BISER: Fast Characterization of Segmental Duplication Structure in Multiple Genome Assemblies

by

Hamza Išerić
B.Sc., University of Sarajevo, 2018

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

in the Department of Computer Science

BISER: Fast Characterization of Segmental Duplication Structure in Multiple Genome Assemblies

by

Hamza Išerić
B.Sc., University of Sarajevo, 2018

Supervisory Committee

Dr. Ibrahim Numanagić, Supervisor
(Department of Computer Science)

Dr. Sean Chester, Departmental Member
(Department of Computer Science)

Supervisory Committee

Dr. Ibrahim Numanagić, Supervisor
(Department of Computer Science)

Dr. Sean Chester, Departmental Member
(Department of Computer Science)

ABSTRACT

The increasing availability of high-quality genome assemblies raised interest in the characterization of genomic architecture. Major architectural elements, such as common repeats and segmental duplications (SDs), increase genome plasticity that stimulates further evolution by changing the genomic structure and inventing new genes. Optimal computation of SDs within a genome requires quadratic-time local alignment algorithms that are impractical due to the size of most genomes. Additionally, to perform evolutionary analysis, one needs to characterize SDs in multiple genomes and find relations between those SDs and unique (non-duplicated) segments in other genomes. A naïve approach consisting of multiple sequence alignment would make the optimal solution to this problem even more impractical. Thus there is a need for fast and accurate algorithms to characterize SD structure in multiple genome assemblies to better understand the evolutionary forces that shaped the genomes of today. Here we introduce a new approach, BISER, to quickly detect SDs in multiple genomes and identify elementary SDs and core duplicons that drive the formation of such SDs. BISER improves earlier tools by (i) scaling the detection of SDs with low homology (75%) to multiple genomes while introducing further 10–34× speed-ups over the existing tools, and by (ii) characterizing elementary SDs and detecting core duplicons to help trace the evolutionary history of duplications to as far as 300 million years.

Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

I would like to thank God, my supervisor, Ibrahim Numanagić, my family and friends.

DEDICATION

I dedicate this work to all scientists with pure intentions and to all people who supported me on this journey.

Chapter 1

Introduction

Segmental duplications (SDs), also known as low-copy repeats, are genomic segments larger than 1 Kbp that are duplicated one or more times in a given genome with a high level of homology. While nearly all eukaryotic genomes harbor SDs, it is the human genome that exhibits the largest diversity of SDs. At least 6% of the human genome is covered by SDs ranging from 1 Kbp to a few megabases [8]. The architecture of human SDs also differs from other mammalian species both in its complexity and frequency [28]. For example, while most species harbor tandem SDs, the human genome is repleted with interspersed (both intra- and inter-chromosomal) SD blocks [10]. Human SDs are also often duplicated multiple times within the genome, often immediately next to or even within an already existing SD cluster. This complex duplication architecture points to a major role that SDs play in human evolution [9, 7, 32]. Human SDs also introduce a significant level of genomic instability that results in increased susceptibility to various diseases [5, 20]. This has led to evolutionary adaptation in the shape of genes and transcripts unique to the human genome that aim to offset the effects of such instability [15]. Finally, SDs display significant diversity across different human populations and can be used as one of the markers for population genetics studies [40].

In order to understand the architecture and evolution of eukaryotic SDs, the first step consists of detecting all SDs within a given genome. However, SD detection is a computationally costly problem. The theoretically optimal solution to this problem—a local alignment of an entire genome to itself—is unfeasible due to large sizes of eukaryotic genomes that render the classical quadratic time algorithms such as Smith-Waterman impractical. Furthermore, the homology levels between SD copies—as low as 75%—prevent the use of the available edit distance approxi-

mations with theoretical guarantees [4, 21]. This is likely to remain so due to the sub-quadratic inapproximability of edit distance metrics [6]. The vast majority of sequence search and whole-genome alignment tools that rely on heuristics to compute the local alignments, such as MUMmer [33] and BLAST [2], also assume high levels of identity between two sequences and therefore are not able to efficiently find evolutionarily older SD regions. Even specialized aligners for noisy long reads, such as Minimap2 [31] or MashMap [25], cannot handle 75% homology that is lower than the expected noise of long reads (up to 15%, although sequencing error rates have been improved recently to 5%) [3]. Finally, even if we use higher homology thresholds (such as 90%) to define an SD, the presence of low-complexity repeats and the complex SD rearrangement architecture often prevents the off-the-shelf use of the existing search and alignment tools for detecting SDs.

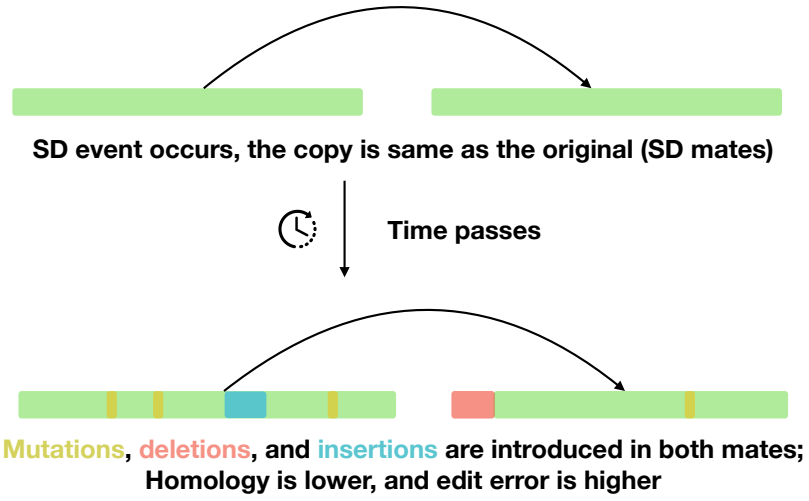


Figure 1.1: When an SD event occurs, SD mates are equal. However, as time passes, the similarity between them goes down, and edit error goes up due to two distinct processes: point mutations (orange color) and large block events (blue and red color).

For these reasons, only a few SD detection tools have been developed in the last two decades, and most of them employ various heuristics and workarounds—often without any theoretical guarantees—to quickly find a set of acceptable SDs. The gold standard for SD detection, Whole-Genome Assembly Comparison (WGAC), uses various techniques such as hard-masking and alignment chunking to find SDs [8]. While its output is used as the canonical set of SDs in the currently available genomes, and as such, forms the basis of the vast majority of SD analysis studies, WGAC can only find recent or highly conserved SDs (i.e., SDs with $> 90\%$ homology) within

primate lineages. Furthermore, WGAC requires specialized hardware to run and takes several days to complete. Few other tools developed as a replacement for WGAC—namely SDdetector [13] and ASGART [14]—are also limited in their ability to find SDs with less than 90% homology. Currently, the only tools that are able to detect SDs with lower homology are SDquest [36] and SEDEF [34].

SEDEF uses Levenshtein edit distance to model the similarity between two sequences (potential SD paralogs). Levenshtein edit distance is the minimal number of edit operations (deletions, insertions, and mutations) needed to make compared sequences the same. Edit error is the edit distance normalized over the alignment length of those sequences.

In order to efficiently catch SD paralogs with higher edit error, ε (edit error) is divided into ε_P (small, PSV mutations) and ε_B (block mutations – large-scale events or gaps), exploiting the unique biological properties of the SD evolutionary process. ε_P is modeled using a Poisson error to obtain the probability of an event in which a k-mer that belongs to an SD is preserved. SEDEF combines this with the MinHash approximation scheme (e.g., Jaccard similarity as a metric for determining similarity between two sequences), previously used for long-read alignment [25], to quickly find SDs even with 75% homology—thus enabling the study of SDs as old as 90 million years [34],—while also providing basic theoretical guarantees about the sensitivity of the search process.

SDquest, on the other hand, relies on k-mer counting to find initial (seed) SD regions, with an assumption that SD regions will share common k-mers that can be later extended and aligned with LASTZ [22].

It should be mentioned that an SD is often formed by copying parts of older, more ancient SDs to a different location. This, in turn, implies that each SD can be decomposed into a set of short building blocks, where each block either stems from an ancient SD or a newly copied genomic region. Such building blocks are called "elementary SDs" [28]. Elementary SDs are often shared across related species within the same evolutionary branch. It has been proposed that a small subset of elementary SDs—often dubbed *seeds* or *core duplicons*—evolutionarily drives the whole SD formation process and that every SD harbors at least one such core duplicon [28]. Based on their cores, SDs can be hierarchically clustered into distinct clades.

For example, the human genome SDs can be divided into 435 duplicon blocks that are further classified into 24 clades, seeded by a set of core duplicons with a total span of 2 Mbps that is often gene-rich and transcriptionally active [28]. The prime

example of a mosaic-like recombination region that is seeded by an SD core is the *LCR16* locus of the human genome that is shared with many other primates [10].

SDquest [36] decomposes SDs into elementary SDs using the A-Bruijn graph, previously developed in [35]. Elementary SDs reveal the mosaic structure of SDs, and this structure allows construction of breakpoint graphs to show the level of complexity of the SDs evolutionary processes [36].

The SD evolutionary history analysis and the detection of core duplicons require a joint analysis of SDs in many related species. However, while existing SD tools can find SDs in single genomes in a reasonable amount of time, none of them can scale—at least not efficiently—to multiple genome assemblies. Furthermore, no publicly available tool can provide a deeper understanding of SD evolutionary architecture or find core duplicons across different species, mostly due to the computational complexity of such analysis because of the large number of existing SDs within different species[1]. For these reasons, only a small subset of previously reported core duplicons was analyzed in-depth (e.g., *LCR16* cores), and often so by manually focusing on narrow genomic regions to make the analysis tractable [10], preventing the emergence of a clearer picture of the SD evolution across different species, especially of those SDs that preclude the primate branch of the evolutionary tree.

Here we introduce BISER (**B**risk **I**nference of **S**egmental duplication **E**volutionary st**R**ucture), a new framework implemented in Seq [38] and C++ that is specifically developed to quickly detect SDs even at low homology levels (75%) across multiple related genomes. BISER is also able to infer elementary and core duplicons and thus enable an evolutionary analysis of all SDs in a given set of related genomes. The key conceptual advances of BISER consist of a novel linear-time algorithm that can quickly detect regions that harbor SDs in a given set of genomes and a new approach to infer elementary SDs. The linear speedup was achieved by presenting ordered Jaccard similarity and a new way of tracing the same shared k-mers between two similar regions. We also show a comparison between two methods developed for decomposing SDs into a set of elementary SDs. One of them is based on the union-find algorithm, while the other one is based on k-mer chaining. BISER can find and align SDs in the human genome in 44 CPU minutes (one core)—a 10× speedup over SEDEF and 34× speedup over SDquest—and find all shared SDs in seven primate

[1]The source code that was used for older analyses [28] is not publicly available. SDquest, on the other hand, can detect elementary SDs but only at the single genome level. Furthermore, it does not provide exact genomic coordinates of the detected elementary SDs.

genomes in 4 hours on a standard 8-core laptop computer (without the need for using clusters). The flexibility of BISER will make it a useful tool for SD characterizations that will open doors towards a better understanding of the complex evolutionary architecture of these functionally important genomic events.

Chapter 2

Methods

2.1 Preliminaries

Consider a genomic sequence $G = g_1 g_2 g_3 \dots g_{|G|}$ of length $|G|$ and alphabet $\Sigma = \{A, C, G, T, N\}$. Let $G_i = g_i \dots g_{i+n-1}$ be a substring of G of length n that starts at position i in G. To simplify the notation, the length is assumed to be n. We will use an explicit notation $G_{i:i+n}$ for a substring of length n starting at position i when a need arises. Let $s_1 \circ s_2$ represent a string concatenation of strings s_1 and s_2.

