Supplemental Material for

Characterization of Structural Variants with Single Molecule and Hybrid Sequencing Approaches

Anna Ritz*[1], Ali Bashir[1,3], Suzanne Sindi[4], David Hsu[5], Iman Hajirasouliha[1] and Benjamin Raphael†[1,6]

[1]Department of Computer Science, Brown University, RI
[2]Department of Genetics and Genomic Sciences, Mount Sinai School of Medicine, NY
[3]Institute for Genomics and Multiscale Biology, Mount Sinai School of Medicine, NY
[4]School of Natural Sciences, University of California, Merced, CA
[5]Pacific Biosciences, Menlo Park, CA
[6]Center for Computational Molecular Biology, Brown University, RI

Contents

*annaritz@vt.edu (Current Affiliation: Department of Computer Science, Virginia Tech, VA)
†braphael@cs.brown.edu

1 Supplemental Methods

1.1 Multi-Reads

A τ-multi-read is a fragment of DNA that is sequenced in τ portions. Single reads, including long reads, are 1-multi-reads because they are sequenced in one contiguous portion. Paired reads are 2-multi-reads because only the ends of the DNA fragment is sequenced. Multi-linked reads, including strobe reads, have $\tau > 1$ distinct segments of the fragment sequenced. Figures 1 and 2 highlight the distinction between the number of reads sequenced from the technology and the number of segments that are aligned to a reference genome.

Multi-breakpoint-mappings, on the other hand, describe a τ-multi-read as it is aligned to a reference genome. A 1-multi-read, for example, may align to four distinct regions of the reference genome, producing a t-multi-breakpoint-mapping. τ-multi-reads where $\tau > 1$ contain segments of "dark" nucleotides that are not sequenced by the technology; however, they may still align to $t \geq \tau$ portions of the reference genome. Fragments with $\tau = 1$ (such as long reads) may still produce t-multi-breakpoint-mappings if they span a structural variant (Figure 2).

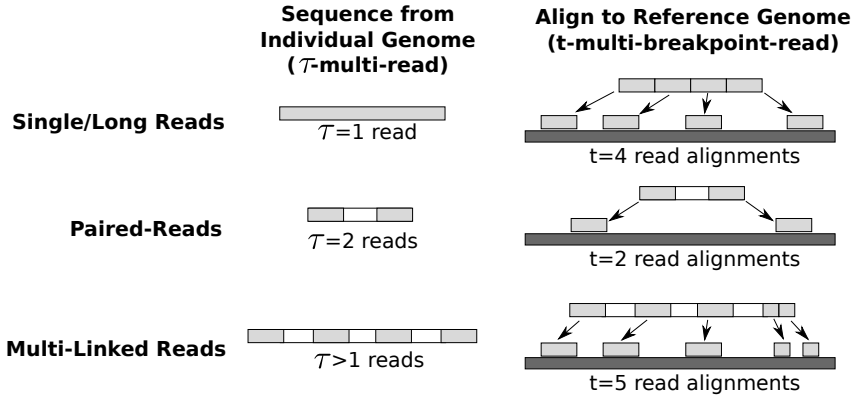

Figure 1: Multi-Reads and Multi-Breakpoint-Mappings.

1.2 A Model For Structural Variation

We model an individual genome that is generated by adding, duplicating, and rearranging segments of a reference genome as well as adding novel sequences. Each pair of adjacent coordinates in the individual genome that are not adjacent in the reference genome is called an *novel adjacency*. We assume that these novel adjacencies are independent. In Figure 1, the t-multi-breakpoint-mapping example for the single read contains 3 novel adjacencies.

We assume that DNA fragments are selected from the individual genome uniformly, and the leftmost coordinates of the fragments follow a Poisson process with parameter λ equal to the expected number of DNA fragments starting at each base in the individual genome. We first describe generating τ-multi-reads S for $\tau > 1$. They are generated by the following process:

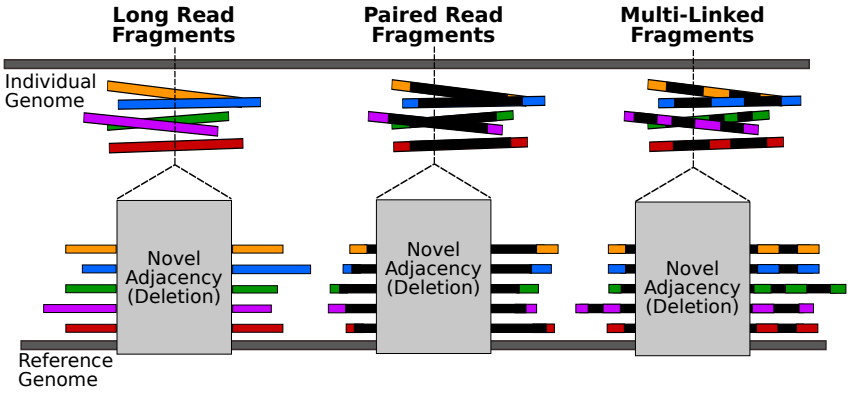

Figure 2: τ-**Multi-Reads from Different Sequencing Platforms.** Each novel adjacency is supported by five fragments, where the colored portions are sequenced and the black portions are not sequenced. Long read fragments have $\tau = 1$, paired-read fragments have $\tau = 2$, and multi-linked fragments may have $\tau \geq 2$. All three fragment types in this example result in t-multi-breakpoint-mappings with $t > 1$.

1. Select a starting position x_1 sampled uniformly from a diploid individual genome. Select either the forward or reverse strand with equal probability.
2. Sequence $R_1^{(S)} \sim N(\mu_1, \sigma_1)$ bases according to the Pacific Biosciences error model.
3. For $i = 2, \ldots, t$

   Advance $b \sim N(\mu_2, \sigma_2)$ bases.

   Sequence $R_i^{(S)} \sim N(\mu_1, \sigma_1)$ bases according to the Pacific Biosciences error model.

Thus, S is comprised of a set of τ reads $\{R_1^{(S)}, R_2^{(S)}, \ldots, R_\tau^{(S)}\}$ (Figure 3). From a set $\mathbf{S} = \{S_1, \ldots, S_n\}$ of multi-reads and the individual genome, we wish to infer the set of novel adjacencies in the individual genome.

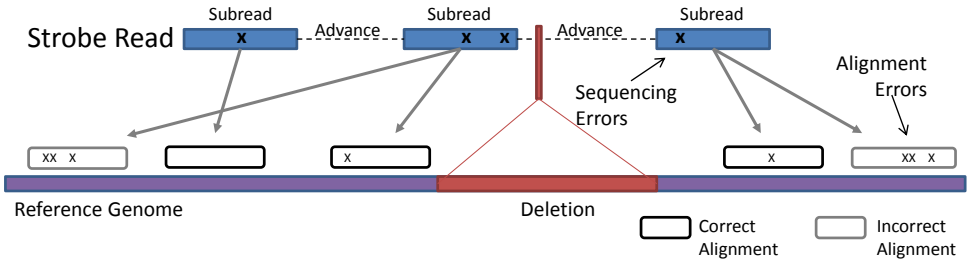

Figure 3: Strobe Sequencing by Pacific Biosciences. In this example, a multi-read consists of three reads from the same fragment of DNA separated by two advances (top). The reads are then independently aligned to a reference genome using BLASR [2]. There are several sources of error in this pipeline, including sequencing errors, alignment errors, and incorrect alignments to the reference due to repetitive regions. We assume that there is at most one correct subread alignment for each subread.

1.2.1 Concordant and Discordant Consecutive Read Pairs

To determine novel adjacencies, it is convenient to consider the set $A(R_i^{(S)})$ of full alignments to the reference genome.[1] Each read alignment $a \in A(R_i^{(S)})$ provides (i) the interval $[x_a, y_a]$ in

[1]For simplicity, we describe this step when $\tau > 1$ and $\tau = t$.

the reference genome corresponding to the alignment location, (ii) the orientation $sign_a$ of the alignment, (iii) and the edit distance ϵ_a of the alignment. Let $\mathcal{P}$ be the set of pairs of alignments from adjacent reads in each multi-read, or the *consecutive read pairs*:

$$\mathcal{P} = \bigcup_{S \in \mathbf{S}} \bigcup_{i=1}^{t-1} \left\{ A(R_i^{(S)}) \times A(R_{i+1}^{(S)}) \right\}$$

We infer the presence of novel adjacencies by considering $\mathcal{P}$, the set of pairs of alignments from adjacent reads.

