Fast Characterization of Segmental Duplications in Genome Assemblies

Ibrahim Numanagić 1,2, Alim S. Gökkaya 3, Lillian Zhang 1, Bonnie Berger 1,2, Can Alkan 3,* and Faraz Hach 4,5, *  

1 Computer Science and AI Lab, Massachusetts Institute of Technology, Cambridge 02139, MA, USA  
2 Department of Mathematics, Massachusetts Institute of Technology, Cambridge 02139, MA, USA  
3 Department of Computer Engineering, Bilkent University, 06800, Ankara, Turkey  
4 Vancouver Prostate Centre, V6H 3Z9, Vancouver, Canada  
5 Department of Urologic Sciences, University of British Columbia, V5Z 1M9, Vancouver, Canada  

ABSTRACT  
Motivation: Segmental duplications (SDs), or low-copy repeats (LCR), are segments of DNA greater than 1 Kbp with high sequence identity that are copied to other regions of the genome. SDs are among the most important sources of evolution, a common cause of genomic structural variation, and several are associated with diseases of genomic origin including schizophrenia and autism. Despite their functional importance, SDs present one of the major hurdles for de novo genome assembly due to the ambiguity they cause in building and traversing both state-of-the-art overlap-layout-consensus and de Bruijn graphs. This causes SD regions to be misassembled, collapsed into a unique representation, or completely missing from assembled reference genomes for various organisms. In turn, this missing or incorrect information limits our ability to fully understand the evolution and the architecture of the genomes. Despite the essential need to accurately characterize SDs in assemblies, there is only one tool that has been developed for this purpose, called Whole Genome Assembly Comparison (WGAC); its primary goal is SD detection. WGAC is comprised of several steps that employ different tools and custom scripts, which makes this strategy difficult and time consuming to use. Thus there is still a need for algorithms to characterize within-assembly SDs quickly, accurately, and in a user friendly manner.  
Results: Here we introduce a SEgmental Duplication Evaluation Framework (SEDEF) to rapidly detect SDs through sophisticated filtering strategies based on Jaccard similarity and local chaining. We show that SEDEF accurately detects SDs while maintaining substantial speed up over WGAC which translates into practical run times of minutes instead of weeks. Notably, our algorithm captures up to 25% "pairwise error" between segments, whereas previous studies focused on only 10%, allowing us to more deeply track the evolutionary history of the genome.  
Availability: SEDEF is available at https://github.com/vpc-ccg/sedef  

1 INTRODUCTION  
Segmental duplications (SDs) are defined as genomic segments of size greater than 1 Kbp that are repeated within the genome with at least 90% sequence identity (Bailey et al., 2001) in either tandem or interspersed organization. Almost all genomes harbor large SDs; for example the build 37 release of the human reference genome (GRCh37) contains a total of 159 Mbp gene-rich duplicated sequence, which corresponds to approximately 5.5% of its entire length1. It is known that SDs played a major role in evolution (Marques-Bonet et al., 2009; Prado-Martinez et al., 2013; Sudmant et al., 2013), and are one of the most important factors that contribute to human disease either directly (Gonzalez et al., 2005; Yang et al., 2007; Hollox et al., 2008), or through leading to other forms of structural variation (Alkan et al., 2011a; Mills et al., 2011). Furthermore, human populations show SD diversity that may be used as markers for population genetics studies (Alkan et al., 2009; Sudmant et al., 2010).  

Despite their functional importance, SDs are poorly characterized due to the difficulties they impose on constructing accurate genome assemblies (Alkan et al., 2011b; Chaisson et al., 2015; Steinberg et al., 2017), as well as the ambiguities in read mapping (Treangen and Salzberg, 2012; Firtina and Alkan, 2016). Building inaccurate assemblies due to SDs is an important problem, since it may lead to potentially incorrect conclusions about the evolution of the species of interest as in the case of the giant panda assembly (Li et al., 2010). In the giant panda genome analysis, the authors concluded that the panda genome included substantially less repeats and duplications compared to other mammalian genomes, however, it is shown that this result is likely incorrect due to mis-assembly of these regions. Additionally, duplications give rise to gaps in the assembly, which likely contain genes and other functionally active regions (Bailey et al., 2001, 2002; Alkan et al., 2011b). Accurate assembly of duplicated regions remains a difficult and unsolved problem, which may be ameliorated through the use of ultra-long reads generated by the Oxford Nanopore platform (Jain et al., 2017b) or Linked-Read sequencing (e.g., 10x Genomics (Mostovoy et al., 2016; Yeo et al., 2017)) if the duplicated segment is shorter than the read or linked-read length, respectively. However, characterization of the SD content in existing assemblies is still important for two reasons: (i) to evaluate the “completeness” of these genome assemblies, and (ii) understand genome evolution for comparative genomics studies.  

1 http://humanparalogy.gs.washington.edu/build37/build37.htm
Segmental duplications (SDs) are generated by large-scale copy events that have occurred during the evolution of the genome. After such a copy event, both sites involved in SD may have undergone a number of changes during the evolutionary history of the genome. Formally, consider a genomic sequence $G = g_1g_2g_3\ldots g_{|G|}$ of length $|G|$, where $g_i \in \{A,C,G,T\}$ for any $i$. Let $G_{i+\delta} = g_{i+\delta}, \ldots, g_{i+n-\delta}$ be a substring in $G$ of length $n$ that starts at position $i$. Furthermore, let $X_k$ be the set of all $k$-mers in the substring $G_{i+\delta}$. We assume that $k$ is predefined and fixed.