Segmental duplications are long, low-copy repeats generated during genome evolution over millions of years. Following such an event, different copies of a repeat get subjected to different sets of mutations, causing them to diverge from each other over time. Thus, it is necessary to introduce a similarity metric between two strings in order to detect SDs in a given genome. To that end, we use Levenshtein's [29] *edit distance* metric $\mathcal{E}$ between two strings s and s' that measures the minimum number of edit operations (i.e., substitutions, insertions, and deletions at the single nucleotide level) in the alignment of s and s'. Let ℓ be the length of such alignment; it is clear that $\max(|s|, |s'|) \leq \ell \leq |s| + |s'|$. We can also define an *edit error* $\mathrm{err}(\cdot, \cdot)$ between s and s' (or, in the context of this paper, an *error*) as the normalized edit distance: $\mathrm{err}(s, s') = \mathcal{E}(s, s')/\ell$. Intuitively, this corresponds to the sequence divergence of s and s'. Now we can formally define an SD as follows:

Definition 1. *A segmental duplication (SD) within the error threshold ε is a tuple of paralog sequences (G_i, G_j) that satisfies the following criteria:*

1. $\mathrm{err}(G_i, G_j) \leq \varepsilon$;

2. *$\ell \geq 1,000$, where ℓ is the length of the optimal alignment between G_i and G_j; and*

3. *the overlap between G_i and G_j is at most $\varepsilon \cdot n$* [1].

Some SD events can be part of other, more ancient SD events. This means that those SDs share some common building blocks, and exhibit a mosaic structure. Those building blocks are called *elementary SDs*. For tracing the evolutionary history of all SDs, we would like to find all those elementary SDs. The process of finding all elementary SDs given set of all SDs is called the *decomposition* (discussed further in Section 2.1.3).

Given a set of genomes $G^1, \ldots, G^\gamma$ and their mutual evolutionary relationships, our goal is to:

- find a set of valid SDs, $\mathcal{SD}^i$, within each G^i **(SD detection)**;
- find all copies of both s and s' for $(s, s') \in \mathcal{SD}^i$ in other genomes $G^j, j \neq i$, if such copies exist **(SD cross-species conservation detection)**; and
- decompose each SD from $\mathcal{SD} = \mathcal{SD}^1 \cup \cdots \cup \mathcal{SD}^\gamma$ into a set of *elementary SDs* E, and determine the set of core duplicons that drive the formation of SDs in $\mathcal{SD}$ **(SD decomposition)**.

To that end, we developed BISER, a computational framework that is able to perform these steps efficiently, and we describe the algorithms behind it in the following sections. For the sake of clarity, unless otherwise noted, we assume that we operate on a single genome G. Since SDs are by definition different from low-complexity repeats and transposons, we also assume that all genomes $G^1, \ldots, G^\gamma$ are hard-masked, meaning that they do not contain such elements, wheres soft-masked genomes preserve low-complexity repeats represented in lowercase notation. The size of the SD length l is biologically defined, and we follow that definition [8].

2.1.1 SD Error Model

Different paralogs of an SD are mutated independently of each other. Therefore, the sequence similarity of paralogs is correlated with the age of the duplication event—more recent copies are nearly identical, while distant ancestral copies are dissimilar.

[1] Ideally, the SD mates should not overlap; however, due to the presence of errors, we need to account for at most $\varepsilon \cdot n$ overlap.

It has been proposed that the sequence similarity of older SDs (e.g., those shared by the mouse and human genomes) falls as low as 75% [34]. In other words, the dissimilarity between different copies of an old SDs exceeds 25% (i.e., $\text{err}(s, s') \geq 0.25$ for SD paralogs s and s', according to the definition above).

Detection of duplicated regions within such a large error threshold is a challenging problem, as nearly any edit distance approximation technique with or without theoretical guarantees breaks down at such high levels of dissimilarity [4, 25], provided that this error is truly random. However, that is not the case: it has been previously shown that the SD mutation process is an amalgamation of two independent mutation processes (Figure 1.1), namely the background point mutations (also known as *paralogous sequence variants*, or PSVs) and the large-scale block edits. As such, the overall error rate ε can be expressed as a sum of two independent error rates, ε_P (PSV mutation rate) and ε_B (block edit rate), where only ε_P is driven by a truly random mutation process.

In the case when paralogs share the 75% sequence identity, it has been shown that the random point mutations (PSVs) contribute at most 15% ($\varepsilon_P \leq 0.15$) towards the total error ε [34] (this also holds for many other mammalian genomes, as their substitution rate is often lower than the human substitution rate [17]). The remaining 10% (knowing that ε_P and ε_B are additive) is assumed to correspond to the block edit rate ε_B. Note that these mutations are clustered *block* errors and, as such, are not randomly distributed across SD regions. The probability of a large block event is roughly 0.005 (0.5 %) based on the analysis of existing SD calls.

On the other hand, we assume that PSVs between two SD paralogs s and s' follow a Poisson error model [25, 18] and that those mutations occur independently from each other. The Poisson error model for mutation is being used for 70 years [16], where mutations are considered to be independent Poisson events whose rate is constant, justifying the model assumption. It follows that any k-mer in s' has accumulated on average $k \cdot \varepsilon_P$ mutations compared to the originating k-mer in s, provided that such k-mer was part of the original copy event. By setting a Poisson parameter $\lambda = k \cdot \varepsilon_P$, we obtain the probability of a duplication event in which a k-mer is preserved in both SD paralogs (i.e., that a k-mer is error-free) to be $e^{-k\varepsilon_P}$.

2.1.2 SD Detection

Let us return to the main problem of determining whether two strings s and s' are "similar enough" to be classified as SDs. As mentioned before, classical edit distance calculation algorithms would be too slow for this purpose. Instead, we use an indirect approach that measures the similarity of strings s and s' by counting the number of shared k-mers in their respective k-mer sets $\mathbf{K}(s)$ and $\mathbf{K}(s')$. The subsequence of size k in a sequence s is called k-mer, and the k-mer set of sequence s is the set of all subsequences of size k in s. It has been shown that Jaccard index of these sequences, s and s', defined as $\mathcal{J}(\mathbf{K}(s), \mathbf{K}(s')) = \frac{|\mathbf{K}(s) \cap \mathbf{K}(s')|}{|\mathbf{K}(s) \cup \mathbf{K}(s')|}$ is a good proxy for $\mathcal{E}(s, s')$ under the Poisson error model [25]. Thus we can combine the Poisson error model with the SD error model and obtain the expected value of Jaccard index τ between any two strings s and s', whose edit error $\mathrm{err}(s, s')$ follows the SD error model and is lower than $\varepsilon = \varepsilon_P + \varepsilon_B$, to be [34]:

$$\tau = \mathbb{E}[\mathcal{J}(\mathbf{K}(s), \mathbf{K}(s'))] \geq \frac{1 - \varepsilon_B}{1 + \varepsilon_B} \cdot \frac{1}{2e^{k\varepsilon_P} - 1}. \tag{2.1}$$

Lower threshold is derived by following [34]:

1. $|\mathbf{K}(s) \cap \mathbf{K}(s')| = |\mathbf{K}(s)^P \cap \mathbf{K}(s')^P|$, where $\mathbf{K}(s)^P$ and $\mathbf{K}(s')^P$ are subsets of k-mers that do not correspond to large gaps;

2. $t_B/(t_P + t_B) \leq \varepsilon_B \Rightarrow t_B \leq t_P \cdot \varepsilon_B/(1 - \varepsilon_B)$, where $t_B = |\mathbf{K}(s)^B| = |\mathbf{K}(s')^B|$ and $t_P = |\mathbf{K}(s)^P| = |\mathbf{K}(s')^P|$; and

3. $|\mathbf{K}(s)^B \cup \mathbf{K}(s')^B| \leq 2|\mathbf{K}(s')^B| = 2t_B$ because $|\mathbf{K}(s)^B| = |\mathbf{K}(s')^B|$ (equality holds when $|\mathbf{K}(s)^B \cap \mathbf{K}(s')^B| = \emptyset$).

From this, it follows that:

$$\begin{aligned}
J(\mathbf{K}(s), \mathbf{K}(s')) = \frac{|\mathbf{K}(s) \cap \mathbf{K}(s')|}{|\mathbf{K}(s) \cup \mathbf{K}(s')|} &= \frac{|\mathbf{K}(s)^P \cap \mathbf{K}(s')^P|}{|\mathbf{K}(s)^P \cup \mathbf{K}(s')^P| + |\mathbf{K}(s)^B \cup \mathbf{K}(s')^B|} && \text{(by 1.)} \\
&\geq \frac{|\mathbf{K}(s)^P \cap \mathbf{K}(s')^P|}{|\mathbf{K}(s)^P \cup \mathbf{K}(s')^P| + 2t_B} && \text{(by 3.)} \\
&\geq \frac{|\mathbf{K}(s)^P \cap \mathbf{K}(s')^P|}{|\mathbf{K}(s)^P \cup \mathbf{K}(s')^P| + \frac{2\delta_B}{1-\delta_B}|\mathbf{K}(s)^P \cup \mathbf{K}(s')^P|} && \text{(by 2.)}
\end{aligned}$$

$$= \frac{1-\varepsilon_B}{1+\varepsilon_B} \frac{|\,\mathbf{K}(s)^P \cap \mathbf{K}(s')^P|}{|\,\mathbf{K}(s)^P \cup \mathbf{K}(s')^P|} = \frac{1-\varepsilon_B}{1+\varepsilon_B} J(\mathbf{K}(s)^P, \mathbf{K}(s')^P).$$

Knowing that $J(\mathbf{K}(s)^P, \mathbf{K}(s')^P) = c/(2t_P - c)$, where $c = |\,\mathbf{K}(s) \cap \mathbf{K}(s')|$, and the expected value of c/t_P is $e^{-k\varepsilon_P}$, we get equation 2.1 [34].

As we cannot use local alignment to efficiently enumerate all SDs in a given genome due to both time and space complexity (which is quadratic), we utilize a heuristic approach to enumerate all pairs of regions in G that are likely to harbor one or more segmental duplications. We call these pairs *putative SDs*. These pairs are not guaranteed to contain a "true" SD and must be later aligned to each other to ascertain the presence of true SDs. Nevertheless, such an approach will *filter out* the regions that do not harbor SDs, and thus significantly reduce the amount of work needed for detecting "true" SDs. The overall performance of our method, both in terms of performance and sensitivity, will depend on how well the putative SDs are chosen.

The problem of putative SD detection can be, thanks to the SD error model, easily expressed as an instance of a filtering problem: find all pairs of indices i, j in G such that $\mathcal{J}(\mathbf{K}(G_i), \mathbf{K}(G_j)) \geq \tau$, where τ is the lower bound from the Equation 2.1. Here we assume that the size of G_i and G_j exceeds the SD length threshold (1,000 bp), and no k-mer occurs twice in either G_i or G_j. Even if it does, the above-derived Jaccard score-based filter performs well in practice.

The filtering approach has already been successfully used in other software packages and forms the backbone of both SEDEF (SD detection tool) and MashMap (Nanopore read aligner). However, both methods maintain the k-mer sets $\mathbf{K}(s)$ and $\mathbf{K}(s')$ to calculate the Jaccard index between the sequences s and s'. As these methods dynamically grow s and s' (as the length n is not known in advance), the corresponding sets $\mathbf{K}(s)$ and $\mathbf{K}(s')$ are constantly being updated, necessitating a costly recalculation of $\mathbf{K}(s) \cap \mathbf{K}(s')$ on each update. A common trick is to use the MinHash technique to reduce the sizes of $\mathbf{K}(s)$ and $\mathbf{K}(s')$, and thus the frequency of such updates. However, the frequent recalculation of the Jaccard index still remains a major bottleneck even in the MinHash-based approaches because calculating union and intersection of k-mers for each pair of subsequences in G is a costly operation.