A pair of consecutive read alignments $(a_1, a_2) \in \mathcal{P}$ is *concordant* if the aligned distance and orientation of the pair is expected given the generative model. That is, if

$$sign_{a_1} = sign_{a_2} \text{ and } \begin{cases} L_{\min} \leq x_{a_2} - y_{a_1} \leq L_{\max} & \text{if } sign_{a_1} \text{ is positive} \\ L_{\min} \leq x_{a_1} - y_{a_2} \leq L_{\max} & \text{if } sign_{a_1} \text{ is negative} \end{cases}$$

for bounds L_{min} and L_{max} on the advance length (Figure 4 Left). Typically $L_{\min}$ and $L_{\max}$ are chosen to be three standard deviations away from the mean. If (a_1, a_2) is not concordant according to the definition above, then it is *discordant*, and is taken to imply a novel adjacency in the individual genome. We denote the set of discordant alignments $P_{\text{discord}} \subset \mathcal{P}$. The type of novel adjacency (deletion, insertion, inversion, translocation) depends on how the conditions above are violated. We focus on deletions and insertions. For deletions (Figure 4 Middle), there exists coordinates (x, y) in the reference genome such that

$$sign_{a_1} = sign_{a_2} \text{ and } \begin{cases} L_{\min} \leq (x - y_{a_1}) + (x_{a_2} - y) \leq L_{\max} & \text{if } sign_{a_1} \text{ is positive} \\ L_{\min} \leq (x - x_{a_2}) + (y_{a_1} - y) \leq L_{\max} & \text{if } sign_{a_1} \text{ is negative} \end{cases}.$$

For inversions (Figure 4 Right), there exists coordinates (x, y) in the reference genome such that

$$sign_{a_1} \neq sign_{a_2} \text{ and } \begin{cases} L_{\min} \leq (x - y_{a_1}) + (y - x_{a_2}) \leq L_{\max} & \text{if } sign_{a_1} \text{ is positive} \\ L_{\min} \leq (y_{a_1} - x) + (x_{a_2} - y) \leq L_{\max} & \text{if } sign_{a_1} \text{ is negative} \end{cases}.$$

The example in Figure 4 Right shows an inversion where the condition above is satisfied for both pairs $\left(a_1 \in A(R_1^{(S)}), a_2 \in A(R_2^{(S)})\right)$ and $\left(a_1 \in A(R_2^{(S)}), a_2 \in A(R_3^{(S)})\right)$. Other inequalities hold for other types of novel adjacencies.

1.2.2 Multi-Breakpoint-Mappings

Each multi-read $S \in \mathbf{S}$ has a set of *multi-breakpoint-mapping* $A(S)$, defined from the read alignments $A(R_t^{(S)})$ as

$$A(S) = \left\{ A(R_1^{(S)}) \times A(R_2^{(S)}) \times \ldots \times A(R_t^{(S)}) \right\} \cup \emptyset. \tag{1}$$

The true multi-breakpoint-mapping alignment might be missing from the set of read alignments (for example, if a read falls within an insertion on the individual genome with novel sequence). Thus, we include the empty set $\emptyset$ as an element of $A(S)$ to account for missing alignments. A selection of one multi-breakpoint-mapping alignment from each multi-read is called a *mapping* M:

$$M \in \left(a_s \in A(S) \; \forall \; S \in \mathbf{S}\right), \tag{2}$$

where some elements in M may be duplicates (i.e. in the case where multiple multi-reads have $\emptyset$ selected).

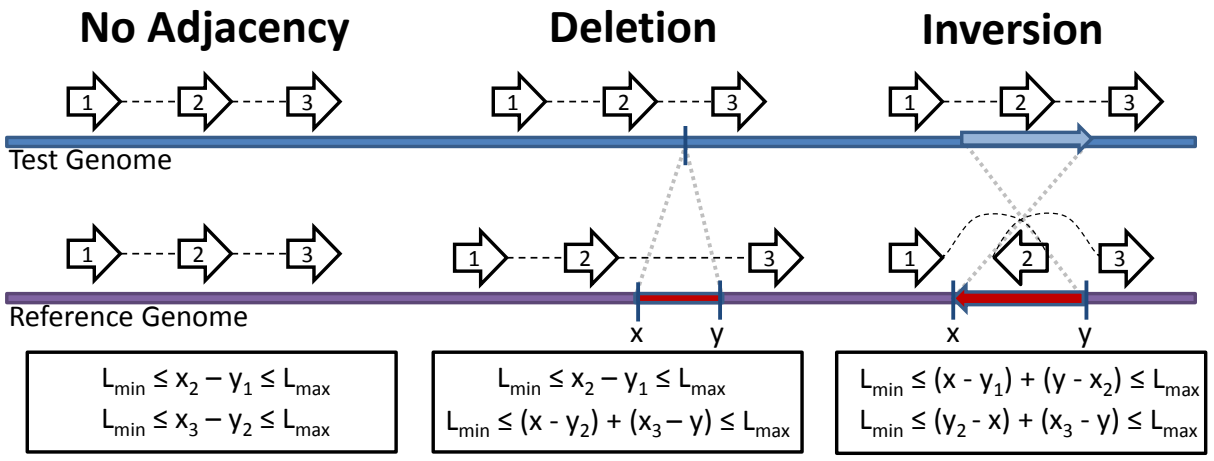

Figure 4: Concordant and Discordant Pairs from Consecutive Read Alignments. (Left) If there is no novel adjacency in the individual genome, the mapping for the multi-breakpoint read contains all concordant pairs. (Middle) A deletion between two of the reads results in one discordant and one concordant pair. (Right) An inversion that reverses the middle read results in two discordant pairs.

Removing Concordant Multi-Breakpoint Read Alignments A concordant multi-breakpoint-mapping for S consists of only concordant consecutive read pairs. Multi-reads that come from non-rearranged parts of the genome and are aligned correctly will contain such a multi-breakpoint-mapping; further, most of the individual genome will not be rearranged. Thus, if such a concordant multi-read alignment exists, then we assume that is the true mapping and we remove such multi-reads from consideration, since they do not indicate any adjacency. This drastically reduces the number of multi-reads that must be considered.

1.3 Cluster Diagrams

A geometric interpretation of pairs was introduced for paired reads by [9], and is extended here for multi-breakpoint-mappings. The geometric interpretation of a discordant pair $d \in P_{\text{discord}}$ is a trapezoid in $\mathbb{R}^2$, where the interior of the trapezoid contains the set of points (x, y) that could correspond to the novel adjacency. If d implies a deletion, for example, (x, y) is defined by the inequality in Equation (1). If two discordant consecutive read pairs $d_1, d_2 \in P_{\text{discord}}$ intersect in $\mathbb{R}^2$, the intersection contains possible novel adjacencies (x, y) that are consistent with both pairs.

For the set of pairs P_{discord} (which are now represented as trapezoids in $\mathbb{R}^2$), we want to concisely describe all trapezoid intersections. This can be done by constructing the Hasse diagram for the partially ordered set $(P_{\text{discord}}, \subset)$, which can be constructed in the following steps:

1. Construct the directed graph $G_h = (V_h, E_h)$ of $(P_{\text{discord}}, \subset)$. The nodes V_h are the finite collection of all subsets of P_{discord} with nonempty trapezoid intersections. There is a directed edge (u, v) if $u \subset v$.

2. Remove all edges implied by the transitive property. That is, if $u \subset v$ and $v \subset z$, remove the edge (u, z).

The Hasse diagram contains redundant information, and can be very large in practice. We apply the following node-contraction which greatly reduces the storage space of the graphs:

1. Contract nodes with a single outgoing edge.

2. Remove transitive edges after node contraction.

3. Repeat steps 1 and 2 until all nodes have more than one outgoing edge.

We call the resulting graph $G = (V \subseteq V_h, E \subseteq E_h)$ a *cluster diagram*, which can be efficiently computed using a line-sweep algorithm similar to [9] with additional book-keeping.