The Levenshtein (Levenshtein, 1966) *edit distance* is defined as: $ed(G_{i+\delta}, G_{j+\delta})$ of two substrings $G_{i+\delta}$ and $G_{j+\delta}$ (further simplified as $G_i$ and $G_j$) to be a minimal number of edit operations (i.e., single nucleotide substitutions, insertions, and deletions) that are needed to convert the string $G_i$ into $G_j$. The length of the alignment between $G_i$ and $G_j$ is denoted as $l$, and clearly $l \geq \max(m,n)$. Let us define the notion of *edit error* (further referred to as *just error*) between two strings $G_i$ and $G_j$ as $err(G_i,G_j) = ed(G_i,G_j)/l$—the edit distance normalized over alignment length. Intuitively, this is the average number of the edits needed to turn $G_i$ into $G_j$. Clearly, two strings are identical if $err(G_i,G_j) = 0$. We consider $G_i$ and $G_j$ as a *segmental duplication* of length $l$ with error $\delta$ if the following SD conditions are met:

- $l \geq 1,000$ where $l$ is the length of alignment between $G_i$ and $G_j$,
- $err(G_i,G_j) \leq \delta$.

We also assume that the overlap between the $G_i$ and $G_j$ in the genome is at most $\delta \cdot n$.

### 2.1 Edit distance model

Each segmental duplication is generated by a past structural variation (SV) event that copied a substring of length $n$ in $G$ from locus $i$ to locus $j$. This copy was initially perfect, meaning that corresponding strings $G_i$ and $G_j$ (initially both of length $n$) were identical. However, various changes during the evolutionary history of the genome—such as point mutations, small indels, and other structural variants—have altered both the original and duplicate strings independently. Thus it is necessary to take such changes into consideration when identifying the potential SDs. While previous SD studies focused only on SDs with pairwise error at most 10%, here we...
aim to focus on SDs whose error rate can go up to 25% (in another words, δ ≤ 1/4). Higher δ allows us to track the evolutionary history of the human genome to earlier periods. However, it also significantly renders the SD detection problem more difficult since the majority of the known filtering techniques that operate on the edit distance metric space assume much lower values of δ (Andoni et al., 2010). We address this challenge by leveraging our knowledge from the biology of human and other genomes that different mutation events contribute unequally to the total value of δ in order to better optimize our SD detection algorithms.

A key conceptual advance of our work that helps model such events in the human genome is to separately consider germline mutation rates (denoted as small mutations), and de novo SV rates. It is estimated that the substitution rate in the human genome is roughly 0.5 × 10⁻⁹ per basepair per year (Scally, 2016), and we may assume a similar rate for other mammalian genomes. The evolutionary split of the human and chimpanzee species is estimated to have occurred approximately 7 million years ago (Hedges and Kumar, 2009). Thus, we expect that the probability of a basepair being mutated since the split is roughly 3.5 × 10⁻³. If we also account for small indels (with an even smaller mutation rate than those of substitutions (Montgomery et al., 2013)), the edit error between any two paralogs of an SD that have occurred after the evolutionary split is not larger than 0.1%. Even if we consider SDs that occurred much earlier in history (e.g., after the lowest common ancestor of human and mouse roughly 90 million years ago), the total edit error will not be larger than 10%. However, the edit error between two paralogs of an SD can be much larger due to large structural variations. One such example is insertion of transposons within an SD. These events can be visualized as large gaps within an edit distance string of two segmental duplications which contribute a large share towards the total edit error (Figure 1)⁵. Thus we assume that small mutations contribute at most δ₂ ≤ 0.15 towards the total edit error δ (both paralogs can be mutated up to 7.5%; this default setting is higher than the estimated rates (above) for human and mouse genomes in order to be able to handle older species as well. Analogously, large-scale events (subsequently referred to as gaps) contribute the remaining δ₃ = δ − δ₂ of the edit error. We assume that the probability of a large gap occurring at any basepair in the genome is not larger than 0.005 (as estimated by analysis of existing human SDs; similar value can be derived for other species). Note that the gap penalty is typically calculated via affine gap model, where gap openings are heavily penalized while gap extensions are either ignored or assigned a very low penalty. Many human SDs have δ₂ ≥ 0.15 if calculated by standard Levenshtein distance metric. SEDEF uses standard (Levenshtein) gap distance metric while locating seed SDs (meaning all seed SDs have δ₂ ≤ 0.15); however, this restriction is lifted in a later step where we switch to the affine gap penalty.

Furthermore, we assume that the mutations within SD paralogs follow a Poisson error model (Jain et al., 2017a; Fan et al., 2015), and that those mutations occur independently of each other. It follows that any k-mer in X₂ (the set of k-mers of G₂) has accumulated on average k · δ₂ mutations compared to the originating k-mer in X₁, provided that such a k-mer was part of the original copy. By setting a Poisson parameter λ = k · δ₂/3, we obtain the probability of an event in which a k-mer is preserved in both paralogs of an SD (i.e. that it is error-free): Pr(δ = 0 | λ) = e⁻ᵏδλ.⁶

We call this error model SD error model, and assume that any SD of interest satisfies the error constraints mentioned above.

### 2.2 Jaccard similarity

Suppose that we ask whether two substrings G₁ and G₂ are similar to each other, where the length of both strings is n. One way to measure the similarity of these substrings is to analyze their respective k-mer sets X₁ and X₂ and to count the number of shared k-mers between them. This metric is known as Jaccard similarity of sets X₁ and X₂, and is formally defined as

\[
J(X₁, X₂) = \frac{|X₁ ∩ X₂|}{|X₁ ∪ X₂|}
\]

Clearly, higher similarity of strings G₁ and G₂ implies a larger value of J(X₁, X₂).

