are interval edges and even edges are adjacency edges, or vise-versa.

A *chromosome* in a karyotype is an alternating path starting and ending with telomeres. Note that a karyotype may be decomposable into chromosomes in several ways.

2.2 Model

We now turn to define the possible operations that alter a karyotype in our model (compare Figure 1).

Double-cut-and-join (DCJ): A *DCJ* operation selects two adjacency edges $(a, b), (c, d)$, removes a copy of each from the graph, and adds copies of new edges by joining two loose ends, either $(a, c), (b, d)$ or $(a, d), (b, c)$. In order to support splitting of a chromosome and introducing additional telomere copies, we also allow a special case of DCJ called *Single-cut-and-join* (*SCJ*). SCJ cuts an existing adjacency (a, b) and connects each loose end to a new telomere copy $t_1, t_2 \in \mathcal{T}$. The result of this SCJ is $(a, t_1), (b, t_2)$, i.e., splitting the adjacency into two separate chromosomes capped with new copies of telomeres t_1, t_2. Note that SCJs create new telomere copies that may or may not be part of the final karyotype. The identity of each t_i can be arbitrary chosen.

Tandem duplication: Let $v_1, \ldots, v_{2m}$ be an alternating path without telomeres starting with an interval edge. A *tandem duplication* adds edge copies for each edge in the path and adds another adjacency edge copy (v_1, v_{2m}). We call v_1, v_{2m} the *anchors* of the duplication. In terms of the sequence, this operation corresponds to $\ldots g_0 \underline{g_1 g_2 \ldots g_{m-1} g_m} g_{m+1} \ldots \to \ldots g_0 \underline{g_1 g_2 \ldots g_{m-1} g_m} \underline{g_1 g_2 \ldots g_{m-1} g_m} g_{m+1} \ldots$ where g_i is the gene corresponding to node v_{2i-1}.

Chromosome duplication: Let $t_0, v_1, \ldots, v_{2m}, t_{2m+1}$ be an alternating path such that t_0, t_{2m+1} are telomeric extremities and (v_1, v_2) is an interval edge. A *chromosome duplication* adds edge copies for each edge along the path and increases the CN of the two telomeres by one.

Segmental deletion: Let $v_1, \ldots, v_{2m}$ be an alternating path without telomeres staring with an adjacency edge. A segmental deletion removes edge copies for each edge along the path and adds an adjacency edge copy (v_1, v_{2m}). We call v_1, v_{2m} the *anchors* of the deletion. In terms of the sequence, this operation corresponds to $\ldots g_0 g_1 g_2 \ldots g_{m-1} g_m g_{m+1} \ldots \to \ldots g_0 g_1 g_m g_{m+1} \ldots$.

Chromosome deletion: This is a special case of segmental deletion where v_1 and v_{2m} are telomeric nodes. We do not add the edge (v_1, v_{2m}) and thus it corresponds to deleting an entire chromosome with its telomeres.

The Karyotype Sorting Problem: The input is S, T, d, where $S = (s\_cn, s\_adj)$ is a source karyotype, $T = (t\_cn, t\_adj)$ is a target karyotype, and d is an integer. Our goal is to find a shortest series of $\leq d$ operations transforming S into T, or declare that no such sequence exists. An example of a series of operations of length 11 is given in Figure 3.

Notice that the sorting problem is not symmetric. Moreover, there may be a sorting scenario from S to T, but not from T to S if, for example, T has lost all copies of some segment in S. New material can be gained in the model by duplications (tandem and chromosomal). Telomeres can also be gained by SCJs.

Theorem 1. The karyotype sorting problem is NP-hard.

Proof. Let $G = (V, E)$ be a directed graph with n nodes in which all in- and out-degrees are 2. Deciding if a such a graph contains a Hamiltonian cycle is NP-hard (Plesnik, 1979). Let $(y, x) \in E$ be some edge. Deciding if there is a Hamiltonian path from x to y in G is still NP-hard. We assume w.l.o.g. that G is strongly connected, since otherwise it would not contain a Hamiltonian path from x to y. Notice that in that case G is also Eulerian.

We construct a source karyotype S as follows: for each node in $v \in V$ we create a gene g_v with CN=2 and for each $(u, v) \in E \setminus \{(y, x)\}$ we add one copy of the adjacency (g_u^h, g_v^t). In addition, we add two genes w, z with CN=1 and connect them with adjacencies $(w^h, g_x^t), (g_y^h, z^t)$ of CN=1. To make it a valid karyotype, we add a left and right telomeres t_1, t_2 and connect them to w^t and z^h respectively. In other words, S is a karyotype with a single chromosome $t_1, w, x, P_e, y, z, t_2$, where P_e corresponds to some Eulerian path from x to y in G. Our target karyotype T would be composed of n single gene chromosomes of the form t_1, g_v, t_2 and an additional chromosome t_1, w, z, t_2. Namely, each gene will have CN=1, for telomeres CN=$n + 1$, and the adjacencies would be of the form (t_1, g_v^t) and (g_v^t, t_2) for $v \in V$ plus $(t_1, w^t), (w^h, z^t), (z^h, t_2)$. Notice that all chromosome paths start and end with the same telomeres t_1, t_2. We will show that there is a sorting scenario of S to T of length $n + 1$ if and only if G admits a Hamiltonian path from x to y.