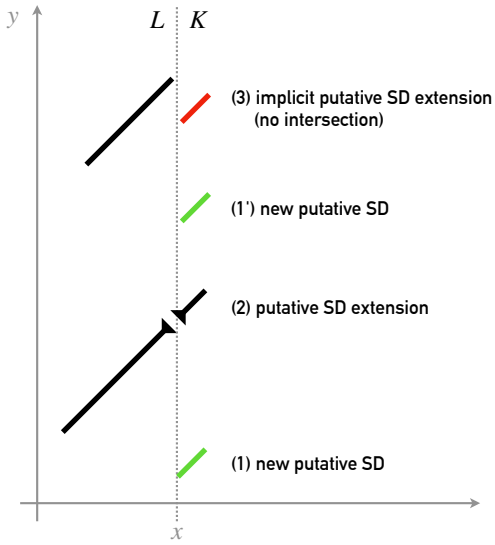
Here we note that the Jaccard index calculation can be significantly simplified by not having to maintain the complete k-mer sets $\mathbf{K}(s)$ and $\mathbf{K}(s')$. The need for keeping such sets arises from the fact that the calculation of $\mathbf{K}(s) \cap \mathbf{K}(s')$ allows any k-mer in

(a)

Algorithm 1: Algorithm for finding putative SDs.

Input : Genomic sequence G, its k-mer index I_G, threshold Δ.
Output: A list $\mathcal{SD}$ of putative SDs.
$L \leftarrow []$; $\mathcal{SD} \leftarrow []$;
for $x \leftarrow 1$ **to** $|G|$ **do**
  $K \leftarrow I_G[G_{x:x+k}]$; $(i_K, i_L) \leftarrow (1, 1)$;
  append (x, K_{i_K}, k) to L if L is empty;
  while $i_K \leq |K|$ **and** $i_L \leq |L|$ **do**
    $y \leftarrow K_{i_K}$; $(\ell_x, \ell_y, l) \leftarrow L_{i_L}$;
(1) **if** $y < \ell_y$ **then**
      insert (x, y, k) to L before i_L;
      advance i_K and i_L;
(2) **else if** $y \geq \ell_y + l$ **and** $\max(x - \ell_x, y - \ell_y) - l \leq \Delta$ **then**
      extend L_{i_L} to cover $G_{x:x+k}$ and $G_{y:y+k}$;
      increase counts for $\bigcup(L_{i_L})$ and $\bigcap(L_{i_L})$;
      advance i_K; $i_L \leftarrow$ `CheckJaccard` (L_{i_L})
(1') **else if** $y \leq start(L_{i_L+1})$ **then**
      insert (x, y, k) to L before i_L;
      advance i_K and i_L;
(3) **else**
      increase count for $\bigcup(L_{i_L})$;
      $i_L \leftarrow$ `CheckJaccard` (L_{i_L})
    end
  end
end

(b)



(c)

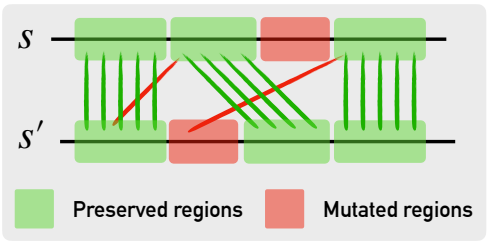


Figure 2.1: **(a)** A plane-sweep algorithm for finding putative SDs. **(b)** Visual guide for the algorithm. The algorithm sweeps a vertical dashed line through the set of winnowed k-mers in a genome G (x axis). At each k-mer starting at the location x, it queries the index I_G to obtain a sorted list K of k's occurrences in G (right side of the sweep line). The algorithm then scans K, and the list L of putative SDs found thus far at the same time. At each step, it examines i_L-th element of L and i_K-th element of K, and decides whether to start a new putative SD ((1) and (1'), green k-mers on the right), extend the current putative SD with the current k-mer ((2), black k-mer on the right), or subsume the current k-mer within the current putative SD ((3), red k-mer). **(c)** A visual representation of a valid k-mer matching in a valid alignment (shown by green lines). Red matching would render the alignment invalid as red matchings are not co-linear with the green matchings.

$\mathbf{K}(s')$ to match any k-mer in $\mathbf{K}(s)$. However, such a loose intersection requirement is not only redundant for approximation of edit distance under the SD error model but is even undesirable as such intersections can introduce cross-over k-mer matches that are not possible in the edit distance metric space (see Figure 2.1c for an example of valid and invalid matchings).

By disallowing such cross-over cases, we can significantly optimize the calculation of the Jaccard index. Let us show how to do that without sacrificing sensitivity. Let us first introduce $s \circledast s'$ as an alternative way of measuring the k-mer similarity between strings s and s'.

For that purpose, let us introduce a notion of a *co-linear k-mer matching* between s and s' as a set of index pairs (i, j) $(1 \leq i \leq |s|, 1 \leq j \leq |s'|)$ such that the k-mers that start at i and j in s and s' respectively are equal, and such that all pairs (i, j) in matching are co-linear (i.e., for each (i, j) and (i', j'), either $i < i'$ and $j < j'$, or $i > i'$ and $j > j'$).

A $\circledast$ operation describes the size of a maximum co-linear matching of k-mers between s and s'. In other words, we want to select a maximal set of matching k-mers in $\mathbf{K}(s)$ and $\mathbf{K}(s')$ such that no two k-mer matchings cross over each other (see Figure 2.1c for an example of cross-over, or non-co-linear, matchings). We can replace $\mathbf{K}(s) \cap \mathbf{K}(s')$ with $s \circledast s'$ and introduce an *ordered Jaccard index* $\hat{\mathcal{J}}(s, s')$, formally defined as:

$$\hat{\mathcal{J}}(s, s') = \frac{s \circledast s'}{|\,\mathbf{K}(s) \cup \mathbf{K}(s')|}.$$

The following lemma allows us to use an ordered Jaccard index $\hat{\mathcal{J}}$ in lieu of classical Jaccard index $\mathcal{J}$:

Lemma 1. *The ordered Jaccard index $\hat{\mathcal{J}}(s, s')$ of two strings s and s' is equal to the Jaccard index $\mathcal{J}(\mathbf{K}(s), \mathbf{K}(s'))$ (under the assumptions of SD error model, namely the separation of ε_B and ε_P), assuming that s and s' only share k-mers that have not been modified by PSVs following the originating copy event.*

Proof. It is sufficient to prove that the size of $|\,\mathbf{K}(s) \cap \mathbf{K}(s')|$ always corresponds to the size of maximal co-linear matching between s and s'.

To show that $s \circledast s' \leq |\,\mathbf{K}(s) \cap \mathbf{K}(s')|$, it is enough to note that matched k-mers in any co-linear matching are by definition identical and thus belong to $\mathbf{K}(s) \cap \mathbf{K}(s')$.

We will prove that $s \circledast s' \geq |\,\mathbf{K}(s) \cap \mathbf{K}(s')|$ by contradiction. First, note that the

string s is equal to s' immediately after the duplication event (i.e., before the occurrence of PSVs) and that all k-mers are co-linear in their maximal matching because s contains no repeated k-mers (an assumption made by the SD error model). Now, suppose that there is a cross-over in $\mathbf{K}(s) \cap \mathbf{K}(s')$. That implies either a cross-over between s and s' before PSVs occurred—contradicting the previous observation—or a cross-over after it, contradicting the assumption that any matched k-mer pair was matched before the occurrence of PSVs.

Hence $\mathbf{K}(s) \cap \mathbf{K}(s')$ cannot contain any cross-overs, and $s \circledast s' = |\mathbf{K}(s) \cap \mathbf{K}(s')|$.

If the conditions of Lemma 1 are satisfied, we can calculate $s \circledast s'$ in linear time by a simple scan through s and s' at the same time. A linear calculation of $s \circledast s'$, together with the fact that the lower bound τ in Equation 2.1 equally holds for $\hat{\mathcal{J}}$ as well (a direct consequence of Lemma 1), allows us to use a plane sweep technique to select all pairs of substrings (s, s') in G whose ordered Jaccard distance $\hat{\mathcal{J}}(s, s')$ exceeds τ, and as a result, select all putative SDs in G (see Figure 2.1 for details).

We begin by creating a k-mer index I_G that connects each k-mer in G to an ordered list of its respective locations in G. For its creation, we use a *hash index* data structure. The index size will be $O(4^k)$, and the time complexity of adding a new k-mer location is $O(1)$. Then we sweep a vertical line in G from left to right while maintaining a sorted list L of putative SDs found thus far. For each location x in G encountered by a sweep line, we query I_G to obtain a sorted list K containing loci of $G_{x:x+k}$'s copies in G. Then, for any y in K, we check if it either (1) begins a new potential putative SD that maps x to y, (2) extends an existing putative SD, or (3) is covered by existing putative SD in L. We can see these steps in Figure 2.1.
If a putative SD in L is too distant from y, it is promoted to the final list of putative SD regions if it satisfies the ordered Jaccard index threshold τ and the other SD criteria from Definition 1. Note that we do not allow a k-mer to extend a putative SD if the distance between it and the SD exceeds the user-defined threshold Δ (set to 250 by default; as the maximum gap size for a minimal length of an SD—due to a possible indel—is 250). It takes $|L| + |K|$ steps to process each k-mer in G because both L and K are sorted. However, because the size of $|L|$ is kept low by the distance criteria, and because $|K|$ is low enough in practice[2], the practical time complexity of Algorithm 1 is $O(|G|)$ (theoretically, the worst-case complexity is $O((|L| + |K|) \cdot |G|)$) for constructing the index I_G, and linear in terms of the genome

[2]The average size of L in our experiments was 370, and the average size of K is 30.

size for plane sweeping.

The key assumption in Lemma 1—that two paralogs only share the k-mers that have not been mutated since the copy event— does not always hold in practice on real data. As such, Algorithm 1 might occasionally underestimate the value of $\hat{\mathcal{J}}$, potentially leading to some false negatives. We control that by using Δ—the same parameter that controls the growth of putative SDs by limiting the maximum distance of neighboring k-mers in $s \circledast s'$ (Figure 2.1)—to limit the growth of under-estimated SDs and thus start the growth of potentially more successful SDs earlier.

This heuristic might cause a large SD to be reported as a set of smaller disjoint SD regions. For that reason, we post-process the set of putative SDs upon the completion of Algorithm 1 and merge any two SDs that are close to each other if their union satisfies the ordered Jaccard index criteria. We also extend each putative SD by 5 Kbp both upstream and downstream to account for the small SD regions that might have been filtered out during the search process. This parameter is user-defined and might be adjusted for different genome assemblies.

To show the correctness of Algorithm 1, we will discuss its steps in terms of the classic plane sweeping algorithm. Three basic elements that have to be maintained in every plane sweeping algorithm are:

- the *partial solution* that has already been constructed to the left of the sweep line: a partial solution will be generated, finding all SDs left of the sweeping line position in subsequence of the genome that is read so far.

- the *sweep-line status*: sweep line will read genome G, and for each k-mer that it intersects, it will fetch all locations of that exact k-mer in a subsequence that is read so far. We will then: (1) update I_G, appending new k-mer location and therefore building I_G dynamically (as we process all SDs in G), and (2) update L, which is a list of potential SDs that map to the current region.

- a subset of the *future events* to be processed: these are SDs right of the sweeping line that will be read in the future.

The efficiency of the plane sweeping algorithm depends on how efficient update and store operations are. Appending new location in I_G is $O(1)$. Because L and $K = I_G^{k_i}$ are always maintained sorted, updating L with locations from $I_G^{k_i}$ only require $O(|L| + |K|)$ steps by using merge procedure (both are relatively small in practice). Also, L is implemented as a *linked list*, so removing elements from it is $O(1)$ as well.

Lemma 2. *Two subsequences from the genome G, s and s' that satisfy SD criteria (s and s' being mates of the same SD) and are subject to the error model shown in Section 2.1.1 share the same k-mers that can efficiently be chained and tracked in a plane sweeping fashion.*

Proof. Without loss of generality, assume s starts at location i in the genome G, s' in location j, and $i < j$. If we apply a plane sweeping algorithm from Section 2.1.2, we will, initially, save the locations of k-mers from the sequence s in the index I_G. When we reach location j in G with the sweeping line, we will fetch locations of k-mers belonging to s from I_G for all k-mers being intersected by the sweeping line. Those locations will be chained and saved in L efficiently because both L and I_G^k are sorted (I_G^k being a list of k-mer locations in I_G obtained so far). When we finish reading the last shared k-mers from s', the node that represents SD of s and s' will satisfy both lower Jaccard threshold (τ) and the length criteria. As we proceed to read k-mers after s', Jaccard similarity will decrease as the union of k-mers of those two regions continues to go up (but not the intersection). When it falls below the threshold, we will remove that node from L, save it as an SD if it previously satisfied SD criteria (which did in this case), keeping only still active SDs in L for the current position of the sweeping line.