The calculation of Jaccard similarity between two sets can be approximated via the MinHash technique developed by Broder (Broder, 1997), who proved that given a universe of all k-mers U and a random permutation h on U (typically a hash function with no collisions), it follows that

\[
J(X₁, X₂) = Pr(h_{min}(X₁) = h_{min}(X₂))
\]

where h_{min}(T) is the minimal member of T with respect to h. Furthermore, J(X₁, X₂) can be estimated and efficiently computed by calculating

\[
\frac{|S(X₁ ∩ X₂) \cap S(X₁) \cap S(X₂)|}{|S(X₁ ∪ X₂)|}
\]

where S(X₁) is the sketch of X₁ and stands for a subset of s elements from X₁, whose hash values are minimal with respect to the hash function h (such elements are called minimizers of set X₁). This estimate is unbiased as long as h is random, and its accuracy depends on the sketch size s. In practice s is much smaller than |X₁|, calculating a MinHash estimate is substantially faster compared to the calculation of J(X₁, X₂).

The performance of the MinHash technique can be further improved in the context of large strings, as shown by Jain et al (Jain et al., 2017b). Instead of computing S(X₁), it is possible to compute S(W(X₁)), where W(X₁) is a winnowing fingerprint of the corresponding string G₁. W(X₁) is calculated by sliding a window of size w through G₁ and by taking in each window a k-mer of minimal hash value (in case of a tie, the rightmost k-mer is selected). The expected size of W(A) for a random sequence A is 2|A|/(w + 1) (Schlemier et al., 2003). The main benefit of winnowing, aside from speeding up the construction of sketch S(A), is the fact that winnow W(A) can be computed efficiently in linear time and O(w) space in a streaming fashion with appropriate data structures (Carnuthers-Smith, 2013).

Moreover, it has been empirically shown that J(X₁, X₂) can be efficiently estimated by calculating a winnowed MinHash score of sets X₁ and X₂ (Jain et al., 2017a):

\[
\frac{|S(W(X₁) ∩ W(X₂)) \cap S(W(X₁)) \cap S(W(X₂))|}{|S(W(X₁) ∪ W(X₂))|}
\]

Furthermore, given the minimal desired value τ of Jaccard similarity between the two sets, it follows that |W(X₁) ∩ W(X₂)| ≥ s · τ (Jain et al., 2017a). This estimate can be used to efficiently filter out any sets X₁ and X₂ whose Jaccard similarity is below a given threshold with high confidence.

### 3 METHODS

The segmental duplication detection problem can be formulated as follows: find all pairs of loci (i, j) inside a genome G with i ≥ 1,000 such that (i) the edit error (i.e. divergence) between Gᵢ and Gᵢ₊₁ is at most δ; and (ii) the corresponding alignment between Gᵢ and Gᵢ₊₁ is of size at least l and is not contained within a larger alignment satisfying the SD criteria. In other words, for any pair (i, j) we aim to find a maximal valid region alignment of size l between Gᵢ and Gᵢ₊₁ that satisfies the criteria and error model of segmental duplications.

A naïve method for locating SDs within any genomic sequence G consists of locally aligning G onto itself, followed by the analysis of all acceptable paths within a local alignment matrix. However, this strategy is impractical for large |G| since the best known algorithms for optimal local alignment require O(|G|^2) time and space. Another possible approach, which we take, is to solve this problem by iterating through each pair of indices (i, j) within G and testing whether the matching Gᵢ and Gᵢ₊₁ satisfy the SD criteria through global alignment (given a fixed size n of Gᵢ and Gᵢ₊₁). While this method, if implemented naively, is still too slow for larger genomes and

⁵ We note that inversion and translocation events contribute more to sequence divergence due to incorrect alignments, but such events are rare.
Fig. 1. (Left) Simplified representation of a segmental duplication lifetime. Initially, a large-scale duplication forms an SD, at which point both the original region and the copy are identical. Then, both the original region and copy undergo various independent changes, such as large-scale deletions (in red), insertions (in blue), and small repeat insertions (in fuchsia). Finally, various germline mutations (in yellow) affect both regions. The resulting SD as seen today, defined region and the copy are identical. Then, both the original region and copy undergo various independent changes, such as large-scale deletions (in red), insertions (in blue), and small repeat insertions (in fuchsia). Finally, various germline mutations (in yellow) affect both regions. The resulting SD as seen today, defined initially we aim to find all pairs of strings $G_i$: generically called seed SD—such that the length of both strings is $n \geq 1000 \cdot (1 - \delta) = 750$, and such that $G_i$ and $G_j$ are believed to satisfy the SD criteria. We achieve this by iterating through the genome, and for each locus $i$ in the genome we can separate the error effects of the small mutations and $M_k$ as a union of two disjoint sets $X_i$ and $X_j$, respectively. Note that some repeats also increase the proportion of shared $k$-mers. Colors denote same as on Left.

Fig. 2. Step-by-step depiction of the SEDEF framework. Our contribution is highlighted above the steps in the dark gray boxes.

3.1 Identifying seed SDs

In order to verify if the strings $G_i$ and $G_j$ share error $\leq \delta$ under the SD error model, we will calculate the Jaccard similarity of their corresponding $k$-mer sets $X_i$ and $X_j$ and check if it is $\geq \tau$. For the sake of explanation, we will assume that $n = |G_i| = |G_j|$ (analogous reasoning holds if this is not the case). If $c = |X_i \cap X_j|$ (number of shared $k$-mers), and $t = n - k + 1$ (the size of sets $X_i$ and $X_j$), we can express the Jaccard similarity of those sets as $J(X_i, X_j) = c / (2t - c)$. We will also assume that no $k$-mer occurs twice in these sets; this assumption is sufficient for the calculation of lower bound below.

Simply using error $\delta$ to calculate the expected lower bound of Jaccard similarity $\tau$ is infeasible in practice due to the large value of $\delta$ in our setting. However, as noted above, differences between duplicated regions are not chosen uniformly at random, and thus we can separate the error $\delta$ into $\delta_M + \delta_G$, where $\delta_M$ is the error rate of the small mutations and $\delta_G$ is the error rate of large indels, as defined in the preliminaries. This separation of the two error rates is one of the novel contributions of our work.