To provide some intuition about G, consider a set $P'_{\text{discord}} \subset P_{\text{discord}}$ of trapezoids that share a common intersection in $\mathbb{R}^2$ (Figure 5). That is, there are points (x, y) that are supported by all discordant pairs in P'_{discord}. G will then be a single node with the set P'_{discord}. However, suppose the trapezoids in P'_{discord} do not share a common area of intersection. Then, we are restricted to selecting a subset of pairs in P'_{discord} that are consistent with points (x, y) that indicate the adjacency. Each node in G is an informative subset of P'_{discord}; that is, it either is a set of a maximal number of pairs in P'_{discord} that are consistent with a single adjacency or it is a set of pairs in P'_{discord} that are consistent with multiple adjacencies.

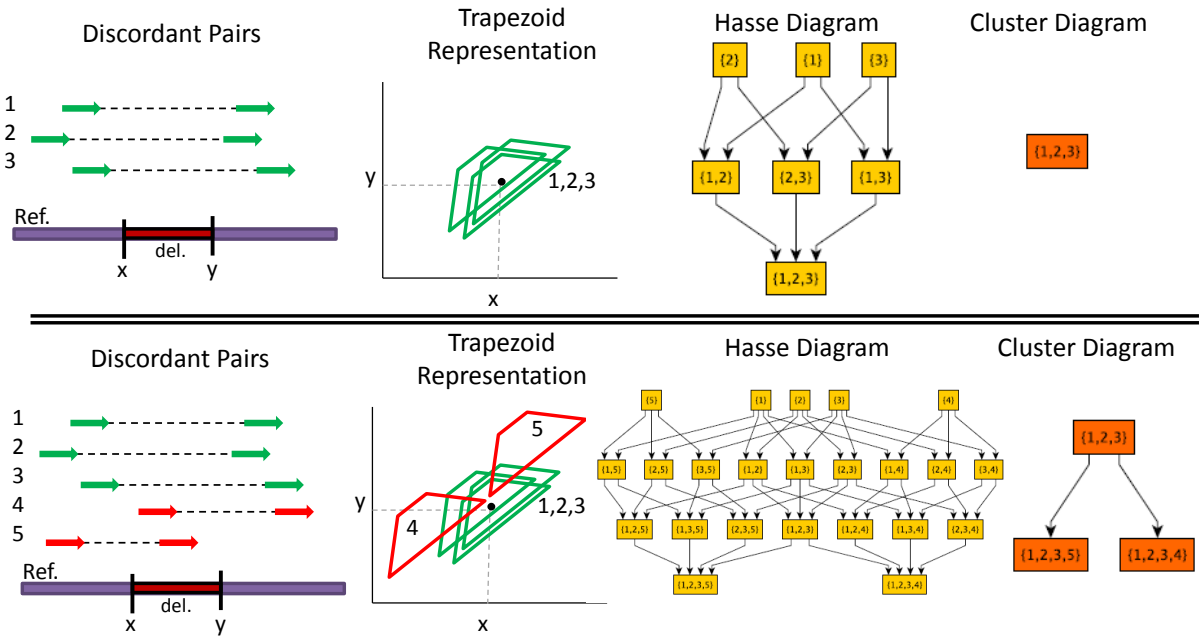

Figure 5: Cluster Diagrams for (Top) discordant pairs with a common area of intersection and (Bottom) discordant pairs with no common area of intersection. Each pair $d \in P_{\text{discord}}$ can be represented as a trapezoid in $\mathbb{R}^2$, and the set of all transitively-reduced intersections is a Hasse diagram (a partially ordered set on the pairs with the subset operator). After node contraction that removes redundancy in the Hasse diagram, we get a cluster diagram. For sets of pairs with a common intersection, the cluster diagram is simply a single node.

Set cover algorithm for identifying implied novel adjacencies M Each node in G represents a region supporting a different adjacency, but many of the nodes contain the same discordant pairs in the sets. Let $P_{\text{discord}}(M)$ is the set of discordant pairs implied by M. Each pair in this set may imply exactly one potential adjacency, and thus may only contribute to one node in G. From above, each node $v \in V$ in the cluster diagram consists of a set of discordant pairs.

To find the set of adjacencies from M, we make a parsimony assumption (similar to the objectives in [8, 3]) that we select the smallest number of adjacencies that completely describe the pairs in $P_{\text{discord}}(M)$. We employ an extension of the greedy approximation of set cover to determine a set of pair sets U_M.

Algorithm 1 SetCover(M)

$U_M \Leftarrow \emptyset$
$D \Leftarrow P_{\text{discord}}(M)$ *// set of discordant pairs in M*
while $D \neq \emptyset$ **do**
  $v' \Leftarrow \arg\max_{v \in V} v \cap D$ *// v' contains the most pairs in the current set.*
  $u \Leftarrow v' \cap D$
  $U_M \Leftarrow U_M \cup u$ *// Add the set of pairs*
  $D \Leftarrow D \setminus u$ *// Remove pairs from consideration.*
end while
return U_M

1.4 Probabilistic Model

The probabilistic model incorporates two pieces of information from the alignments in M: the quality of the read alignments and the novel adjacencies created by alignments of consecutive pairs of read alignments in M. Our goal is to compute $P(M|D)$, the probability of a mapping M given the data. When we apply Bayes' Rule to $P(M|D)$, the probability $P(D|M)$ of the data given the mapping consists of two terms: the probability of the selected read alignments from $A(\mathbf{S})$ and the probability of the novel adjacencies, which are represented as a cluster diagram G.

$$P(M|D) = \frac{P(D|M)P(M)}{P(D)} = \frac{P(A(\mathbf{S})|M)P(G|M)P(M)}{P(D)}. \tag{3}$$

We first describe how to compute $P(A(\mathbf{S})|M)$.

$P(A(\mathbf{S})|M)$ **Computation.** A mapping M can be partitioned into the set $A(M)$ of read alignments in M and the number e_M of missing alignments (empty sets). Thus, the probability $P(A(\mathbf{S})|M)$ is the product of the probability of just the alignments in $P(A(M))$, and the probability of observing e_M missing alignments,

$$P(A(\mathbf{S})|M) = P(A(M), e_M) = P(A(M))P(e_M). \tag{4}$$

The probability $P(A(M))$ of the read alignments depends on the error rates and fragment lengths of the sequencing technology. We have a fixed probability p_{seq} of the sequencing error and a fixed probability p_{err} of completely missing an alignment. For a mapping M, let $\epsilon(M) = \sum_{a \in A(M)} \epsilon_a$ be the total edit distance and $\ell(M)$ be the total length of all read alignments in M. We use a binomial distribution to model the probability of observing $\epsilon(M)$ errors in a string of length $\ell(M)$ when the sequencing error is p_{seq}. We approximate the Binomial term with a Normal distribution when $\ell(M)p_{seq} > 6$ (which is often the case).

$P(G|M)$ **Computation.** We now move on to computing $P(G|M)$. Each mapping M determines a subset $P_{\text{discord}}(M) \subseteq P_{\text{discord}}$ of discordant pairs. We use the cluster diagram G to identify the set of novel adjacencies that M implies. Specifically, we find the smallest number of nodes in the cluster diagram G that cover the discordant pairs in $P_{\text{discord}}(M)$, which corresponds to the smallest number of implied novel adjacencies from $P_{\text{discord}}(M)$. We efficiently find such nodes using a set cover approximation on the leaves of G (Section 1.3). We partition the discordant pairs $P_{\text{discord}}(M)$ into the selected novel adjacencies; the number of pairs for each adjacency is called the adjacency's *support*. Let the non-zero supports be a vector $\Phi(M)$; then $P(G|M) = P(\Phi(M))$ is the probability of observing novel adjacencies with supports $\Phi(M)$. We assume that the novel adjacencies are independently distributed in the individual genome, and the multi-breakpoint-mappings are uniformly sampled from the individual genome. We model

the expected support of a novel adjacency as a Poisson process with parameter $\lambda_d = \lambda L_{\text{avg}}(t-1)$ where λ is the sequence coverage, t is the number of reads in the multi-breakpoint-mapping, and L_{avg} is the average advance length.