The performance of the plane sweep technique can be further improved by winnowing the set of k-mers used for the construction of I_G [25]. Instead of indexing all k-mers in G, we only consider k-mers in a *winnowing fingerprint* $W(G)$ of G. $W(G)$ is calculated by sliding a window of size w through G and by taking in each window a lexicographically smallest k-mer (the rightmost k-mer is selected in case of a tie).

The expected size of $W(G)$ for a random sequence G is $2|G|/(w+1)$ [37]. The main benefit of winnowing is that it can significantly speed up the search step (up to an order of magnitude) without sacrificing sensitivity. The winnow $W(G)$ can be computed in a streaming fashion in linear time using $O(w)$ space with the appropriate data structures (deque) [11].

Following the discovery of putative SDs, we locally align paralogs from each putative SD and only keep those regions whose size satisfies the SD criteria mentioned above. BISER uses a two-tiered local chaining algorithm described previously in SEDEF that uses a seed-and-extend approach and efficient $O(n \log n)$ chaining method following by a SIMD-parallelized sparse dynamic programming algorithm to calculate the boundaries of the final SD regions and their alignments [1, 31, 41].

2.1.3 SD Decomposition

Once the set of final SDs $\mathcal{SD} = \{(s_1, s_1'), \dots\}$ is discovered and the precise global alignment of each paralog pair $(s, s') \in \mathcal{SD}$ is calculated, we proceed by decomposing the set $\mathcal{SD}$ into a set of evolutionary building blocks called *elementary SDs*. More formally, we aim to find a minimal set of elementary SDs $E = \{e_1, \dots, e_{|E|}\}$, such that each SD paralog s is a concatenation of $\hat{e}_1^s \circ \dots \circ \hat{e}_{n_s}^s$. Each $\hat{e}_i$ either belongs to E or there is some $e_j \in E$ such that $\mathrm{err}(\hat{e}_i, e_j) \leq \varepsilon$. An example of such a decomposition is given in Figure 2.2.

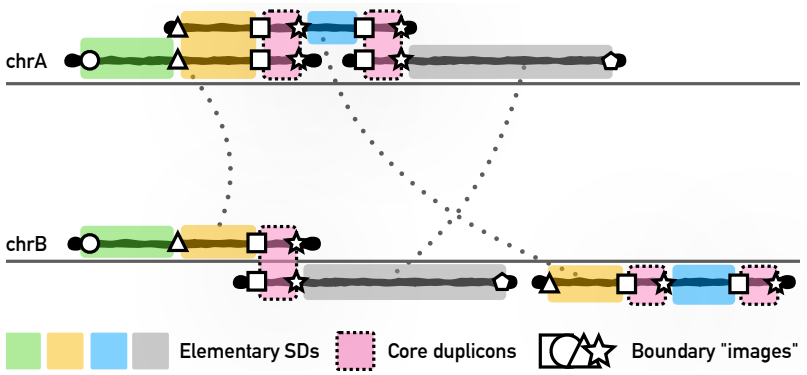


Figure 2.2: A decomposition of three partially overlapping SDs into a set of elementary SDs. Each paralog pair is indicated by a pair of two thick black lines linked by a dashed line. Each elementary SD is represented as a colored box. The boxes of core duplicons—elementary SDs shared by all SDs—are depicted with a dashed border. Note that a boundary of each elementary SD is induced by a boundary of an existing SD. Different boundaries are represented by different shapes, and their images (paralog copies) also share the same shape. For the sake of simplicity, we only show the identifiers (shapes) for locations that define elementary SD boundaries.

Two algorithms for decomposition are presented below. The first one is based on merging homologous locations using the union-find algorithm, and the second one is based on chaining the identical k-mers into elementary SDs.

2.1.4 Union-Find approach

For the first approach, we noted that each locus covered by an SD paralog is either copied to another locus during the formation of that SD (in other words, it is "mirrored" by its paralog) or belongs to an alignment gap. As SD events can copy over the regions that already form an existing SD, a single locus might "mirror" a large number of existing locations. In order to find all locations that a locus i mirrors,

we use a modification of Tarjan's union-find disjoint set algorithm [43] to link together all mirrored locations. This algorithm begins by giving each locus in a genome covered by an existing SD a distinct identifier (represented by a distinct shape in Figure 2.2). It then iterates over the set of SDs, and for each pair of SD paralogs (s, s') uses the global alignment between s and s' to link the identifiers of any two loci that are mirrored by the SD event. Linking merges the two identifiers into a single identifier. Upon the completion of the algorithm, all copies ("mirrors") of each locus will share the same identifier. The process of merging identifiers is similar to the gluing procedure from RepeatGluer [35].

Note that the boundaries of SD paralogs, or their images, correspond to the boundaries of elementary SDs, as each paralog by definition starts and ends with an elementary SD (Figure 2.2). However, as the set of elementary SDs should be minimal, it is not only necessary but sufficient to focus only on the identifiers that belong to a boundary of an existing SD paralog to construct the set of elementary SDs E. These identifiers describe a set of locations in G that form the boundaries of elementary SDs. Thus, we can obtain the final set E by iterating over G and checking if a locus is identified with a boundary-covering identifier.

In practice, SD boundaries and SD alignments are highly uneven, and SDs themselves exhibit a complex mosaic structure that often introduces "mirror loops" [36] that can collapse multiple unrelated loci into a single identifier. BISER heuristically handles these cases by discarding any mirrored loci that lie within a close distance of an already existing elementary SD boundary.

After decomposing SDs into the set of elementary SDs E, we select some of them as *core duplicons*. We define these duplicons as the minimal set of elementary SDs that cover all existing SDs (an SD is covered by an elementary if either paralog is composed of that elementary SD). We use a classical set-cover approximation algorithm [12] to determine a set of core duplicons from E.

Despite having good CPU performance (see Results section), this approach could not provide precise elementary SD boundaries for some of the most complex mosaic structures due to inconsistent alignments that cause alignment loops. Keeping in mind that the SDs are rich in overlapping regions, alignment loops cause most elementary SDs to fall into the same elementary "super" set that can not be further decomposed.

2.1.5 k-mer chaining approach

The alternative approach for the decomposition problem is conceptually similar to the previously presented Algorithm 1 for finding putative SDs. We start by denoting the set of all regions in genome G that contain SDs as R. Due to the high similarity of SDs, elementary SDs are supposed to have similar sequences and therefore should consist of identical k-mers that can be chained. We define *chaining* as the merging of proximal locations of identical k-mers ($k = 10$). Chaining k-mers can be used to simulate local multiple sequence alignment on R. The length of each local multiple sequence alignment (LMSA), denoted as μ, has to be at least 100bp long, following the definition of minimal elementary SD length from [28]. Two k-mers can be chained if their locations are within defined parameter d_g. Parameter d_g has two purposes: (1) it defines the maximum distance up to which one k-mer location can be appended onto another, and (2) it ensures that there will be at least one matching k-mer every d_g locations in each LMSA, reducing the number of false positives and random hits. Bigger values for d_g enable chaining k-mers that are further from the last appended k-mer. Longer regions are covered if matched k-mers are more distant; however, the possibility for false-positive putative elementary SD increases with d_g. At the same time, the quality of the local multiple sequence alignment decreases due to the (1) lower count of included k-mers, and (2) their sparse distribution. If d_g is too small, the method will not catch small gaps or the regions that might contain point mutations (each point mutation will affect k k-mers). For the predefined value of μ, we set the value of d_g to be 50, large enough to catch regions with mutations and small gaps, and small enough to ensure the good quality of the LMSAs.

The decomposition process goes as follows. We build a k-mer index I_k that points to a list of k-mer locations as values (this is the first difference compared to the search algorithm where we used only winnowed k-mers). We again scan all sequences using the same sweeping line algorithm previously discussed in Section 2.1.2. Earlier, a sweeping line was used to track a single copy of an existing region that is being scanned. Now it is used for finding all copies of sub-regions in R that satisfy the presented LMSA criteria. A list L keeps putative elementary SDs found so far. Whenever we process a new k-mer, we will take all locations from I_k and see if we can: (1) append them to an existing putative elementary SD from L (if L is empty, we initialize it with the current k-mer's positions); (2) create a new potential elementary SD; or (3) remove an existing one if it satisfies the deletion criteria. A new

location from I_k can be appended to an existing elementary SD if its distance from the last appended k-mer to that elementary SD is within d_g. A putative elementary SD is removed if no new k-mer location is appended to that putative elementary SD in d_g steps. The main difference from the search step (Section 2.1.2) is that we need to track multiple copies of one region (one true elementary that might be spanning through multiple SD loci) instead of only one. For this purpose, when we remove a node, instead of only removing that node from L, we will also remove all nodes from L, saving nodes that are larger than the threshold μ as one elementary SD set. All those nodes satisfy both of the LMSA criteria.

The computational performance of this approach heavily depends on the size of an I_k. To reduce its size, we cluster all overlapping SDs, merge sequences that overlap, and apply the same algorithm on every cluster separately in parallel, reducing each cluster's index size. Clustering SDs is done using the union-find algorithm. The largest cluster for human SDs contains around 31,000 sequences covering 91Mbp. This implies that those SDs in this cluster have a rich evolutionary history, and that by breaking them into elementary SDs (building blocks) we can shed light on the evolutionary processes that drove the creation of these SDs.

The union-find approach had issues with loops that can arise as a result of inconsistent alignments. k-mer chaining approach avoids loops by not using original alignments but by instead simulating local multiple sequence alignment on regions containing SDs. Therefore, this approach is able to provide more precise results at the cost of the moderately decreased runtime performance.

2.2 Multiple Genomes

The above method can be efficiently scaled to γ distinct genomes $G^1, \ldots, G^\gamma$ by constructing a set of k-mer indices $I_{G^1}, \ldots, I_{G^\gamma}$, and by running the search procedure on each G^i in parallel. There we obtain SDs for each genome $G^1, \ldots, G^\gamma$ that we later use for mapping to other genomes.

Note that the size of the joint genome index grows sub-linearly with the number of new appended genomes, as most share the large number of common k-mers. Also, note that this method can be trivially extended to search for reverse-complemented SD copies by adding an additional iteration over $\bar{G}$, where $\bar{G}$ is a reverse complement of G.