If there exists a valid SD spanning $G_i$ and $G_j$, set $X_i$ can be considered as a union of two disjoint sets $X_i^G$ and $X_i^M$, where $X_i^M$ represents the $k$-mers initially copied by the SD event and might have undergone small mutations, while $X_i^G$ contains all “new” $k$-mers introduced by subsequent large events such as SVs and large indels (analogous separation applies to $X_j$ as well). In this way we can separate the effects of the small mutations and large-scale events. Ideally $X_i^G$ shares no $k$-mers with $X_i$, while $X_i^M$ shares a predetermined number of $k$-mers with $X_i$. We will use this property to choose $k$-mers with a maximal $\tau$.
while \( X_M \cap X_M' \neq \emptyset \) since we expect some shared k-mers to remain error-free after small mutations (see Figure 1 for visualization). Now let us also express \( t \) as \( t_M + t_G \), where \( t_M = |X_M' \cap X_M| \) and \( t_G = |X_M' \cap X_M| \) (note that \( |X_i| = |X_i'| \) applies \( |X_M' \cap X_M| \approx |X_M' \cap X_M| \) because the small mutations keep strings that generate \( |X_M' \cap X_M| \) and \( |X_M' \cap X_M| \) similar in size; thus we assume \( \log \) that \( |X_M' \cap X_M'| = |X_M'| \).

Let \( e/t_M \) be the ratio of \( k \)-mers that are not mutated in both \( X_M \) and \( X_M' \) (we assume that \( X_M' \cap X_M \) share no common \( k \)-mers, which is a valid assumption for a lower bound calculation). Its expected value, provided a Poisson error model introduced above, is \( E[e/t_M] = e^{-k|M|} \) (Jain et al., 2017a).

Now we proceed to estimate the minimal required Jaccard similarity \( J(X_i, X_j) \) of \( X_i \) and \( X_j \). Note that so far:

1. \( |X_i \cap X_j| = |X_i' \cap X_j'| \); 
2. \( t_G/(t_M + t_G) \leq \delta_G \Rightarrow t_G \leq t_M + \delta_G \); and 
3. \( |X_M' \setminus X_M| = 2t_G \) because \( |X_M'| = |X_M| \) (equality holds for the ideal condition where \( |X_M' \cap X_M'| = \emptyset \)).

It follows that:

\[
J(X_i, X_j) = \frac{|X_i \cap X_j|}{|X_i \cup X_j|} = \frac{|X_i' \cap X_j'|}{|X_i' \cup X_j'|} = \frac{|X_M' \cap X_M|}{|X_M' \cup X_M|} + 2\delta_G \\
\geq \frac{|X_M \cap X_M'|}{|X_M \cup X_M'|} + 2\delta_G \\
\geq \frac{|X_M \cap X_M'|}{|X_M \cup X_M'|} + \frac{2 \delta_G}{1 + \delta_G} \\
\geq 1 - \frac{\delta_G}{1 + \delta_G} \frac{|X_M \cap X_M'|}{|X_M \cup X_M'|} \\
\geq 1 - \frac{\delta_G}{1 + \delta_G} \frac{|X_M' \cap X_M|}{|X_M' \cup X_M|} = 1 - \frac{\delta_G}{1 + \delta_G} J(X_M', X_M).
\]

Since \( J(X_M', X_M') = e/(2t_M - e) \) and the expected value of \( e/t_M \) is \( e^{-k|M|} \), it clearly follows that the minimum required expectation of Jaccard similarity \( \tau \) is at least:

\[
\tau = E[J(X_i, X_j)] \geq 1 - \frac{\delta_G}{1 + \delta_G} \frac{1}{\frac{2k|M| - 1}{1}}
\]

To find the seed SDs, we follow a similar strategy as described in (Jain et al., 2017a), where our \( G_i \) and \( G_j \) corresponds to the long reads and the genomic hierarchy. We start by indexing a genome \( G \) and constructing an index \( G_i \) of genome \( G \) that is a sorted list of unique pairs \((i, x)\) where \( x \) is a \( k \)-mer in the winnow \( W(X_i) \) and \( i \) is a starting position of \( x \) in \( G \). We also construct a reverse index \( G_i^{-1} \): it provides for any input \( k \)-mer \( x \) a list of all positions \( i \) in \( G \) such that \((i, x) \in G_i \). These two tables are computationally inexpensive to calculate and allow us to quickly calculate winnow \( W(X_i) \) of any substring \( G_i \) in \( G \). For any locus \( i \) within \( G \), we enumerate a list of all pairs \( C = \{(j, x) | j \in G_i : x \in W(G_j)\} \). By using the winnowed MinHash lemma, we know that substring \( G_j \) starting at some locus \( j \) is a potential SD match for \( G_i \) if \( W(G_j) \) and \( W(G_j) \) share at least \( \tau \cdot k\)-mers, where the sketch size \( s \) is set to \( |W(G_j)| \). Since \( C \) is sorted by index, we can use this lemma to efficiently select all candidate locations \( j \in [j_a, j_b) \) for which \( J(G_i, G_j) \geq \tau \) by “rolling” a MinHash calculation as follows (Jain et al., 2017a).