Suppose G contains a Hamiltonian path $P = x, v_2, \ldots, v_{n-1}, y$. To construct a sorting scenario, first perform n SCJs for each adjacency of the form (g_u^h, g_v^t) that is not part of P, and connect them as (t_1, g_v^t) and (g_u^t, t_2). Now, perform a segmental deletion of the path $g_x, g_{v_2}, \ldots, g_{v_{n-1}}, g_y$ connecting w^h and z^t. The total length of the sorting scenario is $n + 1$.

For the other direction, suppose there is a sorting scenario with $n + 1$ operations from S to T. Since each gene in T has CN=1, the scenario

must contain at least one deletion. Notice that T has n additional copies of the telomeres and the only way to increase the CN of telomeres in the model is either by chromosome duplication or by SCJ. However, each chromosome duplication would require additional deletions to reduce the gene CN to 1 in T. We conclude that the sorting scenario must contain n SCJs and one deletion covering all genes g_v. Since w and z are adjacent in T, the segmental deletion must be anchored at w^h and z^t. Denote the path of this deletion $P = g_x, g_{v_2}, \ldots, g_{v_{n-1}}, g_y$. The corresponding path $P' = x, v_2, \ldots, v_{n-1}, y$ is a Hamiltonian path in G. □

2.3 ILP formulation

We present an ILP formulation for the karyotype sorting problem. The formulation describes $d+1$ karyotype graphs $G^0, \ldots G^d$ corresponding to the genome after each operation. G^0 is set to S and $G^d = T$. The formulation guarantees that difference between consecutive graphs corresponds to one valid operation of the model.

2.3.1 Variables

We define integer variables for each G^k, as follows. For every $k \in [0, d]$ and every $i \in \mathcal{G}$ let $\mathbf{cn_i^k} \in \mathbb{N}$ be the variable for the CN of gene i after k operations. By definition, $cn_i^0 = s\_cn_i$ and $cn_i^d = t\_cn_i$ for every $i \in \mathcal{G}$. Similarly, for every $k \in [0, d]$ and every $u, v \in \mathcal{E}$ let $\mathbf{adj_{u,v}^k}$ be the CN of a adjacency edge (u, v) after k operations. By definition, $adj_{u,v}^0 = s\_adj_{u,v}$ and $adj_{u,v}^d = t\_adj_{u,v}$ for every $u, v \in \mathcal{E}$.

Now, we define binary variables for each type of operation. For every $k \in [0, d]$ and every $u, v \in \mathcal{E}$ let $\mathbf{cut_{u,v}^k} \in \{0, 1\}$ be an indicator variable for cutting the interval adjacency between u and v in the k'th operation. Similarly, $\mathbf{join_{u,v}^k}$ is an indicator variable for joining the two extremities u and v in the k'th operation. By convention, the cut and join are not symmetric, in order to support cutting or joining the same adjacency twice. An SCJ is a DCJ with one existing adjacency and an implicit adjacency of telomeres. To support SCJ operations, binary variables $\mathbf{addTel_{t_1,t_2}^k}$ are introduced for every two telomeres $t_1 \leq t_2 \in \mathcal{T}$. $addTel_{t_1,t_2}^k = 1$ means that new copies of telomeres $t_1, t_2 \in \mathcal{T}$ are created.

For every $k \in [0, d]$ and every $u \leq v \in \mathcal{E}^*$ let $\mathbf{ampAnch_{u,v}^k} \in \{0, 1\}$ be an indicator variable for a tandem duplication starting at u and ending at v in the k'th operation. In addition, the variable $\mathbf{ampAnch_{t_1,t_2}^k} \in \{0, 1\}$ for $t_1 \leq t_2 \in \mathcal{T}$ indicates a chromosome amplification for the chromosome starting and ending at telomeres t_1 and t_2 respectively. Let $\mathbf{ampGene_i^k} \in \{0, 1\}$ be an indicator variable that gene i (i.e., the interval edge (i^t, i^h)) is a part of the duplicated segment, and let $\mathbf{ampAdj_{u,v}^k} \in \{0, 1\}$ be an indicator variable that the adjacency edge (u, v) is a part of the duplicated segment.

Similarly, for every $k \in [0, d]$ and every $u \leq v \in \mathcal{E}$, $\mathbf{delAnch_{u,v}^k} \in \{0, 1\}$ is an indicator variable for a deletion starting at u and ending at v in the k'th operation. $\mathbf{delGene_i^k} \in \{0, 1\}$ is an indicator variable that gene i is a part of the deleted segment, and $\mathbf{delAdj_{u,v}^k} \in \{0, 1\}$ is an indicator variable that the adjacency edge (u, v) is a part of the deleted segment.