$P(M|D)$ in Equation (3) now becomes

$$\frac{\left[\text{Bin}\left(\epsilon(M); \ell(M), p_{\text{seq}}\right) p_{\text{err}}^{e_M}\right]\left[\prod_{k\in\Phi(M)} \text{Poiss}(k;\lambda_d)\right] P(M)}{P(D)} \tag{5}$$

with hyperparameters p_{seq}, p_{err}, and λ_d. The hyperparameters can be pre-specified or inferred from the read alignments. We take the prior $P(M)$ to be uniform over all mappings. Although described above for fragments from a single sequencing platform, we have generalized the model to include multiple sequencing technologies (e.g., strobes and pairs), allowing for multiple hyperparameters (see Section 1.7).

1.5 Markov Chain Monte Carlo (MCMC)

1.5.1 The MCMC Algorithm

We have designed a Metropolis-Hastings algorithm that draws samples of M with probability proportional to the probability of the mappings M given the input data. The data D consists of the set of all multi-breakpoint-mappings $A(\mathbf{S})$ and a set of candidate novel adjacencies in the form of a cluster diagram G. Algorithm 2 takes the data D and a number of iterations z, and returns a set of $z+1$ mapping vectors M that are sampled with probability approximating $P(M|D)$.

Algorithm 2 MCMC($D = \{A(\mathbf{S}), G\}$,z)

1: Initialize $M^{(0)}$ with a random assignment of mappings
2: **for** $i = 1 \rightarrow z$ **do**
3: $M' \Leftarrow \begin{cases} M^{(i-1)} & \text{with probability } \frac{1}{2} \\ \text{makeLocalMove}(A(\mathbf{S}), M^{(i-1)}) & \text{with probability } \frac{\beta}{2} \\ \text{makeJumpMove}(A(\mathbf{S}), M^{(i-1)}) & \text{with probability } \frac{1-\beta}{2} \end{cases}$
4: $ratio \Leftarrow \dfrac{P(M'|D)q(M^{(i-1)}|M')}{P(M^{(i-1)}|D)q(M'|M^{(i-1)})}$
5: $M^{(i)} \Leftarrow \begin{cases} M' & \text{with probability } \alpha(M', M^{(i-1)}) = \min(1, ratio) \\ M^{(i-1)} & \text{otherwise} \end{cases}$
6: **end for**
7: **return** $\{M^{(0)}, M^{(1)}, M^{(2)}, \ldots, M^{(z-1)}, M^{(z)}\}$

There are two ways this chain moves through the solution space: via local moves that change the assignment of a single multi-breakpoint read and via jump moves that change the assignment of multiple multi-breakpoint reads. β is a user-defined parameter that determines the proportion of local vs. jump moves: we set $\beta = 0.9$.

Let $M(S)$ be the mapping selected for S from $A(S)$. Algorithm 3 describes the local move, which takes the set of multi-breakpoint read alignments and a mapping vector and returns the vector with a single entry changed.

Algorithm 4 describes the jump move, which moves sets of multi-breakpoint-mappings. The jump move must be symmetric (if we can move from M to M', we must be able to move from M' back to M), and for mathematical convenience the jump move should not be the same as a local move. Thus, we perform this move on a subset of connected components $\tilde{G} = \{G_1, G_2, \ldots, G_k\}$ of G. Consider a connected component $G_k \subseteq G$ of the cluster diagram; there is a set of multi-breakpoint reads $S \subseteq \mathbf{S}$ that have discordant pairs present in G_k. G_k is in $\tilde{G}$ if there are at

Algorithm 3 makeLocalMove($A(\mathbf{S})$,M)

$M' \Leftarrow M$
Select multi-breakpoint read S uniformly from $\mathbf{S}$.
$m_{old} \Leftarrow M(S)$
while $M'(S) = m_{old}$ **do**
    $M'(S) \Leftarrow \begin{cases} \emptyset & \text{with probability } p_{err} \\ a_S & \text{with probability } \frac{1-p_{err}}{|A(S)|} \end{cases}$
end while
return M'

least two strobes $S_i, S_j \in S$ such that $|A(S_i)| = |A(S_j)| = 1$. Since there is only one alignment for each of these strobes, then a move is deterministic (it moves from an error to the alignment or vice versa). Since there are at least two alignments with this characteristic, then moving the alignments of all such strobes cannot be done with a local move.

Algorithm 4 makeJumpMove($A(\mathbf{S})$,M)

$M' \Leftarrow M$
Select connected component G_k uniformly from $\tilde{G}$.
for $S \in G_k$ **do**
    if $|A(S)| = 1$ **then**
        if $M'(S) = \emptyset$ **then**
            $M'(S) \Leftarrow A(S)$
        else
            $M'(S) \Leftarrow \emptyset$
        end if
    end if
end for
return M'

1.5.2 The Proposal Distribution

Observe that for any two mapping vectors M and M', there can be at most *one* type of move, either the local move or the jump move, which could feasibly transition between vectors. Thus, the proposal distribution is described as follows:

$$q(M'|M) = \begin{cases} 1 & \text{if } M = M' \text{ (the lazy chain).} \\ q_{local}(M'|M) & \text{if we can move from } M \text{ to } M' \text{ using a local move.} \\ q_{jump}(M'|M) & \text{if we can move from } M \text{ to } M' \text{ using a jump move.} \end{cases} \quad (6)$$

Below we describe the calculations required for q_{local} and q_{jump}.

q_{local}: Suppose that we have made a move by selecting multi-read S with probability $1/n$. Thus, we have proposed a mapping vector A' after calling makeLocalMove($A(\mathbf{S})$,M). Note that M and M' differ only by $M(S)$ and $M'(S)$, and multi-breakpoint read S has $|A(S)|$ possible alignments. The probability of proposing M' from M is

$$q_{local}(M'|M) = \frac{1}{n} \times \begin{cases} 1 & \text{if } M(S) \neq \emptyset \text{ and } M'(S) = \emptyset \text{ and } |A(S)| = 1 \\ p_{err} & \text{if } M(S) \neq \emptyset \text{ and } M'(S) = \emptyset \text{ and } |A(S)| > 1 \\ \frac{1}{|A(S)|} & \text{if } M(S) = \emptyset \text{ and } M'(S) \neq \emptyset \\ \frac{1-p_{err}}{|A(S)|-1} & \text{if } M(S) \neq \emptyset \text{ and } M'(S) \neq \emptyset \end{cases}.$$

Conversely, the probability of proposing M from M' is

$$q_{local}(M|M') = \frac{1}{n} \times \begin{cases} 1 & \text{if } M(S) = \emptyset \text{ and } M'(S) \neq \emptyset \text{ and } |A(S)| = 1 \\ p_{err} & \text{if } M(S) = \emptyset \text{ and } M'(S) \neq \emptyset \text{ and } |A(S)| > 1 \\ \frac{1}{|A(S)|} & \text{if } M(S) \neq \emptyset \text{ and } M'(S) = \emptyset \\ \frac{1-p_{err}}{|A(S)|-1} & \text{if } M(S) = j \text{ and } M'(S) \neq \emptyset \end{cases}.$$

q_{jump}**:** Suppose that we have made a move by selecting connected component G_k with probability $1/|\tilde{G}|$. Thus, we have proposed a mapping vector M' after calling makeJumpMove($A(\mathbf{S})$,M). By definition, all of the multi-reads with unique alignments that support G_k are flipped (if they are errors in M they are set to the alignment in M', and if they are alignments in M they are set to errors in M'). Thus,

$$q_{jump}(M|M') = q_{jump}(M'|M) = \frac{1}{|\tilde{G}|}.$$

1.5.3 Independent Subproblems and Sampling from the Chain

Running MCMC(D,z) on the entire space of multi-reads $\mathbf{S}$ would require a very long convergence time, since the number of possible solutions grows exponentially with the number of multi-breakpoint reads . However, the repetitive nature of the reference genome often results in mutually exclusive sets of events to decipher. This allows us to divide $\mathbf{S}$ into independent subsets for which the MCMC algorithm can be run in parallel.

A set of multi-reads $\bar{S} \subseteq \mathbf{S}$ is *independent* of all other multi-reads if all alignments from the corresponding multi-breakpoint-mappings appear in clusters with only other multi-breakpoint-mappings from multi-reads from $\bar{S}$. More formally,

$$S, S' \in \bar{S} \text{ if } \exists\ v = \{d_1, d_2, \ldots\} \in V : d_i \in A(S) \text{ and } d_j \in A(S'),$$

where v is a node in the cluster diagram G. We run the MCMC chain for two to ten million iterations for each independent subset, depending on the size of the subproblem:

1. **Very Short** jobs are run for 500,000 iterations; they contain ≤ 5 discordant consecutive read pairs.