We start by setting \( j_a \) and then construct an ordered set
\[
L = \{(y, b) : y \in W(X_i) \cup W(X_j) \}
\]
and \( b = 1 \) if \( y \in W(X_i) \cap W(X_j) \), where each element \( y \) is assigned 1 if it belongs to the intersection of \( W(X_i) \cap W(X_j) \) and zero otherwise.\(^5\) Then we keep “rolling” \( G_j \) by increasing \( j \) corresponds to checking the similarity between \( G_{i+n+1} \) and \( G_{j+n+1} \), where \( n \) is a number that minimizes \( L \) which occurred at position \( j \) and add any minimizer that occurs at the position \( j+n+1 \). Note that any such step costs at most \( O(\log(s)) \) operations (where \( s \) is the sketch size). With the appropriate auxiliary structures, we can calculate the winnowed MinHash estimate of \( W(X_i) \) and \( W(X_j)' \) in \( O(1) \) time. Once we find a \( j \) for which the corresponding MinHash estimate is maximal and above \( \tau \), we add the pair \((i, j)\) to the list of found SD seeds.

3.2 Finding potential SD regions

So far, we have assumed that the value of \( n \) is fixed and that \( n = |G_i| - |G_j| \). Now we lift this restriction and attempt to extend any seed SD as much as possible in both directions in order to ensure that we can find the boundaries of “true” SDs. This can be done by iteratively increasing the values of \( n \) and \( m \) by one (each step takes \( O(\log(s)) \) time), which essentially keeps expanding the sets \( W(X_i) \) and \( W(X_j) \): any minimizer which occurs at loci \( i + n + 1 \) and \( j + m + 1 \) within \( G \) is added to the ordered set \( L \). Here we utilize the same structures as in the previous step (see Section 3.1), and keep extending SD region until the value of the winnowed MinHash estimate goes below \( \tau \). We also terminate on both \( n \) and \( m \) extending SDs (we limit SEDEF to find potential SDs of at most 1 Mbp in length, as per WGAC). Note that the term \(|S(W(X_i) \cup W(X_j))|\) grows while \(|S(W(X_i) \cap W(X_j)|\) stays the same if two regions stop being similar after some time, which iteratively lowers the Jaccard estimate. We also interrupt the extension if the strings \( G_i \) and \( G_j \) begin to overlap. Note that we can perform this extension in the reverse fashion, by slowly decreasing the values \( i \) and \( j \) and applying the same techniques as described above. Finally, we report the largest \( G_i \) and \( G_j \) whose corresponding MinHash estimates are above \( \tau \).

For each potential SD, we also apply a gap-gram filter (Jokinen and Ukkonen, 1991) in order to further reduce the rate of false positives as follows. Define the \( q \)-gram similarity \( Q(G_i, G_j) \) of strings \( G_i \) and \( G_j \) to be the total number of \( q \)-mers shared by both \( G_i \) and \( G_j \). We adapt the well-known \( q \)-gram lemma for our problem as follows: any \( G_i \) and \( G_j \) whose edit error is below \( \delta \) and satisfies the SD error model will share at least \( n(1 - \delta_G - q\delta_M) - (n\tau + 1) \cdot (q - 1) \)-grams, where we assume that \( n \leq m \) and where \( \tau \) is the expected number of gaps per basepair in the genome. This modification allows us to losslessly reject any pair of substrings \( G_i \) and \( G_j \) that do not satisfy the SD error model for the given value of \( p_G \).

The aforementioned algorithm performed on the whole human genome produces more than 500 million potential SD regions due to the presence of various small repeats in the genome. In order to alleviate this problem, we only use \( k \)-mers that contain at least one non-repeat-masked nucleotide during the detection of seed SDs. In order to allow the case of repeats being inserted in the SD during the evolutionary process, the SD extension step uses any available \( k \)-mer to extend seed SDs. Finally, we pad each potential SD region with a pre-defined number of bases (which is a function of the size of the potential SD region) in order to further increase the probability of locating large SDs within the potential regions.

3.3 Detecting final SDs

After finding the potential SD regions, we enumerate all local alignments of size 1,000 within those regions that satisfy the SD criteria. In order to do this efficiently, SEDEF employs a two-tiered local chaining algorithm similar to those in (Abouelhoda and Ohlebusch, 2003; Myers and Miller, 1995). In the first part, we use a seed-and-extend method to construct the list of matching seed locations (of size 11 and higher), and proceed by finding the longest chains formed by those seeds via an \( O(n \log n) \) sparse dynamic programming algorithm as described in (Abouelhoda and Ohlebusch, 2003; Myers and Miller, 1995). In this step, we restrict the maximum gap size between the seeds to \( 1 - \delta_G \) in order to cluster the seeds within a chain as “close” as possible. After finding these initial chains (which might span less than 1,000bp), we refine them by further

\[^5\] Such a set can be efficiently implemented with a balanced binary tree where any update operation costs only \( O(\log |L|) \).
chaining them into the large final chains by allowing larger gaps. In order to retain compatibility with WGAC, which allows arbitrary large gaps within the SD (since it does not penalize the gap extension), we use the affine gap penalty during the construction of SD chains; however, we limit gaps to no longer than 10,000bp in order to avoid low-quality alignments. Chaining is accompanied by the global alignments which are done with the KSW2 library, which utilizes “single instruction, multiple data” (SIMD) parallelization through SSE instructions to speed up the global sequence alignment (Li, 2017a). Importantly, we report all our alignments in standard BEDPE format, together with corresponding edit strings in CIGAR format (Li et al., 2009) and various other useful metrics similar to WGAC such as Kimura two parameter genetic distance (Kimura and Ohta, 1972) and Jukes-Cantor distance (Jukes and Cantor, 1969).