2.3.2 Constraints

We now describe the ILP constraints for each stage $0 \leq k \leq d-1$. We will describe constraints for each type of operation and general constraints for updating the karyotype graph.

Updating the karyotypes: The CN of a non-telomeric gene $i \in \mathcal{G}^*$ is increased by amplifications and decreased by deletions:

$$cn_i^{k+1} = cn_i^k + ampGene_i^k - delGene_i^k$$

For telomeric gene $i \in \mathcal{T}_\mathcal{G}$ with a corresponding extremity $t \in \mathcal{T}$, the CN can increase if new copies of the telomere are introduced via SCJ. An SCJ can add either two copies of the same telomere or one copy of two telomeres:

$$cn_i^{k+1} = cn_i^k + ampGene_i^k - delGene_i^k + 2addTel_{t,t}^k + \sum_{t' \neq t \in \mathcal{T}} addTel_{t,t'}^k$$

Updating adjacency CNs for internal nodes $u \neq v \in \mathcal{E}^*$:

$$\begin{aligned} adj_{u,v}^{k+1} = adj_{u,v}^k + ampAnch_{u,v}^k + ampAdj_{u,v}^k + delAnch_{u,v}^k \\ - delAdj_{u,v}^k - cut_{u,v}^k - cut_{v,u}^k + join_{u,v}^k + join_{v,u}^k \end{aligned} \quad (1)$$

In words, we increase the CN of the adjacency if it is either, the anchor of a duplication, along the duplication path, the anchor of a deletion or its ends are joined. The adjacency CN is decreased if it is along a deletion path or it is cut. The cut and join variable can decrease or increase the CN by at most 2 if the same adjacency is used twice in a DCJ.

Updating adjacency CN for loop edges (u, u) is similar to (1), but uses only single $cut_{u,u}^k, join_{u,u}^k$ variables.

For telomere $t \in \mathcal{T}$ and internal node $v \in \mathcal{E}^*$ we update the CN as follows:

$$\begin{aligned} adj_{t,v}^{k+1} = adj_{t,v}^k + ampAdj_{t,v}^k + delAnch_{t,v}^k - delAdj_{t,v}^k \\ - cut_{t,v}^k + join_{t,v}^k \end{aligned}$$

In addition, we require that in each stage k there will be at most one operation:

$$\begin{aligned} \frac{1}{2} \sum_{u,v \in \mathcal{E}} cut_{u,v}^k + \sum_{u \leq v \in \mathcal{E}^*} ampAnch_{u,v}^k + \sum_{u \leq v \in \mathcal{E}} delAnch_{u,v}^k \\ + \sum_{t_1 \leq t_2 \in \mathcal{T}} ampAnch_{u,v}^k \leq 1 \end{aligned}$$

DCJs: An adjacency cannot be cut more times than its CN. Therefore, for every $u \in \mathcal{E}, v \in \mathcal{E}$ and $u \neq v$:

$$cut_{u,v}^k + cut_{v,u}^k \leq adj_{u,v}^k$$

Similarly, for adjacencies of the form (u, u): $cut_{u,u}^k \leq adj_{u,u}^k$.

For each adjacency $u \in \mathcal{E}$, the number of cuts equals to the number of joins:

$$\sum_{v \in \mathcal{E}} cut_{u,v}^k + \sum_{v \in \mathcal{E}} cut_{v,u}^k = \sum_{v \in \mathcal{E}} join_{u,v}^k + \sum_{v \in \mathcal{E}} join_{v,u}^k$$

For every pair of telomeres $t_1 \leq t_2 \in \mathcal{T}$, SCJ introduces an explicit adjacency (t_1, t_2) which is cut immediately as part of a DCJ:

$$cut_{t_1,t_2}^k = addTel_{t_1,t_2}^k$$

In addition, if $t_1 < t_2$, set $cut_{t_2,t_1}^k = 0$. We also restrict that at most one pair of telomere copies is introduced as an SCJ in every stage:

$$\sum_{t1 \leq t_2 \in \mathcal{T}} addTel_{t_1,t_2}^k \leq 1$$

Amplifications: A gene $i \in \mathcal{G}$ cannot be amplified if it has $CN = 0$:

$$ampGene_i^k \leq cn_i^k \quad (2)$$

Similarly, an adjacency $u, v \in \mathcal{E}$ can only be amplified if it has a positive CN:

$$ampAdj_{u,v}^k \leq adj_{u,v}^k \quad (3)$$

For every internal node $u \in \mathcal{E}^*$, its corresponding gene is amplified if and only if one of its adjacencies is amplified or it is an anchor of an

amplification:

$$ampGene^k_{g(u)} = \sum_{v \in \mathcal{E}^*} ampAnch^k_{u,v} + \sum_{v \in \mathcal{E}} ampAdj^k_{u,v}.$$