2. **Short** jobs are run for two million iterations: they contain ≤ 10 discordant consecutive read pairs.

3. **Medium** jobs are run for four million iterations: they contain between 11 and 500 discordant consecutive read pairs.

4. **Long** jobs are run for twenty million iterations: they contain more than 500 discordant consecutive read pairs.

The numbers above were selected to balance the number of jobs for $30X$ coverage simulations; the low-coverage multi-read simulations contain mostly short jobs. Note that the number of discordant consecutive read pairs does not directly correlate to the number of multi-reads (the

size of $\bar{S}$). Some subproblems contain few multi-reads, each with many ambiguous alignments; these may be considered medium or long jobs. Other subproblems contain many multi-reads with unique read alignments; these may be considered short jobs.

Burnin Methods Rather than recording all mapping vectors M sampled by the chain, in practice we run the chain for a number of iterations before we start recording the sampled states. In all experiments we had a burnin time of 90% of the iterations, and recorded the last 10%. We also tested our method with a shorter burnin (10% burnin rate) and a thinning method (sampling every 100,000th iteration); however, the longer burnin of 90% produced the most stable results due to the convergence of the chain.

1.6 Final Adjacency Predictions and Benchmarking

The MCMC method returns a set of $z+1$ mapping vectors M that are sampled with probability proportional to $P(M|D)$. However, we wish to select a subset of nodes $\hat{\mathbf{U}} \subseteq V(G)$ that denotes the predicted adjacencies from the candidate cluster diagrams. There are many ways to perform this transformation: one straightforward way is to select a mapping vector $\hat{A}$ and use the $\Phi(G|\hat{A})$ function to get an adjacency vector. One may select $\hat{A}$ as the maximum likelihood estimate or by thresholding the mapping frequencies, for example. However, this metric relies on a single $\hat{A}$ to describe the entire space and thus it has limited power. Instead, we have developed a metric that utilizes information from the entire space.

To determine the precision and accuracy of our predictions, we establish two different metrics to determine which predictions are correct. The first metric counts true positives at an adjacency breakpoint level, which is used in the experiments with real data where we have a reported novel adjacency coordinate. The second metric counts true positives at an alignment level, which is used in our simulation experiments where we know where each multi-breakpoint read comes from in the constructed genome.

1.6.1 Computing the Probability of Adjacencies

A discordant pair d may appear in a number of mappings M; thus the probability $P(d)$ of the discordant pair is computed from the mapping probabilities $P(M|D)$,

$$P(d) = \sum_{d \in P_{\text{discord}}(M)} P(M|D), \tag{7}$$

where $P_{\text{discord}}(M)$ are the discordant pairs present in mapping M.

The probability $P(d)$ is calculated so that if many multi-breakpoint-mappings are sampled with relatively low frequency but they all contain the same discordant pair d, the probability of that pair will be large. Here, we describe how to take these discordant pair probabilities and use them to calculate the probability of a particular adjacency, represented as a node v in the cluster diagram G.

Each node $v \in V$ represents a set of discordant pairs in P_{discord}. Consider the leaf nodes $l \in V : outdeg(l) = 0$: these represent maximal sets of discordant pairs in the cluster diagram. This notion of maximal is similar to [9]; there are no discordant pairs to add to the set that will still have a non-empty area of intersection in $\mathbb{R}^2$. We would like to compute the probability $P(l|\text{supported by } k)$ that the leaf l was sampled with support k. The probability $P(l|\text{supported by } k)$ for $k = 1, \ldots, k_{max}$ is then the probability of all combinations of $|l|$-choose-k selections of discordant pairs in the node. Formally, let $\{b, \ldots, b_{|l|}\}$ be a binary vector that determines which discordant pairs are selected for node l.

$$P(l|\text{supported by } k) = \sum_{\substack{\{b_1,\ldots,b_{|l|}\}:\\ b_i\in\{0,1\},\\ \sum_i b_i = k}} \left[\prod_{i:b_i=1} P(d_i) \prod_{i:b_i=0} (1 - P(d_i)) \right].$$

This is the probability that there are k correct pairs and $|l|-k$ incorrect pairs in l. If $k > |r|$, then this probability is zero. Note that $P(v|\text{supported by } k)$ includes terms only for the parents of v; however, if we believe that v is the correct variant, then all discordant pairs in the connected component that are *not* in v should be assigned elsewhere. Thus, after we have computed the probabilities for all nodes in G, we adjust them by multiplying them by the pairs that should be assigned elsewhere. Define $\bar{d_v}$ as the pairs that are in nodes in the connected component of v but does not appear in v itself. The final probability for each node $v \in V$ and support $k = 1, \ldots, k_{max}$ is then

$$P(l|\text{supported by } k) = \sum_{\substack{\{b_1,\ldots,b_{|l|}\}:\\ b_i\in\{0,1\},\\ \sum_i b_i = k}} \left[\prod_{i:b_i=1} P(d_i) \prod_{i:b_i=0} (1 - P(d_i)) \right] \prod_{d\in\bar{d_l}} (1 - P(d)).$$

Example Consider a node v in the cluster diagram as the associated set of discordant pairs:

$$v = \{d_1, d_2, \ldots d_{|v|}\} \subseteq P_{\text{discord}}.$$

If v is a true novel adjacency, then we assume that mappings with at least some of the discordant pairs in v are highly probable. Other discordant pairs *not* in v that intersect at least one of $\{d_1, d_2, \ldots d_{n_v}\}$ must be incorrect, so we assume that mappings containing these pairs are less probable. Call these pairs $\bar{v} = \{\bar{d}_1, \bar{d}_2, \ldots, \bar{d}_{|\bar{v}|}\}$. As a concrete example, consider the right child in the cluster diagram in Figure 5 bottom. Here, there are four discordant pairs in $v : \{d_1, d_2, d_3, d_4\}$, and one discordant pair in $\bar{v} : \{d_5\}$.

We wish to compute the probability that v is supported by k discordant pairs for $k = 0, 1, \ldots, |v|$. First, however, we must determine the probability of a single discordant pair $d \in P_{\text{discord}}$ The discordant pair d may appear in a number of mappings M; so we simply count the number of times the discordant pair appears in the chain of $\{M^{(0)}, M^{(1)}, M^{(2)}, \ldots, M^{(z-1)}, M^{(z)}\}$:

$$P(d) = \frac{1}{z+1} \sum_{i=0}^{z} \mathbf{1}_{d\in P_{\text{discord}}(M^{(i)})}.$$

Note that the chain may be thinned or have a burnin time. To compute the probability that v is supported by k discordant pairs, we could compute the probability of all $\binom{|v|}{k}$ options. In our example above, for $k = 3$ we compute the following:

$$\begin{aligned} P(v =&\{d_1, d_2, d_3, d_4\}, \bar{v} = \{d_5\} \text{ is supported by 3 discordant pairs}) = \\ &P(\text{select } d_1, d_2, d_3 \text{ and not } d_4, d_5)+ \\ &P(\text{select } d_1, d_2, d_4 \text{ and not } d_3, d_5)+ \\ &P(\text{select } d_2, d_3, d_4 \text{ and not } d_1, d_5). \end{aligned}$$

We assume the discordant pairs are independent, so each term is simply a product of discordant pair probabilities determined by Equation 8. For nodes with many discordant pairs in v, however, this enumerative method is prohibitively slow. Instead, we have a dynamic program takes as input v and $\bar{v}$ and returns the probability that v is supported by k discordant pairs for

$k = 0, 1, \ldots, |v|$. Once we have the probability that v is supported by k discordant pairs, the probability of the variant depends on the coverage and sequencing technology. After testing a range of parameters (data not shown), we use the following tail probabilities:

$$P(v, \bar{v}|\text{strobe data}) = \sum_{k=1}^{|v|} P(v, \bar{v} \text{ supported by } k \text{ discordant pairs}) \tag{8}$$

$$P(v, \bar{v}|\text{paired data}) = \sum_{k=10}^{|v|} P(v, \bar{v} \text{ supported by } k \text{ discordant pairs}) \tag{9}$$

Figure 6 shows a simplified version of a cluster diagram from the 1000 Genomes Simulation. First, we compute the probability of each multi-read alignment using the sampled mappings. We then combined the multi-read alignments to compute the probability that each cluster is supported by exactly k multi-reads; note that this implies that the other multi-read alignments that are not members of the cluster are *not* selected. Then, the tail probability that the cluster is supported by $> t$ multi-reads is computed; the cluster with the largest tail probability is selected as the predicted cluster.