In our experiments, we used $k = 12$ for the seed SD stage and $k = 11$ for chaining step (note that this parameter is configurable by user). While lower values of $k$ may improve the sensitivity, we found that any such improvement is rather negligible and not worth the increase in the running time. On the other hand, higher values of $k$ improve the running time while lowering the sensitivity.

|        | Total         | Seeding and Extending | Chaining and Aligning |
|--------|---------------|-----------------------|-----------------------|
|        | Human (hg19)  | Mouse (mm8)           |                       |
| 1 core | 10h 30m       | 7h 33m                | 2h 57m                |
| 0h 14m | 0h 10m        | 0h 04m                |                       |
| 80 cores | 13h 07m       | 7h 53m                | 5h 14m                |
| 0h 30m | 0h 10m        | 0h 20m                |                       |

4 RESULTS

We evaluated SEDEF using the human reference genome (UCSC hg19) and mouse reference genome (UCSC mm8), and compared its calls to WGAC calls 7. WGAC calls are the current gold (and only) standard of SDs in both human and mouse genomes, and are used as segmental duplication annotations by UCSC Genome Browser.

In case of human genome, the entire process took around 10 CPU hours with the peak RAM usage of 7 GB in single-CPU mode. SEDEF is also highly parallelizable, and it took only 14 minutes for the whole process to finalize on 80 CPU cores. This is a significant improvement over WGAC, which takes several weeks to complete (private communication). Similar running times were observed in mouse genome, despite the fact that mouse genome contains significantly more repeats than human genome and thus necessitates longer running times She et al. (2008). Run times on a single CPU and 80 CPU cores (when ran in parallel via GNU Parallel (Tange, 2011)) are given in Table 4.

SEDEF initially detected around 2,250,000 seed SD regions in human genome. After the chaining process, the final number of SDs was reduced to $\approx 186,400$. Finally, after filtering out the common repeats and other spurious hits, we report 67,882 final SD pairs that cover 219 Mbp of the human genome. This is a significant increase over WGAC data, which reports 24,477 SD pairs that cover 159 Mbp of the genome, as compared to 140 Mbp covered by 117,213 WGAC SDs. Of the additional 120 Mbp found by SEDEF, 45 Mbp belongs to non common repeat regions.

4.1 Filter and alignment accuracy

4.1.1 Simulations We also evaluated the accuracy of the seeding and chaining process based on total error rate $\delta$. For this purpose, we generated 1,000 random sequences of sizes 1–100Kbp for each $\delta \in \{0.01, 0.02, \ldots, 0.30\}$ (i.e. up to 30%), and for each such sequence generated a random segmental duplication according to the SD criteria defined above (where $\delta_{SD}$ and $\delta_{SE}$ are randomly chosen such that they are both less than $\min\{0.15, \delta\}$). All sequences and mutations were randomly generated with uniform distribution. These two sequences (original one and the randomly mutated one) were fed to SEDEF, and then we checked whether SEDEF finds a match between these two sequences, and whether this match covers the original SDs (a match covers SD if more than 95% of the SD bases are included in the match). As can be seen in Figure 3, SEDEF’s overall sensitivity is 99.94%, and the sensitivity drops slowly as $\delta$ increases. However, even for $\delta = 0.30$, sensitivity remains above 99%

We performed a similar experiment on chromosome 1, where we randomly fetched 10,000 sequences (uniform distribution) of various lengths and introduced random mutations to simulate a SD event. In this experiment, SEDEF had only a 0.15% false negative rate (i.e. undetected SDs), where all missed duplications were very small SDs of lengths $\approx 1,000$.

4.1.2 WGAC coverage It is worth mentioning that SEDEF-detected human SDs completely cover $\approx 98\%$ of the previously reported SD intervals ($\approx 99.6\%$ in basemaps) by WGAC. SEDEF entirely misses less than 0.3% ($70$) of SD intervals reported in WGAC results, and for the 1.4% of WGAC SDs, SEDEF reports partial overlap (i.e. less than 80% reciprocal overlap). All together, SEDEF misses about 0.6 Mbp out of 159 Mbp as reported by WGAC ($\approx 0.4\%$), where 0.5 Mbp contained short common repeats. We note that several WGAC SDs are in fact common repeats, and that several WGAC alignments contain long gaps. This is likely due to the dependency of WGAC on common repeat annotations, which may not be comprehensive. Additionally, WGAC employs several heuristics to reinsert common repeats to fuguized putative duplications that might “glue” very short non common repeat segments into larger segments with high repeat content that show similar alignment properties to a SD. This effect is much more present in mouse genome, where SEDEF misses 16,471 WGAC SDs (14.1%), and partially covers 2.2% of such SDs. However, in terms of basemaps SEDEF only misses 1.5Mbp (0.1 Mbp non

7 Note that as mentioned in the Introduction it is not possible to run WGAC without Sun Solaris operating system, therefore we were not able to benchmark it ourselves. WGAC calls were obtained from http://humanparalogy.gs.washington.edu and http://mouseparalogy.gs.washington.edu
common repeat elements). After extra validation, we found that most of missed WGAC SDs ($\approx 14,470$) are in fact common repeats incorrectly reported as SDs; thus SEDEF misses only 1.7% of the correct WGAC SD calls.

### 4.2 Comparison to other methods

We also evaluated the SD discovery accuracy of whole-genome aligners Minimap2 (Li, 2017b) and MUMmer/nucmer (Marçais et al., 2018) on the human genome assembly (UCSC hg19). These tools do not support SD detection out of the box; however, a self assembly-to-assembly comparison can be performed in order to identify the repetitive regions in the genome. These regions can be refined into SDs after applying further processing with SDDetector (Dallery et al., 2017) and filtering out candidate SDs which consist solely of common short repeats. Compared to these tools, SEDEF is an integrated pipeline for identifying SDs from scratch in a given assembly. Note that other similar tools, such as DupMasker (Jiang et al., 2008), are developed to annotate segmental duplications and require already existing SD database from similar genomes to be able to mark SDs in a given genome.