A telomere $t \in \mathcal{T}$ can only be involved in whole chromosome duplications. Therefore, the telomere is amplified iff it is an anchor iff one of its adjacencies is amplified:

$$ampGene^k_{g(t)} = \sum_{t' \in \mathcal{T}} ampAnch^k_{t,t'} = \sum_{v \in \mathcal{E}^*} ampAdj^k_{t,v}$$

Enforcing path connectivity: One problem with this formulation is that in addition to the amplification path, we may get a collection of disjoint cycles composed of alternating interval and adjacency edges with their corresponding variables $ampGene^k_{g(u)}$,$ampAdj^k_{u,v}$ set to one. For example, consider $S = 1, 1, 2, 2$ and $T = 1, 1, 1, 2, 2, 2$. To get from S to T we need to do two tandem duplications of the genes 1 and 2. However, according to the current formulation, this can be done in one step by assigning $ampGene^0_1 = 1, ampGene^0_2 = 1, ampAnch^0_{1^t,1^h} = 1, ampAdj^0_{2^t,2^t} = 1$.

To force the alternating path of the amplification to be connected, we add flow-like constraints (Bruckner *et al.*, 2010). Suppose $q > r$ are the anchors of the amplification and denote r as the *sink*. Each node along the path from q to r (excluding r) will be as a source of one unit of flow, and we require that all flow will eventually be drained at r. This enforces the connectivity of the path.

Let $-2(n+1) \leq f^k_{u,v} \leq 2(n+1)$ be an integer variable for the directed amount of flow from $u \in \mathcal{E}$ to $v \in \mathcal{E}$. Let $ampNodes^k = 2\sum_{i \in \mathcal{G}} ampGene^k_i$ be an integer variable for the number nodes that are amplified along the path. We seek $ampNodes^k - 1$ source nodes, each providing one unit of flow, and one sink that drains the $ampNodes^k - 1$ units of flow. Let $sink^k_v = \sum_{u>v} ampAnch^k_{v,u}$ be a binary variable indicating that v is a sink. We have the following constraints:

The flow is antisymmetric: $f^k_{u,v} = -f^k_{v,u}$.

An edge can contain flow only if it is amplified: $f^k_{u,v} \leq 2(n+1)ampAdj^k_{u,v}$ if u, v are not from the same gene, and $f^k_{u,v} \leq 2(n+1)ampGene^k_i$ if u, v are nodes of gene i.

Production and conservation of flow in every node $u \in \mathcal{E}$:

$$\sum_v f^k_{u,v} = ampGene^k_{g(u)} - ampNodes^k \cdot sink^k_u$$

By this constraint, if u is not part of an amplification path, we have $\sum_v f^k_{u,v} = 0$. If u is part of the amplification path, but not the sink, we have $\sum_v f^k_{u,v} = 1$, i.e., u adds one unit to the flow. If u is part of the amplification path and the sink, we have $\sum_v f^k_{u,v} = 1 - ampNodes^k$ and it drains all the flow.

Since the term $ampNodes^k \cdot sink^k_u$ is not linear we introduce a new non-negative integer variable $product^k_u$ such that $product^k_u = ampNodes^k \cdot sink^k_u$ using the following constraints:

$$product^k_u \leq ampNodes^k$$

$$product^k_u \leq 2(n+1) \cdot sink^k_u$$

$$ampNodes^k - 2(n+1)(1 - sink^k_u) \leq product^k_u$$

If $ampNodes^k = 0$ or $sink^k_u = 0$ then the first two constraints force $product^k_u = 0$, otherwise $product^k_u \leq ampNodes^k$. If $sink^k_u = 1$, we have $ampNodes^k \leq product^k_u$ and therefore $product^k_u = ampNodes^k$.

Deletions: Genes or adjacencies cannot be deleted if they have a CN of zero. Therefore, we add constraint similar to 2 and 3 using $delGene^k_i$ and $delAdj^k_{u,v}$ instead.

For every internal node $u \in \mathcal{E}^*$, one of its adjacencies is deleted if and only if it is an anchor of a deletion or its gene is deleted:

$$\sum_{v \in \mathcal{E}} delAdj^k_{u,v} = delGene^k_{g(u)} + \sum_{v \in \mathcal{E}^*} delAnch^k_{u,v}$$

A telomere $t \in \mathcal{T}$ can be deleted only if it is part of a whole chromosome deletion:

$$delGene^k_{g(t)} = \sum_{t' \in \mathcal{T}} delAnch^k_{t,t'}$$

If a telomere $t \in \mathcal{T}$ is an anchor of a segmental deletion then one of its adjacencies must be deleted:

$$\sum_{v \in \mathcal{E}} delAnch^k_{t,v} = \sum_{v \in \mathcal{E}} delAdj^k_{t,v}$$

As for amplifications, we add flow constraints to guarantee that the deletion path is connected.