Dynamic Program for Efficient Computation of Node Probabilities If we calculated $P(l|\text{supported by } k)$ naively for each leaf node $l \in G$, then we will re-calculate the same terms multiple times. Specifically, the other nodes in G are terms that are shared by multiple calculations at the leafs. We use this hierarchy to recursively calculate this probability. The probability $P(v|\text{supported by } k)$ for a node $v \in G$ is computed after the probabilities have been computed for all parent nodes $\{g_1, g_2, \ldots, g_G\}$. Each parent g_i may have a different support k_i; we take the product of all combinations of the parents such that the supports sum to k:

$$P(v|\text{supported by } k) = \sum_{\substack{\{k_1,\ldots,k_G\}: \\ 0 \le k_i \le |g_i|, \\ \sum_i^G k_i = k}} \left[\prod_i^G p(g_i|\text{supported by } k_i) \right].$$

We can compute this efficiently using a dynamic program. The dynamic program takes as input v and $\bar{v}$ and returns the probability that v is supported by k discordant pairs for $k = 0, 1, \ldots, |v|$ (Algorithm 5). The method fills a 0-indexed table T, where T_{ij} is the probability that the node is supported by i discordant pairs out of the first j pairs (which are arbitrarily ordered). Lines 2-6 in NodeProbability(v,$\bar{v}$) are initializations for the recurrence. There are two ways that the node can be supported by i discordant pairs in the first j pairs in the recursive step (Line 9): either the jth pair is included in the count (and we use the probability in $T_{(i-1)(j-1)}$ or it is not (and we use the probability in $T_{(i-1)j}$). Lines 12-14 account for the discordant pairs in $\bar{v}$ by multiplying T by the probability that these are not selected (as these would denote other nodes in the cluster diagram G).

Once we have the probability that v is supported by k discordant pairs, the probability that v is supported by k or more discordant pairs is simply the tail of this distribution:

$$P(v \text{ has } \geq k \text{ discordant pairs}) = \sum_{i=k}^{|v|} T_{kj}. \tag{10}$$

1.6.2 Variant Calling Accuracy

From a set of clusters (e.g. the nodes in the cluster diagram G), we determine the subset of clusters that represent the true deletions in the individual genome. Each true deletion in the

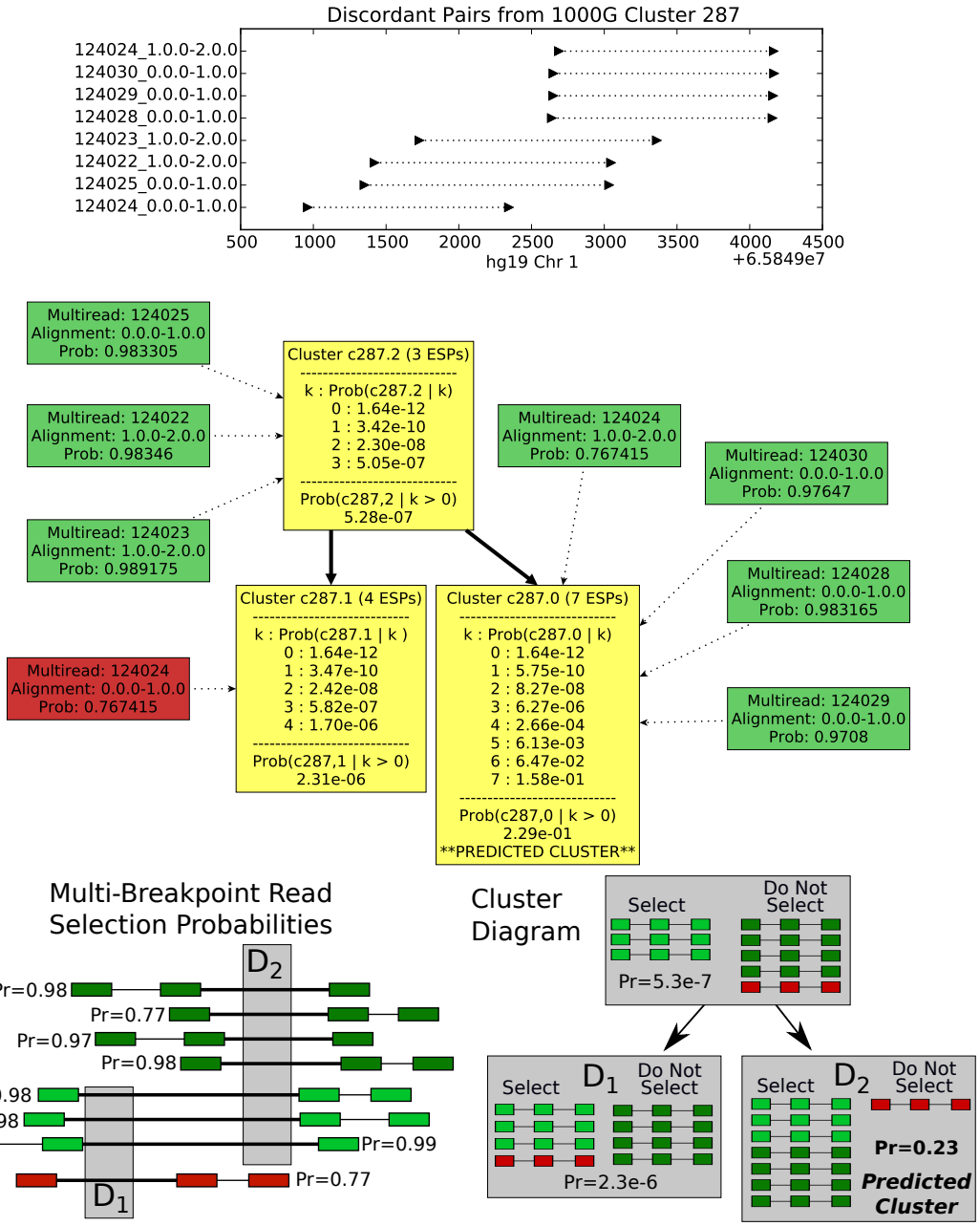

Figure 6: Cluster Diagram from 1000 Genomes Simulation. (Top) Discordant Pairs that cluster to produce the cluster diagram. (Middle) Cluster Diagram as generated by MultiBreak-SV. Yellow nodes and solid lines are in the cluster diagram, green nodes are true positive multi-read alignment, and the red node is a false positive multi-read alignment. For each cluster, we first compute the probability of the cluster given that it is supported by *exactly* k multi-reads. We then compute the probability of the cluster given that it is supported by $> t$ multi-reads (in this example, $t = 0$). The predicted cluster is the one with the largest tail probability; in this case, the predicted cluster contains all the true positive multi-read alignments and avoids the one false positive multi-read alignment. (Bottom) Schematic of the same cluster diagram. (Left) 8 multi-breakpoint read mappings, 7 correct (green) and 1 incorrect (red), result in three different clusters (light green, light and dark green, light green and red). First, we compute the probability of observing each multi-breakpoint read mapping in the solution space from the MCMC method. (Right) Then, we compute the probability of each novel adjacency prediction by computing the probability that we select at least one multi-breakpoint read mapping that supports the cluster and we select *none* of the mappings that support another cluster. The adjacency with the largest probability, D_2, is selected as the predicted cluster.