We ran these tools on 20 CPU cores using the GNU Parallel (Tange, 2011) by aligning all pairs of chromosomes in hg19. Minimap2 identified only 29% of the SD intervals reported by WGAC, which spanned 33% of the duplicated basepairs (53 Mbp out of 159 Mbp). MUMmer/nucmer approached better SD coverage performance, which identified 98.8% of WGAC regions that spanned 89% of duplicated basepairs (143 Mbp out of 159 Mbp), but the analysis was much slower and completed in 20 hours in the same compute setting. Minimap2 required 1.5 hours of run times using 20 CPU cores (in comparison, SEDEF takes only 36 minutes on 20 cores).

Overall, Minimap2 analysis misses a significant amount of duplications in a self-comparison task when using the recommended parameters of intra-species assembly-to-assembly comparison. Meanwhile MUMmer/nucmer-based approach covers the SD regions more consistently with those reported by WGAC; however it still misses many WGAC calls which are found by SEDEF. Finally, SEDEF is able to find more calls compared to the other tools in much shorter amount of time, as shown in Table 4.2.

#### Table 2. SD coverage of the human genome (hg19) as reported by different tools. Misses and Extra are calculated with respect to the WGAC SD calls, which are currently the gold standard of SD calls. Note that we did have filtered out all calls where at least one mate is composed solely of common short repeats (Minimap2, MUMmer/nucmer and SDDetector) as we did on SEDEF. All running times were adjusted for 20 CPU cores (all tools which support parallelization were run on 20 cores).

| Tool                | Covers | Misses | Extra | Time (h:m) |
|---------------------|--------|--------|-------|------------|
| WGAC (gold standard)| 159.5  | 0.0    | 0.0   | weeks      |
| SEDEF               | 218.8  | 0.6    | 60.0  | 0:36       |
| Minimap2            | 53.3   | 107.3  | 1.1   | 1:30       |
| MUMmer/nucmer       | 142.6  | 30.8   | 13.9  | $\geq 20:00$ |
| SDDetector          | 30.1   | 130.8  | 1.5   | $\geq 1:00$* |

* Adjusted running time for 20 cores; in reality, SDDetector spends $\geq 8$ hours in the single threaded pre-processing stage. Furthermore, the reported running time only includes post-processing and does not include initial BLAST alignment calculations.

### 5 Conclusion

Segmental duplications are among the most important forms of genomic rearrangements that drive genome evolution. However, their accurate identification is lacking due to the unavailability of necessary computational tools. In this manuscript we presented SEDEF to help fill this gap in methodology.

In future work, we aim to characterize the effect of various edit distance embeddings and techniques such as gapped q-grams (Burkhardt and Kärkkäinen, 2002; Bar-Yossef et al., 2004). While many of these techniques have been previously implemented (Hanada et al., 2017), our initial experiments did not show that any such embeddings or techniques are beneficial for strings with large edit distances.
SEDEF is designed as a fast, accurate, and user friendly tool to discover duplicated segments in genome assemblies. Therefore it aims to help researchers easily identify duplicated segments in genomes from several organisms, enabling them to extend their ability to perform comparative genomic studies in complex regions of the genome. We aim to extend it with an A-Bruijn graph based analysis (Jiang et al., 2007) to provide a full view of the evolution of segmental duplications. Armed with the extensions as we mention above, we will then use SEDEF to fully analyze reference genome assemblies from various genomes to both evaluate the assembly accuracy, and to better understand the role of segmental duplications in organism evolution.

ACKNOWLEDGEMENTS:
We thank Evan E. Eichler for early discussions on formulating the problem, and Ashwin Narayan for helpful suggestions. Funding: This work is supported in part by NSF ERC Discovery Grant to F.H., EMBO Installation Grant (IG-2521) to C.A. and NIH grant GM108348 to B.B.

Conflict of interest: None declared.

REFERENCES
Abouelhoda, M. I. and Ohlebusch, E. (2003). A local chaining algorithm and its applications in comparative genomics. In G. Benson and R. D. M. Page, editors, Algorithms in Bioinformatics, pages 1–16, Berlin, Heidelberg, Springer Berlin Heidelberg.

Alkan, C., et al. (2009). Personalized copy number and segmental duplication maps using next-generation sequencing. *Nat Genet*, 41(10), 1061–1067.

Alkan, C., et al. (2011a). Genome structural variation discovery and genotyping. *Nat Rev Genet*, 12(5), 363–376.

Alkan, C., et al. (2011b). Limitations of next-generation genome sequence assembly. *Nat Methods*, 8(1), 61–65.

Altschul, S. F., et al. (1990). Basic local alignment search tool. *J Mol Biol*, 215(3), 403–410.

Andoni, A., et al. (2010). Polylogarithmic approximation for edit distance and the asymmetric query complexity. In *Proc. IEEE 51st Annual Symp. Foundations of Computer Science*, pages 377–386.

Backurs, A. and Indyk, P. (2015). 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. ACM.

Bailey, J. A., et al. (2001). Segmental duplications: organization and impact within the current human genome project assembly. *Genome Res*, 11(6), 1005–1017.

Bailey, J. A., et al. (2002). Recent segmental duplications in the human genome. *Science*, 297(5583), 1003–1007.

Bar-Yossef, Z., et al. (2004). Approximating edit distance efficiently. In *Proc. 45th Annual IEEE Symp. Foundations of Computer Science*, pages 550–559.

Broder, A. Z. (1997). On the resemblance and containment of documents. In *Proc. Compression and Complexity of SEQUENCES 1997* (Cat. No.97TB100171), pages 21–29.

Burkhardt, S. and Kärkkäinen, J. (2002). One-gapped q-gram filters for Levenshtein distance. In *Annual Symposium on Combinatorial Pattern Matching*, pages 225–234. Springer.

Carruthers-Smith, K. (2013). Sliding window minimum implementations. https://people.cs.uct.ac.za/~ksmith/articles/sliding_window_minimum.html. Accessed: 2018-01-28.