2.3.3 Objective function

Our goal is to minimize the number of amplifications, deletions and DCJs:

$$\min \sum_{1 \leq k \leq d} \left(\sum_{t_1 \leq t_2 \in \mathcal{T}} ampAnch^k_{t_1,t_2} + \sum_{u \leq v \in \mathcal{E}^*} ampAnch^k_{u,v} + \sum_{u \leq v \in \mathcal{E}} delAnch^k_{u,v} + \frac{1}{4} \sum_{u,v \in \mathcal{E}} [cut^k_{u,v} + join^k_{u,v}] \right)$$

DCJs have two cuts and two joins and therefore contribute one to the objective function. The objective function can be modified to give different weights to different operations, and even to specific events, e.g., amplifying regions with oncogenes.

2.3.4 Complexity

Overall, the ILP formulation has $O(dn^2)$ variables and $O(dn^2)$ constraints. We can relax the integrality constraints for all non-binary variables since each of them is constrained to be a sum of binary variables.

Since we do not know the optimal value d^* of d, we can perform either a binary or sequential search on d. If there is no feasible solution for some d, we increase d. If $d \geq d^*$, the ILP will find a solution with d^* operations since it can always add stages with no operations in them.

Each DCJ introduces two new adjacencies, while segmental deletions and tandem duplications introduce one new adjacency. A trivial lower bound on d is the number of breakpoints divided by two. A trivial upper bound is $d \leq \sum_{u \leq v \in \mathcal{E}} |s\_adj_{u,v} - t\_adj_{u,v}| + \sum_{i \in \mathcal{G}} |s\_cn_i - t\_cn_i|$. That is, changing the CN of each gene and each adjacency separately.

The ILP problem is NP-hard (Karp, 1972) and the runtime of ILP algorithms is not polynomially bounded. However, modern ILP solvers incorporate powerful heuristics and can handle many large-scale problems. ILP has been a powerful tool in formulating and solving other rearrangement models (Rahmann and Klau, 2006; Shao *et al.*, 2015; Shao and Moret, 2015).

3 Results

3.1 Simulations

To assess performance, we simulated tumor karyotypes and applied the algorithm to them. Here is an overview of the simulation: We start with a diploid karyotype S' with two identical chromosomes $1, \ldots, m$, and perform d operations to derive a tumor karyotype T'. We then compress

Table 1. The optimal number of events computed by the algorithm vs. the simulated number of events

simulated events	1	2	3	4	5	6	7	8
max	1	2	3	4	5	6	7	8
median	1	2	3	4	5	6	7	7
min	1	1	1	2	1	2	3	6

maximal identical segments in S' and T' into singletons. The resulting shorter source and target karyotypes are used as input to the algorithm.

Initially, each chromosome is represented by a sequence of $m = 100$ atomic segments. We perform a series of operations on the karyotype by applying duplications (tandem or chromosomal), deletions (segmental or chromosomal) and DCJs (reversals or translocations). Whole chromosome events are given low probability (5% each), while all other types are chosen uniformly at random. The span of segmental deletions, duplications and inversions was chosen at random and was limited to 30 units in order to avoid rapid loss of the middle segments.

In order to decrease the size of the karyotypes, we compress maximal identical sequences in S' and T'. That is, a simple path that appears in S' and T' is compressed if all interval and adjacency edges along the path have the same copy number, and nodes along the path have no other branching edges beside the path edges. The result is new karyotypes S and T with $n \leq m$ segments. This way, every segment in the compressed karyotypes must be involved in at least one breakpoint. Since all operations act on contiguous paths in the graph and all segments inside a compressed path are symmetric, we conjecture this procedure preserves the optimal distance. This compressed karyotype structure conforms with information provided by most assembly tools in which contiguous segments are determined by detecting breakpoints.

We simulated karyotypes with 3 to 8 operations. Eitan and Shamir (2017) observed based on the analysis of tumor samples from Malhotra *et al.* (2013) that the average number of operations observed in real deep sequencing cancer data was 5-8 per connected component. For each distance, 20 instances were simulated and the optimal distance was computed by the algorithm. In Table 1 we see that the computed distance is bounded by the simulated distance but can sometimes be shorter when d increases.

To test if the scenarios inferred are close to the simulated ones, we performed two additional comparisons, in terms of the types of operations and in terms of the actual operations. The results (Figures S1 and S2) show that the scenarios are quite similar. We also observed that the distance from the karyotype back to the diploid genome is usually lower than the distance from the diploid to the karyotype (Figure S3). This is because one can use a few deletions to get rid of a mutated chromosomes, and then create another copy of a normal chromosome in one operation.

Figure 2 shows the running time of the ILP algorithm as a function of the optimal distance it calculated. The time grows exponentially with the distance. The ILP was solved using Gurobi Optimizer 7.5 (Gurobi optimizer reference manual, Gurobi Optimization, 2018) on a shared Unix server with 72 2.3GHz cores and 800Gb of RAM.