Algorithm 5 NodeProbability(v,$\bar{v}$)

T is a $(|v|+1)$-by-$(|v|+1)$ table
$T_{00} \Leftarrow 1$
$T_{ij} \Leftarrow 0$ for $i > j$
for $j \Leftarrow 1$ to $|v|$ **do**
  $T_{0j} \Leftarrow T_{0(j-1)} \times (1 - P(d_j))$
end for
for $i \Leftarrow 1$ to $|v|$ **do**
  for $j \Leftarrow i$ to $|v|$ **do**
    $T_{ij} = T_{(i-1)(j-1)} \times P(d_j) + T_{i(j-1)} \times (1 - P(d_j))$
  end for
end for
for $i \Leftarrow 0$ to $|v|$ **do**
  $T_{i|v|} \Leftarrow T_{i|v|} \times \prod_{\bar{d} \in \bar{v}} (1 - P(\bar{d}))$
end for
return $T_{i|v|}$ for $0 \leq i \leq |v|$

individual genome corresponds to a pair of coordinates (x, y) in the reference genome. Thus, any clusters whose area of intersection contain (x, y) are considered true positive novel adjacencies. However, small (<20bp) indels and slightly-shifted read alignments to the reference may cause the correct cluster to not contain (x, y). Thus, we consider a more "relaxed" definition of a true positive cluster, modified from the 'double-uncertainty metric' introduced in [10]. This metric captures the uncertainty in the novel adjacency coordinates and the uncertainty in the adjacency predictions; thus it is ideal for real sequencing data from relatively well-characterized individual genomes (such as the Fosmid simulations).

From a true adjacency (x, y), we construct a square $\pm L_{\max}/2$ around the point in $\mathbb{R}^2$. $L_{\max}$ depends on the sequencing technology; thus, $L_{\max}$ is larger for multi-reads than for paired-end reads. Each node $v \in V$ in the cluster diagram corresponds to a polygon in $\mathbb{R}^2$, indicating the possible adjacency coordinates that are supported by the associated discordant pairs. We call cluster v a true positive for the true adjacency (x, y) if these two polygons (the square and the predicted adjacency region) overlap. In the example in Figure 5 Bottom, all three nodes in the cluster diagram might be considered true positives depending on the size of the square around the point (x, y). When we calculate the number of true positive clusters from a set of candidates, we count the number of *true deletions* that are covered by some true positive cluster; in other words, we do not over-count true positive clusters if multiple clusters cover the same true deletion.

To compute ROC and Precision-Recall curves, we select a minimum support t and compute the tail of the cluster probability:

$$P(v|\text{supported by } \geq t). \tag{11}$$

For each connected component in $G_i \in G$, we select the node v_i with the largest cumulative probability from (11). We then sort these nodes in decreasing order and calculate the number of true positives and false positives at each step as we walk down this list. We only count a true positive once: that is, if there are multiple adjacencies that all support the same correct prediction, they count as a single true positive and zero false positives (this is similar to [10]). Thus, the y-axis has an upper bound of the number of true positive deletions we wish to detect.

1.6.3 Mapping Accuracy

For each read's alignments $A(R_i^{(S)})$ from multi-read S, we determine at most one alignment that represents the correct mapping to the reference. We only determine this type of true positive for simulated experiments, where we have explicitly constructed the individual genome from rearranging the reference. For a simulated read $R_i^{(S)}$, we first take the coordinates $(z, z+1)$ in the individual genome and determine the coordinates (x, y) in the reference genome. We then consider all alignments $a \in A(R_i^{(S)})$, and select the alignment a that meets the following criteria:

1. a must be on the same strand as the simulated read $R_i^{(S)}$.
2. a must be at least 80% of the length of the simulated read $R_i^{(S)}$ (it cannot be a "partial alignment").
3. a defines an interval $[x', y']$ on the reference. The following must hold:

$$\frac{[x, y] \cap [x', y']}{[x, y] \cup [x', y']} \geq 0.8.$$

The last item ensures that the two intervals overlap by a large amount, while allowing for slightly misaligned reads. This is useful because BLASR tends to clip the basepairs at the extremities of the reads, producing an alignment that is slightly less than the actual read length. If there are multiple alignments that satisfy the conditions above, the alignment with the largest overlap (Item 3) is selected as the correct mapping to the reference.

To compute ROC and Precision-Recall curves, we compute $P(d)$ for each discordant pair and sort them in decreasing order. A discordant consecutive read pair d is a true positive if *both* read alignments are true positives according to the criteria above.

1.7 Generalizing to Multiple Data Types

Suppose, rather than one set of multi-breakpoint reads $\mathbf{S}$, you have E sets of multi-breakpoint reads $\{\mathbf{S}^{(1)}, \ldots \mathbf{S}^{(E)}\}$ corresponding to E different experiments. $\mathbf{S}^{(e)}$ may also be paired-read data from next generation sequencing technologies, which are translated into 2-read multi-breakpoint reads. For each experiment $1 \leq e \leq E$, we have the following information:

1. $M(\mathbf{S}^{(e)})$: the set of multi-breakpoint read alignments
2. $\lambda_d^{(e)}$: the expected number of multi-breakpoint reads that supports an adjacency
3. $p_{seq}^{(e)}$: the sequencing error rate
4. $p_{err}^{(e)}$: the probability that a multi-breakpoint read is not aligned

We cluster the union of the multi-breakpoint read alignments $\{A(\mathbf{S})^{(1)}, \ldots, A(\mathbf{S})^{(E)}\}$ and get a cluster diagram $G = (V, E)$. For a node $v \in V$, let $v^{(e)}$ be the set of pairs that belong to experiment e. Finally, for a mapping vector M let $M^{(e)}$ be the set of indices that correspond to multi-breakpoint reads from experiment e. Then,

$$P(M|A(\mathbf{S})^{(1)}, \ldots, A(\mathbf{S})^{(E)}, G) \propto P(M)P(A(\mathbf{S})^{(1)}, \ldots, A(\mathbf{S})^{(E)}|M)P(G|M) \tag{12}$$

$$= \prod_{e=1}^{E} \left[B\left(\epsilon(M^{(e)}); \ell(M^{(e)}), p_{seq}^{(e)}\right) \times {p_{err}^{(e)}}^{|\{i:M(S)^{(e)}=\emptyset\}|} \times \prod_{u \in U_M^{(e)}} \text{Poiss}(|u|; \lambda_d^{(e)}) \right] \tag{13}$$

Note that if multi-breakpoint reads from one experiment support a novel adjacency, then the probability of that adjacency is computed for *all* experiments. The MCMC algorithm is applied as before.

To select the final set of adjacencies, instead of computing $P(v|\text{supported by } \geq t)$ for a single threshold t, we have a set of thresholds $t_1, \ldots, t_E$ that corresponds to the coverages of each of the experiments. The probability of an adjacency is then

$$P(v|\text{supp. by } \geq t_e \text{ for } e = 1, \ldots, E) = \frac{1}{E} \sum_{e=1}^{E} \sum_{k=t}^{|v^{(e)}|} P(v^{(e)}|\text{supp. by } k). \tag{14}$$

The results from each experiment are weighted equally above.

1.8 Dataset Processing

1.8.1 Determining Multi-Breakpoint-Mappings from Long Reads

Long read alignments might produce "split reads," which indicate a structural variant *within* the read itself (rather than within the advance length, as described above with τ-multi-reads where $\tau > 1$). As sequenced reads become longer, the opportunity to capture multiple, nearby variants within single reads motivates the need for a 'multiply-split read' framework. In this scenario, each long read is an instance of a 1-multi-read, where the partial alignments are pieced together to produce a t-multi-breakpoint reads that each indicate some number of novel adjacencies (see Figure 1). Unlike multi-reads, however, candidate alignments for each long read may be split in different locations within the read.

1. Determine multi-breakpoint-mappings from partial alignments. Algorithm 6 enumerates all possible virtual t-multi-breakpoint-mappings from a set of partial alignments $A(R)$ from 1-multi-read R. The algorithm also takes a user-specified parameter h, which denotes the minimum length of the non-overlapping portions of alignments when considering whether to consider a pair of alignments in the same multi-breakpoint-mapping. We use an overhang threshold h rather than a minimum overlapping threshold because we found that exact repeats at the novel adjacency coordinates produced alignments that overlapped many basepairs (e.g., a deletion within an Alu might have overlapping alignments spanning hundreds of basepairs). We set $h = 100$, requiring at least 100bp of each read does not overlap in the alignments.

First, we create a directed graph (G) with vertices corresponding to the alignments $A(R)$, and edges corresponding to allowed alignment pairs. For every pair of alignments we look at the query start and query end positions. If the query end of the left alignment is less than the query start of the right alignment (or they overlap but have at least h non-overlapping basepairs on either end) then create an edge. We then traverse this graph in a depth-first manner, retrieving multi-breakpoint-mapping alignments.