Chaissen, M. J., et al. (2015). Genetic variation and the de novo assembly of human genomes. *Nat Rev Genet*.

Dallery, J.-F., et al. (2017). Gapless genome assembly of collotetrichum higginsianum reveals chromosome structure and association of transposable elements with secondary metabolite gene clusters. *BMC genomics*, 18(1), 667.

Fan, H., et al. (2015). An assembly and alignment-free method of phylogeny reconstruction from next-generation sequencing data. *BMC genomics*, 16, 522.

Firtina, C. and Alkan, C. (2016). On genomic repeats and reproducibility. *Bioinformatics*, 32(15), 2243–2247.

Gonzalez, E., et al. (2005). The influence of CCL3L1 gene-containing segmental duplications on HIV-1AIDS susceptibility. *Science*, 307(5714), 1434–1440.

Hanada, H., et al. (2017). On practical accuracy of edit distance approximation algorithms. arXiv preprint arXiv:1701.06134.

Harris, R. S. (2007). *The timetree of life*. OUP Oxford.

Hollox, E. J., et al. (2008). Psoriasis is associated with increased beta-defensin genomic copy number. *Nat Genet*, 40(1), 23–25.

Jain, C., et al. (2017a). A fast approximate algorithm for mapping long reads to large reference databases. In S. C. Sahinalp, editor, *Proceedings of 21st Annual International Conference on Research in Computational Molecular Biology (RECOMB 2017)*, volume 10292, pages 66–81, Cham. Springer International Publishing.

Jain, M., et al. (2017b). Nanopore sequencing and assembly of a human genome with ultra-long reads. *bioRxiv*.

Jiang, Z., et al. (2007). Ancestral reconstruction of segmental duplications reveals punctuated cores of human genome evolution. *Nature genetics*, 39, 1361–1368.

Jiang, Z., et al. (2008). Dupmasker: a tool for annotating primate segmental duplications. *Genome research*, 18(8), 1362–1368.

Jokinen, P. and Ukkonen, E. (1991). Two algorithms for approximate string matching in static texts, pages 240–248. Springer Berlin Heidelberg, Berlin, Heidelberg.

Jukes, T. H. and Cantor, C. R. (1969). Chapter 24—evolution of protein molecules. In H. MUNRO, editor, *Mammalian Protein Metabolism*, pages 21–132. Academic Press.

Kimura, M. and Ohta, T. (1972). On the stochastic model for estimation of mutational distance between homologous proteins. *Journal of Molecular Evolution*, 2(1), 87–90.

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

Li, H. (2017a). KSW2: global alignment and alignment extension. Li, H. (2017b). Minimap2: fast pairwise alignment for long dna sequences. arXiv preprint arXiv:1708.01992.

Li, H., et al. (2009). The sequence alignment/map format and samtools. *Bioinformatics*, 25(16), 2078–2079.

Li, R., et al. (2010). The sequence and de novo assembly of the giant panda genome. *Nature*, 463, 311–317.

Marçais, G., et al. (2018). Mummer4: A fast and versatile genome alignment system. *PLoS computational biology*, 14(1), e1005944.

Marques-Bonet, T., et al. (2009). A burst of segmental duplications in the genome of the African great ape ancestor. *Nature*, 457(7231), 877–881.

Mills, R. E., et al. (2011). Mapping copy number variation by population-scale genome sequencing. *Nature*, 470(7332), 59–65.

Montgomery, S. B., et al. (2013). The origin, evolution, and functional impact of short insertion-deletion variants identified in 179 human genomes. *Genome research*, 23, 749–761.

Mostovoy, Y., et al. (2016). A hybrid approach for de novo human genome sequence assembly and phasing. *Nat Methods*. 
Myers, G. and Miller, W. (1995). Chaining multiple-alignment fragments in sub-quadratic time. In Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA ’95, pages 38–47, Philadelphia, PA, USA. Society for Industrial and Applied Mathematics.

Prado-Martinez, J., et al. (2013). Great ape genetic diversity and population history. Nature, 499(7459), 471–475.

Pu, L., et al. (2018). Detection and analysis of ancient segmental duplications in mammalian genomes. Genome Research.

Scally, A. (2016). The mutation rate in human evolution and demographic inference. Current opinion in genetics & development, 41, 36–43.

Schleimer, S., et al. (2003). Winnowing: local algorithms for document fingerprinting. In Proceedings of the 2003 ACM SIGMOD international conference on Management of data, pages 76–85. ACM.

Schwartz, S., et al. (2003). Human-mouse alignments with BLASTZ. Genome Research, 13, 103–107.

She, X., et al. (2008). Mouse segmental duplication and copy number variation. Nature genetics, 40(7), 909.

Steinberg, K. M., et al. (2017). Building and improving reference genome assemblies. Proceedings of the IEEE, 105(3), 422–435.

Sudmant, P. H., et al. (2010). Diversity of human copy number variation and multicopy genes. Science, 330(6004), 641–646.

Sudmant, P. H., et al. (2013). Evolution and diversity of copy number variation in the great ape lineage. Genome Res, 23(9), 1373–1382.

Tange, O. (2011). Gnu parallel - the command-line power tool. ;login: The USENIX Magazine, 36(1), 42–47.

Treangen, T. J. and Salzberg, S. L. (2012). Repetitive DNA and next-generation sequencing: computational challenges and solutions. Nat Rev Genet, 13(1), 36–46.

Yang, Y., et al. (2007). Gene copy-number variation and associated polymorphisms of complement component C4 in human systemic lupus erythematosus (SLE): low copy number is a risk factor for and high copy number is a protective factor against SLE susceptibility in European Americans. Am J Hum Genet, 80(6), 1037–1054.

Yeo, S., et al. (2017). ARCS: Scaffolding genome drafts with linked reads. Bioinformatics.