3.2 Cancer karyotypes

We analyzed karyotypes from five ovarian cancer genomes that were sequenced as part of *TCGA* (Bell *et al.*, 2011) and were used in the analysis of PREGO (Oesper *et al.*, 2012). PREGO outputs CN per segment as well as for adjacencies based on the read coverage.

For each autosome in the genome, two telomeres were connected to the tail of the first segment and the head of the last segment with their copy numbers matching these segments (see Figure S7). In each sample, we

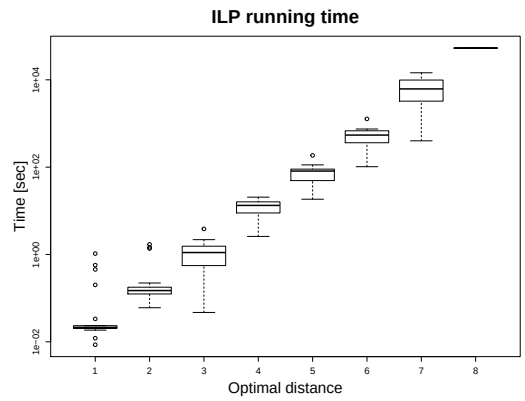

Fig. 2. ILP running time as a function of the optimal distance on simulated instances.

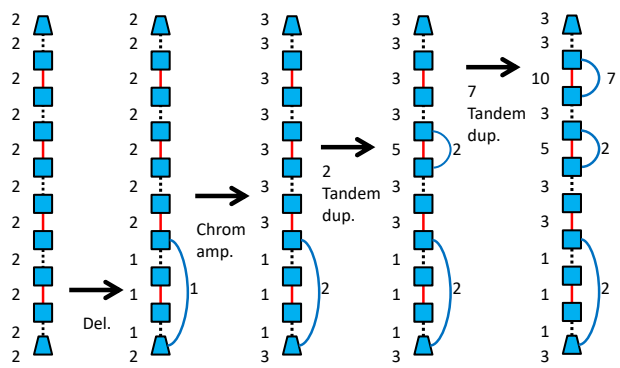

Fig. 3. Example of ovarian cancer sample OV4 chromosome 8 transformation from diploid (left) to tumor (right). Square nodes represent segment extremities and trapezoid nodes represent telomeres. Dotted edges correspond to adjacency edges, full straight edges correspond to interval edges and rounded edges correspond to novel adjacencies caused by the tumor process. The number next to each edge is its CN.

analyzed each connected component in the karyotype graph separately. For each such connected component, we calculated the distance from a matching diploid genome with the same subset of chromosomes. An example of the sequence of operations transforming OV4 chromosome 8 is shown in Figure 3.

To speed up the algorithm on real data we used several preprocessing steps. First, simple tandem duplications were removed from the karyotype and added to the distance. That is, for each $g \in \mathcal{G}$, we remove adj_{g^t,g^h} edges of the form (g^t, g^h) from the graph as they can only be a result of tandem duplications. We again compress the karyotypes after this step.

In addition, some chromosome components exhibit large repetitions of complex chromosomal structures that are not simple tandem duplications. In such cases, we search for the longest path from one telomere to another that repeats itself $k \geq 2$ times. Such a path corresponds to an amplified chromosomal structure and thus we remove $k - 1$ repetitions from the karyotype. We use the algorithm to calculate the edit distance of this path separately and add it $k - 1$ times to the total distance of the reduced graph.

We observed several examples of balanced (Figure S5) and unbalanced (Figure S4) translocations and also provide possible scenarios causing these phenomena. We also observed a breakage/fusion/bridge (*BFB*) cycle in chromosome 18 of OV2 (Figure S6). BFB cycles are a known source of genome instability (Greenman *et al.*, 2012). This aberrant chromosome is further amplified 7 times, and is part of a complex connected component with chromosomes 12 and 16 (Figure S7). Similar observations were also shown by Oesper *et al.* (2012) without addressing the operation sequence.

Table 2. Results of the algorithm on TCGA ovarian samples. The chromosomes involved in each component are shown within brackets. OV1: TCGA-13-0890; OV2: TCGA-13-0723; OV3: TCGA-24-0980; OV4: TCGA-24-1103; OV5: TCGA-13-1411.