We can find SDs in G^i concurrently while building I_{G^i}. As we append new k-

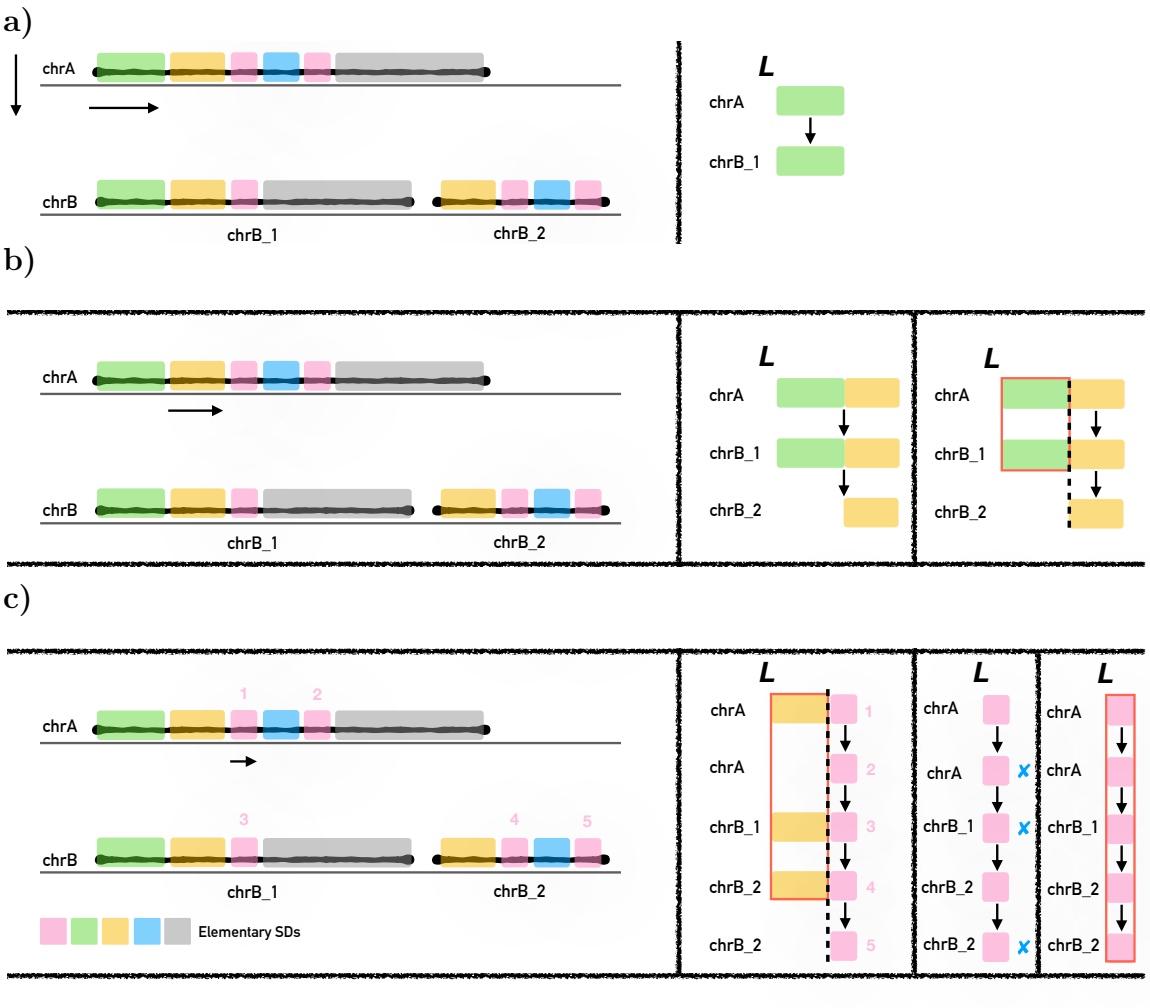


Figure 2.3: k-mer chaining method applied on the example from Figure 2.2. After the clustering and merging steps, we and up with three sequences (*chrA*, *chrB_1*, and *chrB_2*) that we scan to track identical k-mers. The first identical region is the green region in *chrA* that matches the green region in *chrB_1*. Once we start reading the yellow region (b), we notice a new elementary SD (because a new node is created); therefore, we will cut green regions as a separate elementary SD set. Suppose no k-mer can longer be appended to one of the nodes in L. In that case, all nodes that are larger than μ will be saved as one elementary SD (c; pink elementary SDs numbered as 2, 3, and 5 as there is no blue region that can be appended on them). We continue the same process for every other colored region.

mer locations to I_{G^i}, we can track previous locations of those SDs and keep track of putative SDs so far. This optimizes serial SD search in G^i. For preventing conserved regions between two species from being marked as SDs, BISER only maps regions from one genome that contains SDs to another whole-genome sequence.

Chapter 3

Results

We have evaluated all stages of BISER for speed and accuracy on both simulated and real-data datasets. All results were obtained on a multi-core Intel Xeon 8260 CPU (2.40GHz) machine with 1 TB of RAM. The run times are rounded to the nearest minute and are reported for both single-core as well as multi-core (8 CPU cores) modes when ran in parallel via GNU Parallel [42]. All real-data genomes were hard-masked, and all basepair coverage statistics are provided with respect to the hard-masked genomes.

In our experiments, we used $k = 14$ when searching for putative SDs and $k = 10$ during the alignment step (note that both parameters are user-adjustable). The size of the winnowing window was set to 16. We found that the lower values of k significantly impact the running time without providing any visible improvement to the detection sensitivity, while higher values of k significantly lower the detection sensitivity.

For the k-mer chaining method, the value of k was set to 10. Both k and w values (for search, align, and k-mer chaining decomposition) were empirically chosen to maximize sensitivity without impacting the runtime performance.

3.1 Simulations

3.1.1 SD detection

The accuracy of using the strong Jaccard index together with the SD error model as a function of error parameter ε, as well as the overall sensitivity of BISER's SD detection pipeline, was evaluated on a set of 1,000 simulated segmental duplications rang-

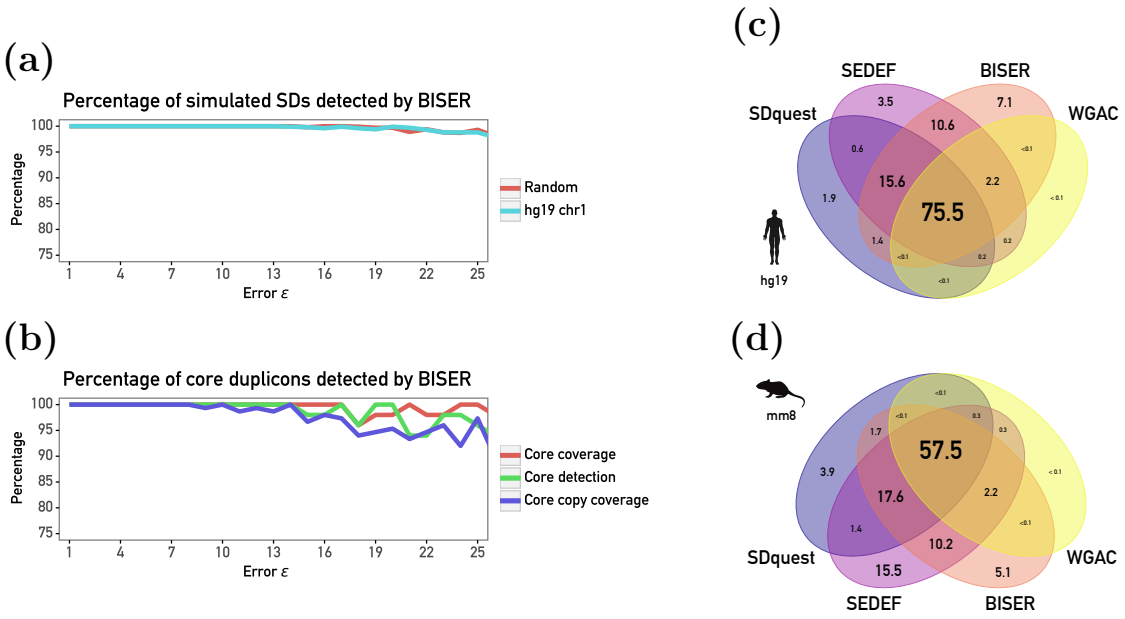


Figure 3.1: **(a)** Performance of BISER's algorithm on simulated SDs. x-axis represents the simulated SD error rate ε, while y axis represents the percentage of correctly detected SDs. Note that the plot area is cropped as BISER detects more than 98% of simulated SDs for any $\varepsilon \leq 0.25$. **(b)** Performance of BISER's core duplicon detection on simulated cores. The red line shows the percentage of ancestral core locations covered by a detected SD; the green line shows the percentage of cores identified as such; while the blue line shows the coverage of later core copies covered by a detected SD. Note that the plot area is also cropped. **(c)** Venn diagram depicts the SD coverage of the BISER, WGAC, SEDEF and SDquest (in Mbp) on the hard-masked human genome (hg19). **(d)** Venn diagram depicts the SD coverage of the BISER, WGAC, SEDEF and SDquest (in Mbp) on the hard-masked mouse genome (mm8). Note that nearly all bases out of ≈17 Mbp bases that are shown to be unique to SEDEF (and not covered by BISER) map to gaps and low-copy repeats and should be therefore treated as noise (not true SDs).

ing from 1 to 100Kbp. All sequences and mutations were randomly generated with uniform distribution according to the SD error model with $\varepsilon \in \{0.01, 0.02, \ldots, 0.25\}$ (i.e., we allowed the overall error rate to reach 25%). Uniform distribution was picked because it was an overall good biological proxy for mutation in known genomes and because it can represent worst-case mutation distribution (having one mutation on each non-overlapped k-mer). We consider a simulated SD as being "covered" if BISER found an SD that covers more than 90% of the original SD's basepairs. As shown in Figure 3.1a, the overall sensitivity is around 99% even for $\varepsilon = 0.25$.

We performed the same experiment on human (hg19) chromosome 1 (Figure 3.1a),

where we selected uniformly at random 10,000 sequences of various lengths and duplicated them within the chromosome. Each duplication was followed by introducing random PSVs according to the SD error model while varying the values of ε as described above. Even in this case, BISER's performance stays the same, and only a handful of very small SDs (of size ≈1,000) were missed.

3.1.2 Core duplicon detection

We also devised a simulation experiment to measure the power of BISER's core duplicon detection module (union-find approach). We began with a random DNA sequence of size 10 Mbp. Then we simulated an evolutionary process by introducing a set of novel SDs for 50 generations. In each generation, we introduced a novel core duplicon that did not overlap with an existing core and introduced a random number of SD events according to the SD error model that contains at least one core duplicon seen thus far. In each iteration, we made sure that the difference between paralogs does not exceed $\varepsilon \in \{0.01, 0.02, \ldots, 0.25\}$ regardless of their age. We finally used BISER to analyze the final sequence to predict SDs and their elementary decomposition (using union-find approach), as well as the core duplicons. BISER was able to successfully cover all ancestral cores, and properly decompose them into the elementary SDs and finally identify them as core duplicons (Figure 3.1b). Furthermore, BISER covered 95% or more of the more recent copies of the ancestral cores. We note that the combination of large block indels leads BISER (union-find approach) to occasionally report a single ancestral core as a set of 2 or more core duplicons or to report an elementary SD that covers less than 90% of a core. These cases are rare and happen less than 8% of the time at the highest levels of ε. The sensitivity of core duplicon detection is further described in Figure 3.1b.

Table 3.1: Running time performance of BISER (single-core and 8-core mode) on Intel Xeon 8260 CPU at 2.40 GHz for single genomes (hg19 and mm8).

Single genome (hg19)

	Total	Putative SDs	Alignment	Decomposition (UF)
1 core	**44m**	19m	24m	<1m
8 cores	**7m**	3m	4m	<1m

Single genome (mm8)

	Total	Putative SDs	Alignment	Decomposition (UF)
1 core	**1h 5m**	19m	46m	<1m
8 cores	**11m**	3m	8m	<1m

Table 3.2: Running time performance of BISER (single-core and 8-core mode) on Intel Xeon 8260 CPU at 2.40 GHz for seven genomes.

	Seven genomes (see below)			
	Total	Putative SDs	Alignment	Decomposition (UF)
1 core	**26h 6m**	9h 1m	17h 4m	6m
8 cores	**4h 1m**	1h 25m	2h 36m	6m

3.2 Single-genome results

We have run BISER on the *H.sapiens* hg19 genome and *M.musculus* mm8 genome and compared it to the published WGAC [8][1], SEDEF [34], and SDquest [36] SD calls. We also compared the runtime performance of BISER to that of SEDEF and SDquest. Note that we were not able to run WGAC due to the lack of hardware necessary for its execution. We did not compare BISER to other SD detection tools—namely SDdetector [13], MashMap2 [26], and ASGART [14]—as it has been previously shown that these tools underperform when compared to SEDEF or SDquest, and require an order of magnitude more resources than either SEDEF or SDquest do (see [34] for the detailed comparisons with these tools). For the same reason, we did not compare BISER to whole-genome aligners such as Minimap2 [31] and MUMmer/nucmer [33], as well as DupMasker [27], as none of these tools were designed to detect *de novo* SDs in a genome.