Sample	Components	Distance	Sample	Components	Distance	Sample	Components	Distance
OV1	(16),(13)	1	OV1	(12,15)	3	OV1	(11,20)	5
OV1	(1,2,3,4,5,6,8,9,10,14,17,19)	NA	OV2	(8,20)	5	OV2	(3,4),(9)	6
OV2	(14,21)	10	OV2	(12,16,18)	26	OV2	(1,2,5,7,10,11,15,19,22)	NA
OV3	(13),(17),(21)	1	OV3	(9),(18)	2	OV3	(2)	6
OV3	(4,8)	7	OV4	(1),(13),(20),(21)	1	OV4	(18)	2
OV4	(3),(15)	6	OV4	(22)	4	OV4	(11)	10
OV4	(8),(9,12)	11	OV4	(5,10,16,19)	20	OV4	(2,4,6,7,14,17)	NA
OV5	(7),(16)	1	OV5	(1,3),(2,17),(9,10)	3	OV5	(12,21),(18)	4

Table 2 shows the distance calculated by the algorithm for each nontrivial connected component of the ovarian samples. The running time on the real karyotypes per connected component ranged from a few seconds for a simple component of distance at most 4, to a few hours for more complex components. In three cases, the algorithm did not find any feasible solution within 24 hours. These cases have very high CN or complex structural variations. All cases involve six or more interconnected chromosomes and contain interval CNs as high as 30.

4 Conclusions

In this study, we present a model for sorting a karyotype using deletions, amplifications, translocations and reversals. This model supports both structural and numerical alternations observed in cancer genomes. It focuses on finding a sequence of operations between two karyotypes and allows breakpoint reuse. We show the sorting problem is NP-hard and devise an ILP formulation that can find a shortest sequence of events that transform a normal into a tumor genome. We apply the algorithm on simulated karyotypes as well as real data of ovarian cancer. The algorithm is able to solve most components of the real tumor genomes.

The algorithm has limited applicability on highly complex karyotypes. As shown on simulated data (Figure 2), running time grows exponentially with the number of operations. Additional work on the ILP formulation may make the approach more practical. On the real karyotype data, the algorithm could not resolve a few extremely rearranged connected components of chromosomes. Nevertheless, typical cancer samples exhibit modest complexity, making this algorithm useful in the majority of real cases (Eitan and Shamir, 2017). Moreover, a highly rearranged karyotype could be a result of noisy read data, tumor heterogeneity or unmodeled global events. Better methods are needed to address these cases.

While the model addresses a relatively wide array of operations, it still has some limitations. For instance, our duplication and deletion operations are restricted to simple paths with a single copy of each segment. However, in some scenarios we may benefit from performing operations on non-simple paths. For example, for a single segment with m tandem repetitions, our model would require m tandem duplications, but only $\log m$ operations will suffice if we allow non-simple path duplications. Other events like non-tandem segmental duplications and BFB (Zakov *et al.*, 2013) are not included (but are expressible, e.g. Figure S6) in the model.

Our karyotypes are represented using their CNs and adjacencies, but this representation is not unique for a specific set of chromosomes. That is, there could be several chromosome sets that may give the same karyotype. Since we do not model chromosomes explicitly, some operations may be artificial and would not correspond to operations on sequences.

In order to apply our method on more cancer data, we intend to improve the running time further. Then, a more systematic employment of the algorithm on a larger set of karyotypes can reveal sequences of operations common to several tumors. In addition, the algorithm can derive a sequence of operations between two tumor genomes (for example, from different time points) and thus help understand the evolution of tumors.

Ultimately, we would like to represent the chromosomes themselves and perform all operations on them. The goal in this case would be to decompose the source and target karyotypes into chromosomes such that the number of operations between them is minimum. Nonetheless, it was recently argued that reconstruction of the exact cancer chromosomes remains a hard challenge (Eitan and Shamir, 2017).

Acknowledgements

We thank Layla Oesper for providing the preprocessed ovarian cancer data and Nimrod Rappoport for helpful comments.

Funding

Study supported in part by the Bella Walter Memorial Fund of the Israel Cancer Association and by Len Blavatnik and the Blavatnik Family foundation. RZ was supported by a fellowship from the Edmond J. Safra Center for Bioinformatics at Tel-Aviv University.

References

Abo, R. P. *et al.* (2014). BreaKmer: Detection of structural variation in targeted massively parallel sequencing data using kmers. *NAR*, **43**(3), e19–e19.

Bader, M. (2010). Genome rearrangements with duplications. *BMC Bioinformatics*, **11 Suppl 1**, S27.

Bafna, V. and Pevzner, P. (1993). Genome rearrangements and sorting by reversals. *Proc. FOCS*, pages 148–157.

Bell, D. *et al.* (2011). Integrated genomic analyses of ovarian carcinoma. *Nature*, **474**(7353), 609–15.

Bergeron, A. *et al.* (2006). A unifying view of genome rearrangements. In P. Bücher and B. M. Moret, editors, *Algorithms in Bioinformatics*, volume 4175 of *LNCS*, pages 163–173. Springer.

Bruckner, S. *et al.* (2010). Topology-free querying of protein interaction networks. *JCB*, **17**(3), 237–252.

Chowdhury, S. A. *et al.* (2014). Algorithms to model single gene, single chromosome, and whole genome copy number changes jointly in tumor phylogenetics. *PLoS Comp. Bio.*, **10**(7), e1003740.