BISER was able to find and align all SD regions in hg19 in 6 minutes and 40 seconds on 8 cores (roughly 45 minutes on a single core) (Table 3.1). To put this into perspective, BISER is around 10× faster than SEDEF, 34× faster than SDquest, and an order of magnitude faster than WGAC that takes days to find human SDs (personal communication; we were not able to run the WGAC pipeline ourselves due to legacy hardware requirements). As a side note, BISER has the same memory requirements as SEDEF or SDquest and needs around 6 GB of RAM per core (it needs around 2 GB for the search step and 6 GB for the align step). Since SEDEF by default operates on a genome that is not hard-masked, we also ran SEDEF on a hard-masked genome to measure its theoretical speed (note that SEDEF was not designed for hard-masked genomes; thus, the basepair analysis is omitted). SEDEF took 21 minutes on 8 CPU cores to process a hard-masked hg19, leaving it still around 3× slower than BISER.

[1] `http://humanparalogy.gs.washington.edu`

Noticeable speedup is obtained in the first step of the algorithm—finding putative SDs—where SEDEF completes in 14 mins while BISER needs only 2 minutes.

Similar performance gains were observed on the mouse (mm8) genome as well. BISER took 11 minutes to find SDs in the mm8 genome (2.5 minutes for finding putative SDs and 8.5 minutes for alignment) while SEDEF needed 1 hour and 24 minutes (33 minutes for finding putative SDs and 51 minutes for align). SDquest took more than 6 hours for the same operation. SEDEF was run on soft masked data; when we ran it on hard masked data, it took 27 minutes. Here, the speedup is shown in the first step—finding putative SDs—where BISER needs 2.5 minutes while SEDEF needs 18 minutes.

In terms of sensitivity, BISER discovers about 1 GB of putative SD regions in the human genome. The exact coverage depends on parameters, like the minimum putative SD length that, in our case, is 500 bp for a query and 100 bp for a reference sequence. If we set them to be 1,000 bp and 300 bp, respectively, this coverage goes down to roughly 400 Mbp. The difference in the coverage does not significantly affect the timing nor the sensitivity of the align step because false positives are being quickly filtered by the align step due to a lack of shared regions that need to be chained. Those regions are usually small (around 10,000 basepairs due to padding of 5 kbp on each side).

After the alignment step, BISER reports 112 Mb of final SD regions within the 75% edit distance in hg19. That is 34 Mbp more than WGAC and 17 Mbp more than SDquest. The total coverage of SEDEF and BISER are similar to each other, differing by 3 Mbp uniquely detected by SEDEF and 7 Mbp uniquely covered by BISER (Table 3.3). BISER misses a few Mbp of SD regions unique to SDquest and a negligible amount unique to WGAC (Figure 3.1). On the mm8 genome, we can observe similar trends. However, we also observed that SEDEF covers roughly 17 Mbp that are not covered by BISER (Figure 3.1). When SEDEF is run on a hard-masked genome, it does not cover these bases; further analysis showed that nearly all bases ($\geq$16.3 Mbp) originally reported as unique to SEDEF actually map either to alignment gaps, soft-masked repeat elements, or small islands ($<$200bp) between the low-copy repeats and as such do not constitute "true" SDs.

Regarding basepair comparison between BISER and hard-masked SEDEF, for both human and mice genomes, BISER found a few hundred Kbp more than SEDEF. Both reported unique $\approx$1Mbp of SDs for each species.

Table 3.3: SD coverage of the human and mouse genomes (hg19 and mm8) and the runtime performance of BISER, SEDEF and SDquest. "Missed" and "Extra" columns are calculated with respect to the WGAC SD calls. All running times are reported on 8 CPU cores. We could not run WGAC as we do not have access to the legacy hardware needed for its execution; the reported runtime is from [34].

hg19

Tool	**Covered** (MBp)	**Missed** (MBp)	**Extra** (MBp)	**Time**
WGAC (standard)	78.2			days
BISER	112.4	0.4	34.6	7m
SEDEF	108.4	0.1	30.3	1h 15m
SDquest	95.2	2.5	19.5	3h 56m

mm8

Tool	**Covered** (MBp)	**Missed** (MBp)	**Extra** (MBp)	**Time**
WGAC (standard)	60.6			days
BISER	94.4	0.8	34.6	11m
SEDEF	105.2	0.1	44.7	1h 24m
SDquest	82.5	2.7	24.5	6h 06m

3.2.1 Decomposition

The first decomposition method (union-find method) found roughly 67,000 elementary SDs that describe hg19 SD calls. Of those, 2,906 were identified as core duplicons. BISER's core duplicons cover all 100 of the core duplicons reported in the earlier work [28], including the cores from the *LCR16* cluster. Note that many previously identified core duplicons in the hg17 version of the human genome turned out to be short tandem repeats in the hg19 version. The whole decomposition process took less than a minute on the final set of roughly 58,000 SDs.

The second decomposition method (k-mer chaining) found 295,388 elementary SDs grouped in 58,040 elementary SD sets. The process took around 57 minutes, and the method covers 91% of the SD basepairs (95% if we only consider regions within SDs that match, i.e., without deletions or insertions). The minimum length of an SD was set to 100bp. To gain higher coverage, we "fixed" SD mates to correct all mismatches that occurred in the matching regions. We take each SD with alignment and make sure that the region in the second mate is the same as the region in the first one by changing their differences (if differences exist). The coverage of SD regions grew by 3% (3 Mbp extra covered). Change in coverage occurs because these extra regions had higher levels of point mutations and could not be caught by our sweeping

line algorithm before "fixing" due to the point mutations causing the nonexistence of the same k-mers in the d_g range. We tried to speed up this process by using winnowing (with k-mer size set to 10 and winnow size to 14), and our basepairs coverage went down to 86%, while runtime did not improve (40 minutes).

The main downside of the k-mer chaining approach is its CPU performance. Wheres, the method based on union-find took ≈1 minute on hg19, the k-mer based approach took 57 minutes. However, the k-mer chaining approach results were more amenable to analysis because the union-find method heavily depends on the given alignments—which can be inconsistent—creating loops where many elementary SD sets get merged into one big, especially in the regions of high SD density. Thus, we used the results obtained by the k-mer chaining method to track evolutionary history and analyze SD regions, enabling us to get phylogenetic trees.

Table 3.4: Comparison between two methods used for an SD decomposition on hg19 SDs. It is noticeable that the union-find approach has much better CPU performance; however, it merges most of its elementary SDs into one elementary SD "super" set due to the inconsistent alignments causing loops.

	Union-find	**k-mer chaining**
Time	0:00:17	0:57:02
Coverage of SD regions (%)	87%	91%
Size of the longest elementary SD set	329,373	198
Number of elementary SD sets	66,891	58,040
Number of elementary SDs	550,956	295,388

3.2.2 Phylogenetic analysis

To obtain a phylogenetic tree of the SD regions, similarity relations between SDs, or regions that contain SDs, must be defined. For that purpose, we calculate similarities between each pair of SDs by comparing elementary SDs that they are composed of. Past work provides phylogenetic trees between regions in the LCR16 (low copy repeat on chromosome 16) based on SDs they contain [10]. Many SDs in LCR16 overlap the *NPIP* (nuclear pore interacting protein) family of genes, indicating the evolutionary importance of those SDs. Elementary SDs (obtained using A-Bruijn graphs [35]) were utilized to build phylogenetic trees between different regions (that correspond to different genes, e.g., *NPIP* gene family in the human genome) in multiple species. For comparison, we extracted coordinates of *NPIP* specific genes, transferred them

to hard-masked coordinates, and created a phylogenetic tree from our elementary SDs covering these genes. Distances between regions were calculated as $d(s_1, s_2) = 1 - \mathcal{J}(s_1, s_2)$, where $\mathcal{J}$ is Jaccard similarity between two sets of elementary SDs (every region consist of a certain number of elementary SDs due to the rich occurrences of SDs in those regions).

While SDquest produces (for one genome) SDs and mosaic SDs composed of indexes of elementary SDs, those indexes do not give us the information on the exact coordinates of each elementary SD needed for tree reconstruction. For that reason, we were not able to compare our results with to SDquest.

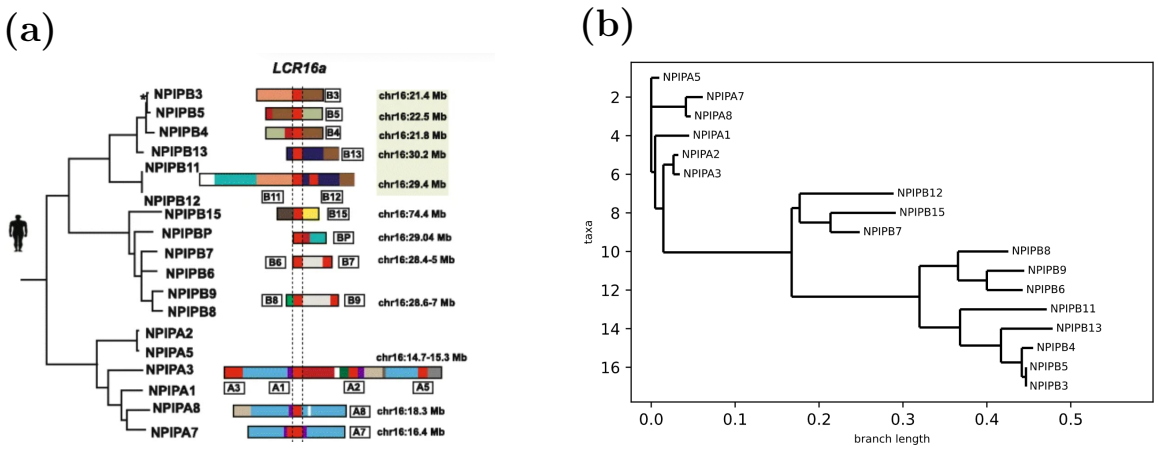


Figure 3.2: **(a)** Pyhlogenetic tree of *NPIP* regions as reported in [28]. **(b)** A phylogenetic tree of *NPIP* regions is built comparing similarities based on elementary SDs obtained using the k-mer chaining. Both were obtained using the neighbor-joining method and *NPIP* genes as leaf taxa. The difference in trees is due to the fact that we find more elementary SDs and can calculate their similarities more precisely. Note that *NPIPA* and *NPIPB* are still distinguished separately.

3.3 Multi-genome results

In addition to running BISER on a single genome, we also ran BISER on the following seven related genomes:

- *C.jacchus* (marmoset, version calJac3),
- *M.mulatta* (macaque, version rheMac10),
- *G.gorilla* (gorilla, version gorGor6),