Ciriello, G. *et al.* (2013). Emerging landscape of oncogenic signatures across human cancers. *Nat. Genet.*, **45**(10), 1127–1133.

da Silva, P. H. *et al.* (2012). Restricted DCJ-indel model: sorting linear genomes with DCJ and indels. *BMC Bioinformatics*, **13**(Suppl 19), S14.

Ding, L. *et al.* (2014). Expanding the computational toolbox for mining cancer genomes. *Nature Reviews Genetics*, **15**(8), 556–570.

Eitan, R. and Shamir, R. (2017). Reconstructing cancer karyotypes from short read data: the half empty and half full glass. *BMC Bioinformatics*, **18**(1), 488.

Feijão, P. and Meidanis, J. (2011). SCJ: a breakpoint-like distance that simplifies several rearrangement problems. *TCBB*, **8**(5), 1318–29.

Fielding, A. K. (2010). Current treatment of Philadelphia chromosome-positive acute lymphoblastic leukemia. *Haematologica*, **95**(1), 8–12.

Greenman, C. D. *et al.* (2012). Estimation of rearrangement phylogeny for cancer genomes. *Genome Research*, **22**(2), 346–61.

Gurobi optimizer reference manual, Gurobi Optimization (2018).

Hannenhalli, S. and Pevzner, P. A. (1995a). Transforming cabbage into turnip. In *Proc. STOC*, volume 46, pages 178–189, NY, NY, USA.

Hannenhalli, S. and Pevzner, P. A. (1995b). Transforming men into mice (polynomial algorithm for genomic distance problem). In *Proc. FOCS*, volume 36, pages 581–592.

Karp, R. M. (1972). *Reducibility among Combinatorial Problems*, pages 85–103. Springer US, Boston, MA.

Korbel, J. O. *et al.* (2007). Paired-end mapping reveals extensive structural variation in the human genome. *Science*, **318**(5849), 420–6.

Malhotra, A. *et al.* (2013). Breakpoint profiling of 64 cancer genomes reveals numerous complex rearrangements spawned by homology-independent mechanisms. *Genome research*, **23**(5), 762–76.

Ng, C. K. *et al.* (2012). The role of tandem duplicator phenotype in tumour evolution in high-grade serous ovarian cancer. *The Journal of Pathology*, **226**(5), 703–712.

Oesper, L. *et al.* (2012). Reconstructing cancer genomes from paired-end sequencing data. *BMC Bioinformatics*, **13 Suppl 6**(Suppl 6), S10.

Ozery-Flato, M. and Shamir, R. (2009). Sorting cancer karyotypes by elementary operations. *JCB*, **16**(10), 1445–60.

Pevzner, P. and Tesler, G. (2003). Human and mouse genomic sequences reveal extensive breakpoint reuse in mammalian evolution. *PNAS*, **100**(13), 7672–7.

Plesnik, J. (1979). The NP-completeness of the Hamiltonian cycle problem in planar digraphs with degree bound two. *Information Processing Letters*, **8**(4), 199–201.

Rahmann, S. and Klau, G. W. (2006). Integer Linear Programs for Discovering Approximate Gene Clusters. In *Proc. WABI*, volume 4175 of *LNCS*, pages 298–309. Springer Berlin Heidelberg.

Raphael, B. J. *et al.* (2003). Reconstructing tumor genome architectures. *Bioinformatics*, **19**(Suppl 2), ii162–ii171.

Schwarz, R. F. *et al.* (2014). Phylogenetic quantification of intra-tumour heterogeneity. *PLoS Comp. Bio.*, **10**(4), e1003535.

Shao, M. and Lin, Y. (2012). Approximating the edit distance for genomes with duplicate genes under DCJ, insertion and deletion. *BMC Bioinformatics*, **13**(Suppl 19), S13.

Shao, M. and Moret, B. M. E. (2015). Comparing genomes with rearrangements and segmental duplications. *Bioinformatics*, **31**(12), i329–i338.

Shao, M. *et al.* (2015). An exact algorithm to compute the double-cut-and-join jistance for jenomes with duplicate genes. *JCB*, **22**(5), 425–35.

Tannier, E. *et al.* (2009). Multichromosomal median and halving problems under different genomic distances. *BMC Bioinformatics*, **10**(1), 120.

Vogelstein, B. *et al.* (2013). Cancer genome landscapes. *Science*, **339**(6127), 1546–58.

Yancopoulos, S. *et al.* (2005). Efficient sorting of genomic permutations by translocation, inversion and block interchange. *Bioinformatics*, **21**(16), 3340–3346.

Zakov, S. *et al.* (2013). An algorithmic approach for breakage-fusion-bridge detection in tumor genomes. *PNAS*, **110**(14), 5546–51.

Zeira, R. and Shamir, R. (2017). Sorting by cuts, joins, and whole chromosome duplications. *JCB*, **24**(2), 127–137.

Zeira, R. *et al.* (2017). A linear-time algorithm for the copy number transformation problem. *JCB*, **24**(12), 1179–1194.