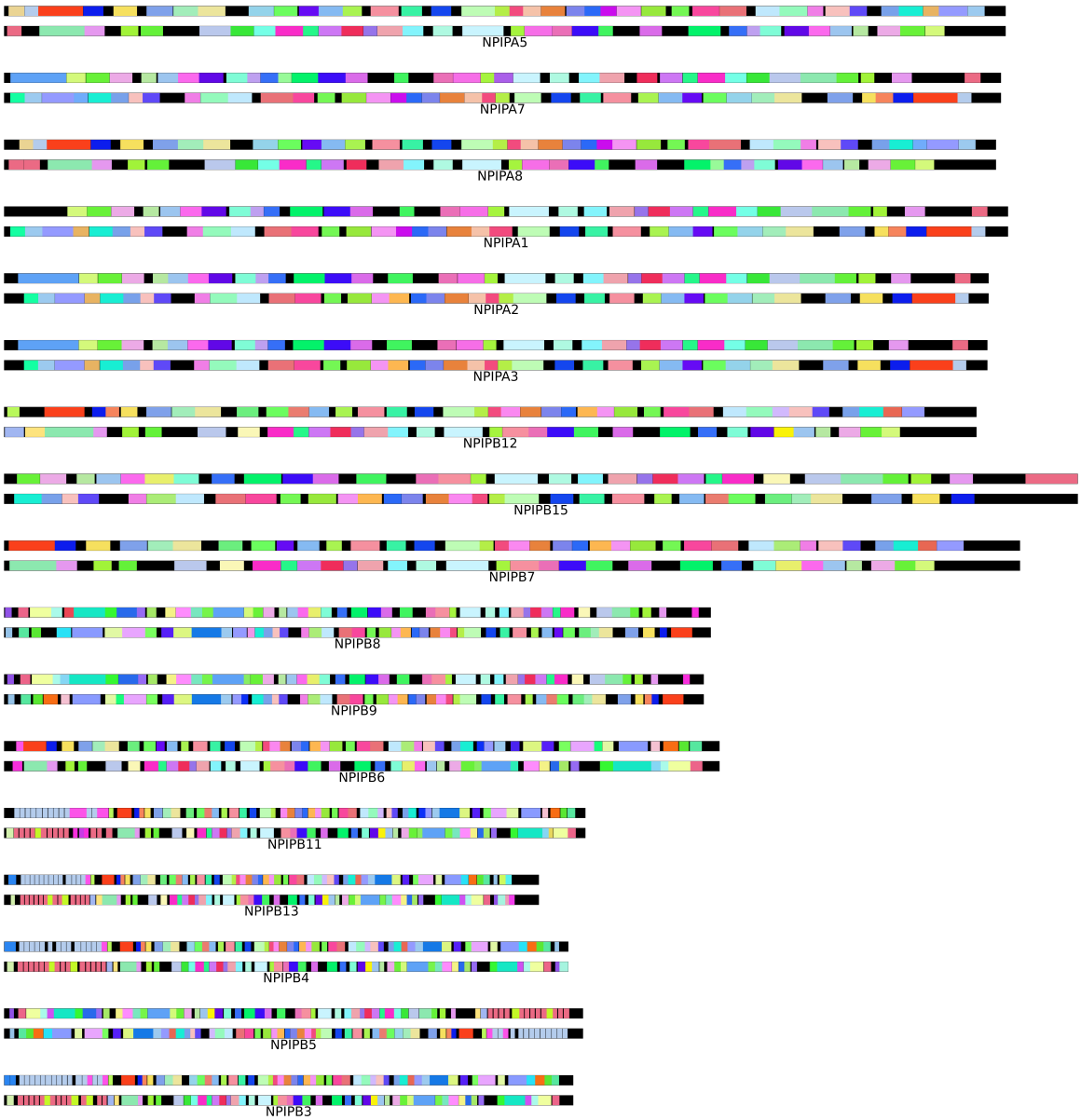


Figure 3.3: Decomposition results on hg19 SDs using the k-mer chaining method. Each color represents a different elementary set. If we compare it to elementary SDs, previously reported [28], we can see that we find many more elementary SDs, which can give us a better picture of how these SDs came to be. For every label of the *NPIP* gene, we show normal and reverse complement elementary SDs that are found in those regions. Black sections are gaps. Sequences are sorted as in the Figure 3.3b.

- *P.abelii* (orangutan, version ponAbe3),
- *P.troglodytes* (chimpanzee, version panTro6),
- *H.sapiens* (human, version hg19), and

- *M.musculus* (mouse, version mm8).

These genomes were analyzed in the previous work [10], with the sole exception of *M.musculus* that is novel to this analysis.

BISER took around four hours to complete the run on 8 cores. Of that, it took around 1 hour and 25 minutes to find putative SDs within and between species (20 minutes for in-species detection and 1 hour for detecting SDs conserved across different species). The remaining time (2h 29m) was spent calculating the final alignments for all reported SDs (Table 3.2). The vast majority of alignment time (1h 37m minutes out of 2h 29m) was spent only on aligning putative SDs from calJac3 genome. We presume that this is due to the high presence of unmasked low-complexity regions in this particular assembly.

The SD decomposition (the first approach using the union-find method) and core duplicon detection took slightly less than 6 minutes to complete on a set of nearly 2,407,000 SDs. BISER found ≈116,000 elementary SD sets seeded by ≈13,000 cores. Previously reported cores [28, 10] were covered by BISER's cores as well.

Chapter 4

Conclusion

More than a decade ago, the Genome 10K Project Consortium proposed to build genome assemblies for 10,000 species [19]. Due to the lack of high-quality long-read sequencing data, this aim was not immediately realized. However, the Genome 10K Project spearheaded the development of other large-scale many-genome sequencing projects such as the Earth BioGenome Project [30] and Vertebrate Genomes Project[1]. Recent developments in generating more accurate long-read sequencing data, coupled with better algorithms to assemble genomes now promise to make the aforementioned and similar projects feasible.

Analyzing the recently and soon-to-be generated genome assemblies to understand evolution requires the development of various algorithms for different purposes, from gene annotation [39] to orthology analysis [23] and the selection and recombination analysis [24]. Although a handful of tools such as SEDEF and SDquest are now available to characterize segmental duplications in genome assemblies, they cannot perform multi-species SD analysis, and they suffer from computational requirements. We developed BISER as a new segmental duplication characterization algorithm to be added to the arsenal of evolution analysis tools.

We demonstrate that (1) BISER is substantially faster than earlier tools; (2) it can characterize SDs in multiple genomes to delineate the evolutionary history of duplications; and (3) it can identify elementary SDs and core duplicons to help understand the mechanisms that give rise to SDs. We believe that BISER will be a powerful and common tool and will contribute to our understanding of SD evolution when thousands of genome assemblies become available in the next few years.

[1] `https://vertebrategenomesproject.org/`

The next step in this line of research would consist of interpreting BISER's results, biological analysis of the reported core duplicons, and applying BISER to a larger set of available mammalian genomes to infer the evolutionary history of ancient duplications. We also plan to further investigate the bias of ordered Jaccard similarity ($\hat{\mathcal{J}}$) when applied to different genomes. Finally, we plan to incorporate an A-Bruijn graph method for decomposition and to automate phylogenetic tree analysis.

Appendix A

Additional Information

For the easier reproducibility of the code, parameter values that are used will be presented below. For the search part of the algorithm, we used:

1. $k = 14$ – k-mer size
2. $w = 16$ – winnow size
3. $d = 250$ – maximal distance between two k-mers in one putative SD
4. $\varepsilon_P = 0.15$ – edit error for point mutations
5. $\varepsilon_B = 0.1$ – edit error for indels
6. $query\_threshold = 300$ – size of minimal query mate
7. $ref\_threshold = 100$ – size of minimal reference mate
8. $pad = 5{,}000$ – padding
9. $INDEX\_CUTOFF = 0.001$ – parameter determining how many k-mers to filter

For the align part of the algorithm, we used:

1. $k = 10$ – k-mer size

For the decomposition (k-mer chaining) part of the algorithm, we used:

1. $k = 10$ – k-mer size
2. $d_g = 50$ – maximal distance between two k-mers in one putative SD
3. $\mu = 100$ – minimal length of each LMSA

Table A.1: Coverage differences between BISER, SDquest, WGAC and SEDEF. SEDEF was run on hard-masked genomes and on soft-masked genomes. As default BISER mode, we use the one where minimal putative SD length in reference is 500bp and 100bp in query.

	SEDEF hard	SEDEF	SEDEF hard	SEDEF normal
BISER	1,425,245	7,119,758	1,070,867	5,103,255
SDquest	2,358,112	1,867,555	4,950,689	3,882,440
SDquest & BISER	473,814	1,355,351	581,316	1,652,880
SDquest & SEDEF	156,321	646,878	282,113	1,350,362
SDquest & SEDEF & BISER	16,492,130	15,610,593	18,716,198	17,644,634
SEDEF	1,440,158	3,536,302	1,349,689	15,500,908
SEDEF & BISER	16,255,088	10,560,575	14,236,853	10,204,465
WGAC	215,951	34,412	407,733	43,761
WGAC & BISER	20,966	47,340	47,113	43,736
WGAC & SDquest	217,737	6,209	344,418	8,720
WGAC & SDquest & BISER	15,386	31,376	24,731	43,214
WGAC & SDquest & SEDEF	8,648	220,176	14,607	350,305
WGAC & SDquest & SEDEF & BISER	75,492,462	75,476,472	57,566,495	57,548,012
WGAC & SEDEF	3,115	184,654	19,166	383,138
WGAC & SEDEF & BISER	2,214,306	2,187,932	2,192,733	2,196,110
BISER	112,389,397	112,389,397	94,436,306	94,436,306
SEDEF	112,062,228	108,423,582	94,377,854	105,177,934
SDquest	95,214,610	95,214,610	82,480,567	82,480,567
WGAC	78,188,571	78,188,571	60,616,996	60,616,996

Table A.2: Runtime performance and coverage of the search step on human and mouse genomes (hg19 and mm8). Time does not change by changing minimal length for reference and query SD paralogs, but coverage does.

Underlined: the best BISER mode	*hg19; k-mer size: 14; winnow size: 16*				
Modes:	8-core time	1-core time	Memory (MB)	Number of regions	Coverage
BISER (gap: 250, 500/100)	0:02:33	0:19:03	1,892	179,045	1,074,902,081
BISER (gap: 250, 700/100)	0:02:46	0:19:22	1,998	179,010	1,074,874,528
BISER (gap: 250, 750/150)	0:02:39	0:19:42	1,970	138,886	938,496,092
BISER (gap: 250, 1000/300)	0:02:38	0:19:33	1,970	53,289	417,445,540
	mm8; k-mer size: 14; winnow size: 16				
Modes:	8-core time	1-core time	Memory (MB)	Number of regions	Coverage
BISER (gap: 250, 500/100)	0:02:38	0:19:24	2,242	614,954	1,043,583,290
BISER (gap: 250, 700/100)	0:02:35	0:19:58	2,140	614,702	1,043,576,833
BISER (gap: 250, 750/150)	0:02:33	0:18:57	2,051	483,238	901,791,090
BISER (gap: 250, 1000/300)	0:02:26	0:18:33	2,034	211,677	396,957,419

Table A.3: Runtime performance and coverage of align step on human and mouse genomes (hg19 and mm8). Coverage in search step does not significantly affect align step due to the false positives being quickly discarded.

hg19:	8-core time	1-core time	Memory (MB)	Number of regions	Coverage
BISER (gap: 250, 500/100)	0:04:16	0:24:31	5,827	57,167	112,713,192
BISER (gap: 250, 700/100)	0:04:28	0:25:35	5,823	57,155	112,713,192
BISER (gap: 250, 750/150)	0:04:05	0:23:23	5,817	54,398	111,241,079
BISER (gap: 250, 1000/300)	0:03:35	0:20:13	5,796	44,708	106,858,478
mm8:	8-core time	1-core time	Memory (MB)	Number of regions	Coverage
BISER (gap: 250, 500/100)	0:08:31	0:48:51	6,925	167,002	94,712,544
BISER (gap: 250, 700/100)	0:08:25	0:48:05	6,925	166,957	94,712,513
BISER (gap: 250, 750/150)	0:07:33	0:42:02	6,917	156,758	93,339,973
BISER (gap: 250, 1000/300)	0:06:24	0:33:40	6,898	120,267	89,781,159

Table A.4: Number of SDs and coverage (in basepairs) found within and between seven species.

Number of regions	calJac3	gorGor6	hg19	mm8	panTro6	ponAbe3	rheMac10
calJac3	603,730	75,650	86,308	21,509	81,343	78,358	81,070
gorGor6	75,650	52,068	95,310	13,564	88,222	79,039	78,602
hg19	86,308	95,310	58,430	14,554	111,670	91,553	94,177
mm8	21,509	13,564	14,554	171,883	14,573	13,426	13,741
panTro6	81,343	88,222	111,670	14,573	53,804	84,497	85,255
ponAbe3	78,358	79,039	91,553	13,426	84,497	38,269	80,873
rheMac10	81,070	78,602	94,177	13,741	85,255	80,873	46,177

Coverage (Mbp)	calJac3	gorGor6	hg19	mm8	panTro6	ponAbe3	rheMac10
calJac3	85,389,375	102,691,375	149,503,760	11,276,903	124,192,147	107,723,359	101,574,889
gorGor6	102,691,375	68,486,659	146,744,933	11,702,009	134,753,741	125,639,802	124,010,586
hg19	149,503,760	146,744,933	112,389,397	13,539,321	198,007,818	174,021,263	182,566,608
mm8	11,276,903	11,702,009	13,539,321	94,436,306	12,687,666	12,074,106	11,066,473
panTro6	124,192,147	134,753,741	198,007,818	12,687,666	92,931,641	135,301,770	131,705,094
ponAbe3	107,723,359	125,639,802	174,021,263	12,074,106	135,301,770	71,912,746	130,643,146
rheMac10	101,574,889	124,010,586	182,566,608	11,066,473	131,705,094	130,643,146	72,121,852

Bibliography

[1] Mohamed Ibrahim Abouelhoda and Enno Ohlebusch. Multiple genome alignment: Chaining algorithms revisited. In Ricardo Baeza-Yates, Edgar Chávez, and Maxime Crochemore, editors, *Combinatorial Pattern Matching*, pages 1–16. Springer Berlin Heidelberg, 2003.

[2] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *J Mol Biol*, 215(3):403–410, Oct 1990.

[3] Shanika L. Amarasinghe, Shian Su, Xueyi Dong, Luke Zappia, Matthew E. Ritchie, and Quentin Gouil. Opportunities and challenges in long-read sequencing data analysis. *Genome Biology*, 21:30, 2020.

[4] A. Andoni, R. Krauthgamer, and K. Onak. Polylogarithmic approximation for edit distance and the asymmetric query complexity. In *Proc. IEEE 51st Annual Symp. Foundations of Computer Science*, pages 377–386, October 2010.

[5] Francesca Antonacci, Jeffrey M Kidd, Tomas Marques-Bonet, Brian Teague, Mario Ventura, Santhosh Girirajan, Can Alkan, Catarina D Campbell, Laura Vives, Maika Malig, Jill A Rosenfeld, Blake C Ballif, Lisa G Shaffer, Tina A Graves, Richard K Wilson, David C Schwartz, and Evan E Eichler. A large and complex structural polymorphism at 16p12.1 underlies microdeletion disease risk. *Nat Genet*, 42(9):745–750, Sep 2010.

[6] Arturs Backurs and Piotr Indyk. Edit distance cannot be computed in strongly subquadratic time (unless SETH is false). In *Proceedings of the Forty-seventh Annual ACM Symposium on Theory of Computing*, STOC '15, pages 51–58, New York, NY, USA, 2015. ACM.

[7] J. A. Bailey, J. M. Kidd, and E. E. Eichler. Human copy number polymorphic genes. *Cytogenet Genome Res*, 123(1-4):234–243, 2008.

[8] J. A. Bailey, A. M. Yavor, H. F. Massa, B. J. Trask, and E. E. Eichler. Segmental duplications: organization and impact within the current human genome project assembly. *Genome Res*, 11(6):1005–1017, Jun 2001.

[9] Jeffrey A Bailey and Evan E Eichler. Primate segmental duplications: crucibles of evolution, diversity and disease. *Nat Rev Genet*, 7(7):552–564, Jul 2006.

[10] Stuart Cantsilieris, Susan M. Sunkin, Matthew E. Johnson, Fabio Anaclerio, John Huddleston, Carl Baker, Max L. Dougherty, Jason G. Underwood, Arvis Sulovari, PingHsun Hsieh, Yafei Mao, Claudia Rita Catacchio, Maika Malig, AnneMarie E. Welch, Melanie Sorensen, Katherine M. Munson, Weihong Jiang, Santhosh Girirajan, Mario Ventura, Bruce T. Lamb, Ronald A. Conlon, and Evan E. Eichler. An evolutionary driver of interspersed segmental duplications in primates. *Genome biology*, 21:202, 2020.

[11] Keegan Carruthers-Smith. Sliding window minimum implementations, 2013. last accessed 28 January 2021.

[12] Vasek Chvatal. A greedy heuristic for the set-covering problem. *Mathematics of operations research*, 4(3):233–235, 1979.

[13] Jean-Félix Dallery, Nicolas Lapalu, Antonios Zampounis, Sandrine Pigné, Isabelle Luyten, Joëlle Amselem, Alexander H. J. Wittenberg, Shiguo Zhou, Marisa V. de Queiroz, Guillaume P. Robin, Annie Auger, Matthieu Hainaut, Bernard Henrissat, Ki-Tae Kim, Yong-Hwan Lee, Olivier Lespinet, David C. Schwartz, Michael R. Thon, and Richard J. O'Connell. Gapless genome assembly of colletotrichum higginsianum reveals chromosome structure and association of transposable elements with secondary metabolite gene clusters. *BMC genomics*, 18:667, 2017.

[14] Franklin Delehelle, Sylvain Cussat-Blanc, Jean-Marc Alliot, Hervé Luga, and Patricia Balaresque. ASGART: fast and parallel genome scale segmental duplications mapping. *Bioinformatics*, 34:2708–2714, 2018.

[15] Max L. Dougherty, Jason G. Underwood, Bradley J. Nelson, Elizabeth Tseng, Katherine M. Munson, Osnat Penn, Tomasz J. Nowakowski, Alex A. Pollen, and Evan E. Eichler. Transcriptional fates of human-specific segmental duplications in brain. *Genome research*, 28:1566–1576, 2018.

[16] Ian T. Dowsett, Jessica L. Sneeden, Branden J. Olson, Jill McKayFleisch, and Scott R. Kennedy & Alan J. Herr Emma McAuley. idowsett/asymmetric-segregation-of-polymerase-errors-and-rate-volatility-diversify-mutation-burden: Pre-publication release v1.1. Jan 2021.

[17] John W. Drake, Brian Charlesworth, Deborah Charlesworth, and James F. Crow. Rates of spontaneous mutation. *Genetics*, 148(4):1667–1686, 1998.

[18] Huan Fan, Anthony R Ives, Yann Surget-Groba, and Charles H Cannon. An assembly and alignment-free method of phylogeny reconstruction from next-generation sequencing data. *BMC genomics*, 16:522, July 2015.

[19] Genome 10K Community of Scientists. Genome 10K: a proposal to obtain whole-genome sequence for 10,000 vertebrate species. *J Hered*, 100(6):659–674, 2009.

[20] Santhosh Girirajan, Megan Y. Dennis, Carl Baker, Maika Malig, Bradley P. Coe, Catarina D. Campbell, Kenneth Mark, Tiffany H. Vu, Can Alkan, Ze Cheng, Leslie G. Biesecker, Raphael Bernier, and Evan E. Eichler. Refinement and discovery of new hotspots of copy-number variation associated with autism spectrum disorder. *Am J Hum Genet*, 92(2):221–237, Feb 2013.

[21] Hiroyuki Hanada, Mineichi Kudo, and Atsuyoshi Nakamura. On practical accuracy of edit distance approximation algorithms. *arXiv preprint arXiv:1701.06134*, 2017.

[22] Robert S. Harris. *Improved Pairwise Alignment of Genomic Dna*. PhD thesis, Pennsylvania State University, University Park, PA, USA, 2007. AAI3299002.

[23] Xiao Hu and Iddo Friedberg. SwiftOrtho: A fast, memory-efficient, multiple genome orthology classifier. *GigaScience*, 8, October 2019.

[24] Martin Hölzer and Manja Marz. PoSeiDon: a Nextflow pipeline for the detection of evolutionary recombination events and positive selection. *Bioinformatics*, July 2020.

[25] Chirag Jain, Alexander Dilthey, Sergey Koren, Srinivas Aluru, and Adam M. Phillippy. A fast approximate algorithm for mapping long reads to large reference databases. In S. Cenk Sahinalp, editor, *Proceedings of 21st Annual International Conference on Research in Computational Molecular Biology (RECOMB 2017)*, volume 10229, pages 66–81, Cham, 2017. Springer International Publishing.

[26] Chirag Jain, Sergey Koren, Alexander Dilthey, Adam M Phillippy, and Srinivas Aluru. A fast adaptive algorithm for computing whole-genome homology maps. *Bioinformatics*, 34(17):i748–i756, 2018.

[27] Zhaoshi Jiang, Robert Hubley, Arian Smit, and Evan E. Eichler. Dupmasker: a tool for annotating primate segmental duplications. *Genome research*, 18:1362–1368, August 2008.

[28] Zhaoshi Jiang, Haixu Tang, Mario Ventura, Maria Francesca Cardone, Tomas Marques-Bonet, Xinwei She, Pavel A Pevzner, and Evan E Eichler. Ancestral reconstruction of segmental duplications reveals punctuated cores of human genome evolution. *Nature genetics*, 39:1361–1368, November 2007.

[29] Vladimir Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966.

[30] Harris A. Lewin, Gene E. Robinson, W. John Kress, William J. Baker, Jonathan Coddington, Keith A. Crandall, Richard Durbin, Scott V. Edwards, Félix Forest, M. Thomas P. Gilbert, Melissa M. Goldstein, Igor V. Grigoriev, Kevin J. Hackett, David Haussler, Erich D. Jarvis, Warren E. Johnson, Aristides Patrinos, Stephen Richards, Juan Carlos Castilla-Rubio, Marie-Anne van Sluys, Pamela S. Soltis, Xun Xu, Huanming Yang, and Guojie Zhang. Earth BioGenome Project: Sequencing life for the future of life. *Proceedings of the National Academy of Sciences of the United States of America*, 115:4325–4333, April 2018.

[31] Heng Li. Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics (Oxford, England)*, 34:3094–3100, September 2018.

[32] Tomas Marques-Bonet, Jeffrey M Kidd, Mario Ventura, Tina A Graves, Ze Cheng, LaDeana W Hillier, Zhaoshi Jiang, Carl Baker, Ray Malfavon-Borja, Lucinda A Fulton, Can Alkan, Gozde Aksay, Santhosh Girirajan, Priscillia Siswara, Lin Chen, Maria Francesca Cardone, Arcadi Navarro, Elaine R Mardis, Richard K Wilson, and Evan E Eichler. A burst of segmental duplications in the genome of the African great ape ancestor. *Nature*, 457(7231):877–881, Feb 2009.

[33] Guillaume Marçais, Arthur L. Delcher, Adam M. Phillippy, Rachel Coston, Steven L. Salzberg, and Aleksey Zimin. MUMmer4: A fast and versatile genome alignment system. *PLoS computational biology*, 14:e1005944, January 2018.

[34] Ibrahim Numanagić, Alim S Gökkaya, Lillian Zhang, Bonnie Berger, Can Alkan, and Faraz Hach. Fast characterization of segmental duplications in genome assemblies. *Bioinformatics*, 34:i706–i714, September 2018.

[35] Glenn Tesler P. A. Pevzner, Haixu Tang. De novo repeat classification and fragment assembly. *Genome Research*, 14(9):1786–1796, Sep 2004.

[36] Lianrong Pu, Yu Lin, and Pavel A Pevzner. Detection and analysis of ancient segmental duplications in mammalian genomes. *Genome research*, 28:901–909, June 2018.

[37] Saul Schleimer, Daniel S Wilkerson, and Alex Aiken. Winnowing: local algorithms for document fingerprinting. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 76–85. ACM, 2003.

[38] Ariya Shajii, Ibrahim Numanagić, Riyadh Baghdadi, Bonnie Berger, and Saman Amarasinghe. Seq: A high-performance language for bioinformatics. *Proc. ACM Program. Lang.*, 3, October 2019.

[39] Alaina Shumate and Steven L. Salzberg. Liftoff: accurate mapping of gene annotations. *Bioinformatics*, December 2020.

[40] Peter H Sudmant, Jacob O Kitzman, Francesca Antonacci, Can Alkan, Maika Malig, Anya Tsalenko, Nick Sampas, Laurakay Bruhn, Jay Shendure, 1000 Genomes Project, and Evan E Eichler. Diversity of human copy number variation and multicopy genes. *Science*, 330(6004):641–646, Oct 2010.

[41] Hajime Suzuki and Masahiro Kasahara. Introducing difference recurrence relations for faster semi-global alignment of long sequences. *BMC bioinformatics*, 19(1):33–47, 2018.

[42] O. Tange. GNU Parallel - the command-line power tool. *;login: The USENIX Magazine*, 36(1):42–47, Feb 2011.

[43] Robert Endre Tarjan. A class of algorithms which require nonlinear time to maintain disjoint sets. *J. Comput. Syst. Sci.*, 18(2):110–127, 1979.