In Algorithm 6, an alignment a has coordinates in the long read queryStart(a) and queryEnd(a). We first construct the graph G, and then enumerate all paths via depth-first traversal. When constructing the graph G, two alignments a, a' overlap if they share coordinates in the long read. a overhangs by $\geq h$ basepairs if queryStart(a')-queryStart(a) $\geq h$. a' overhangs by $\geq h$ basepairs if queryEnd(a')-queryEnd(a') $>\geq h$. The traversal step calls AddAlignmentPaths(), which recursively constructs the multi-breakpoint-mappings.

2. Determine potential deletion coordinates within full alignments. The top alignment of each read was evaluated using a three-state hidden Markov model (HMM). In short, a pairwise BLASR alignment of length l was converted into a new string of length l, corresponding to 3 character states (0 deletion, 1 match/mismatch, 2 insertion). For the match state emission probabilities, we fit a 20% error (symmetrically for insertions and deletions), thus 0.8

Algorithm 6 ConstructMultiBreakpointMappings($A(R)$, h)

$V \leftarrow A(R)$
$E \leftarrow \emptyset$
//First, get edges for graph G
for $a \in A(R)$ **do**
  for $a' \in A(R)$ **do**
    if (a, a' do not overlap and queryEnd(a) < queryStart(a')) or
    (a, a' overlap and a overhangs by $\geq h$ bp and a' overhangs by $\geq h$ bp) **then**
      $E \cup (a, a')$
    end if
  end for
end for
$G = (V, E)$
//Then, enumerate all paths via depth-first traversal
$A \leftarrow \emptyset$
for $a \in V$ **do**
  $P \leftarrow \emptyset$
  $A \leftarrow$ AddAlignmentPaths(G, P, A, a)
end for
return A

Algorithm 7 AddAlignmentPaths($G = (V, E), P, A, u$)

$P \leftarrow P \cup u$
$Adj[u] \leftarrow \{v : (u, v) \in E\}$
if $|Adj[u]| > 0$ **then**
  for $v \in Adj[u]$ **do**
    return AddAlignmentPaths(G, P, A, v)
  end for
else
  return $A \cup P$
end if

Dataset	MultiBreak-SV Hyperparameters		
	p_{seq}	λ_d	p_{err}
Venter Strobes 5X	0.15	4	0.01
Venter Pairs 30X	0.01	18	0.01
Venter Hybrid 2X/30X	0.15/0.01	2/18	0.01
Venter Long 5X	0.15	3	0.01
Fosmids 9X-32X	0.15	5-20	0.005-0.15
CHM1TERT 10X	0.15	8	0.01

Table 1: MultiBreak-SV parameters for all datasets. The Fosmids were used to evaluate parameter robustness.

probability of emitting a match state. For insertions and deletions we used a 0.9 probability of emitting their respective states. We allowed a 0.01 probability of transition from insertion and deletion states to a match state and a strict 1×10^{-10} probability to transition from a match state to either insertion or deletion states. Initialization and termination states were both enforced to be match states, and the Viterbi path was selected to identify potential insertions and deletions. Only deletions greater than 200 base pairs were passed for downstream analysis.

Additional filters. We ignored any t-multi-breakpoint-mapping where $t = 1$ (that is, it aligns in one contiguous piece to the reference), since these are considered concordant multi-reads in the long read framework. We also removed any multi-breakpoint-mapping that aligned within 1Mb of either end of the chromosomes or within 1Mb of the centromere to remove telomeric/centromeric mappings.

From multi-breakpoint-reads to discordant pairs for GASV clustering. From these multi-breakpoint-mappings, we determine concordant and discordant consecutive read pairs, where here the "read pairs" are consecutive partial alignments from the same long read. While strobe and paired-read sequence data considers the advance length to determine $L_{\min}$ and $L_{\max}$, we take the outer coordinates and use the read lengths to compute $L_{\min}$ and $L_{\max}$. For example, suppose we have a consecutive read pair $(a_1, a_2) \in P_{\text{discord}}$; instead of taking the rightmost coordinate of a_1 and the leftmost coordinate of a_2, we take the leftmost coordinate of a_1 and the rightmost coordinate of a_2. Since these are approximating split reads, we expect that the novel adjacency implied by these coordinates still results in a split read; thus we set $L_{\min}$ and $L_{\max}$ where $L_{\max} - L_{\min} = 100$; that is, the implied novel adjacency must be within 50bp of the long read length. Note that, since $L_{\min}$ and $L_{\max}$ are computed with respect to the read lengths, they are different for every discordant pair; we bin the values (in steps of 100bp) to pass to GASV.

1.8.2 MultiBreak-SV Hyperparameters and GASV Clustering Statistics

1.8.3 Comparing Algorithms for Structural Variation Prediction

We compared the results of MultiBreak-SV applied to the Strobe datasets to that of [8], the only other known SV detection algorithm for strobe sequencing. [8] formulated the problem as an optimization problem that minimized the total number of adjacencies given the data, and designed an Integer Linear Program (ILP) to find a single optimal solution. The ILP method thresholds by the minimum support during a graph construction phase. To generate curves for the ROC and Precision-Recall plots, we ran the ILP method 14 times ranging the minimum support from $2, 3, 4, \ldots, 15$. The 15 points are connected to form a curve.

Dataset	# Multi-Reads	# Discordant Multi-Breakpoint-Mappings (% Ambig)	# GASV Clusters	# Conn Comps from Cluster Diagrams (Largest CC Size)
Venter Strobes 5X	377912	1955 (33%)	358	60 (775)
Venter Pairs 30X	11818168	2270 (0%*)	146	4 (6)
Venter Hybrid 2X/30X	11967408	3594 (7%*)	279	45 (37)
Venter Long 5X	229377	24386 (70%)	18849	338 (38)
Fosmids 9X-32X	3099-15728	68-2628 (32%-95%)	1-2503	0-13 (129)
CHM1TERT 10X	3679463	304215 (48%)	245900	338 (171)

*We took unique alignments for paired-read data.

Table 2: Dataset preprocessing and GASV clustering statistics. The Venter simulations are aligned to hg18, the real datasets are aligned to hg19. The last column lists connected components with more than one node. BLASR was used to align all PacBio data; BWA was used to align all Illumina/paired-read data.

Previously-published methods such as GASV [9], GASV-Pro [10], VariationHunter [3], Delly [7] and Hydra [6] are designed for paired-read data. GASV-Pro and Hydra both accommodate paired-end reads with multiple alignments to the reference genome. We compare Strobes and Hybrid to a previously-published parsimony algorithm designed for multi-reads [8].

Algorithm	Paired Reads	Multi-Reads	Variant Calling Parameter	Mapping Parameter
MultiBreak-SV	✓	✓	Pr[support $\geq k$]	Pr[Alignment]
Parsimony-Based [8]	✓	✓	min. support k	min. support k
GASV [9]	✓		min. support k	min. support k
GASV-Pro [10]	✓		Pr[support $\geq k$]	Pr[Alignment]
Hydra [6]	✓		min. Hydra score	min. Hydra score
Delly [7]	✓		min. support k	min. support k
VariationHunter [3]	✓		min. support k	min. support k

Table 3: Algorithms for Comparison. The Variant Calling Parameter and the Mapping Parameter are thresholds that determine the number of adjacency and mapping predictions. Probability computations are described in the Methods.

2 Supplemental Results

2.1 Venter Simulation (Figure 7)

Venter Simulation ROC curves for variant calling accuracy and precision-recall curves for mapping accuracy. The curves are divided by (Row 1) MultiBreak-SV vs. Parsimony Solution for 1X,2X, and 5X strobes, (Row 2) MultiBreak-SV vs. Parsimony Solution for 1X,2X, and 5X hybrid datasets, (Row 3) the paired dataset run with a variety of methods, and (Row 4) all the MultiBreak-SV runs. GASVPro-HQ is the original predictions output by GASVPro on unique alignments, and GASVPro-HQ Pruned removes overlapping variants from the predictions. Hydra-HQ is the Hydra method applied to unique alignments. In addition, we also ran the probabilistic method on 60X coverage of pairs; strobe and hybrid datasets still outperform 60X pairs in terms of variant calling accuracy.

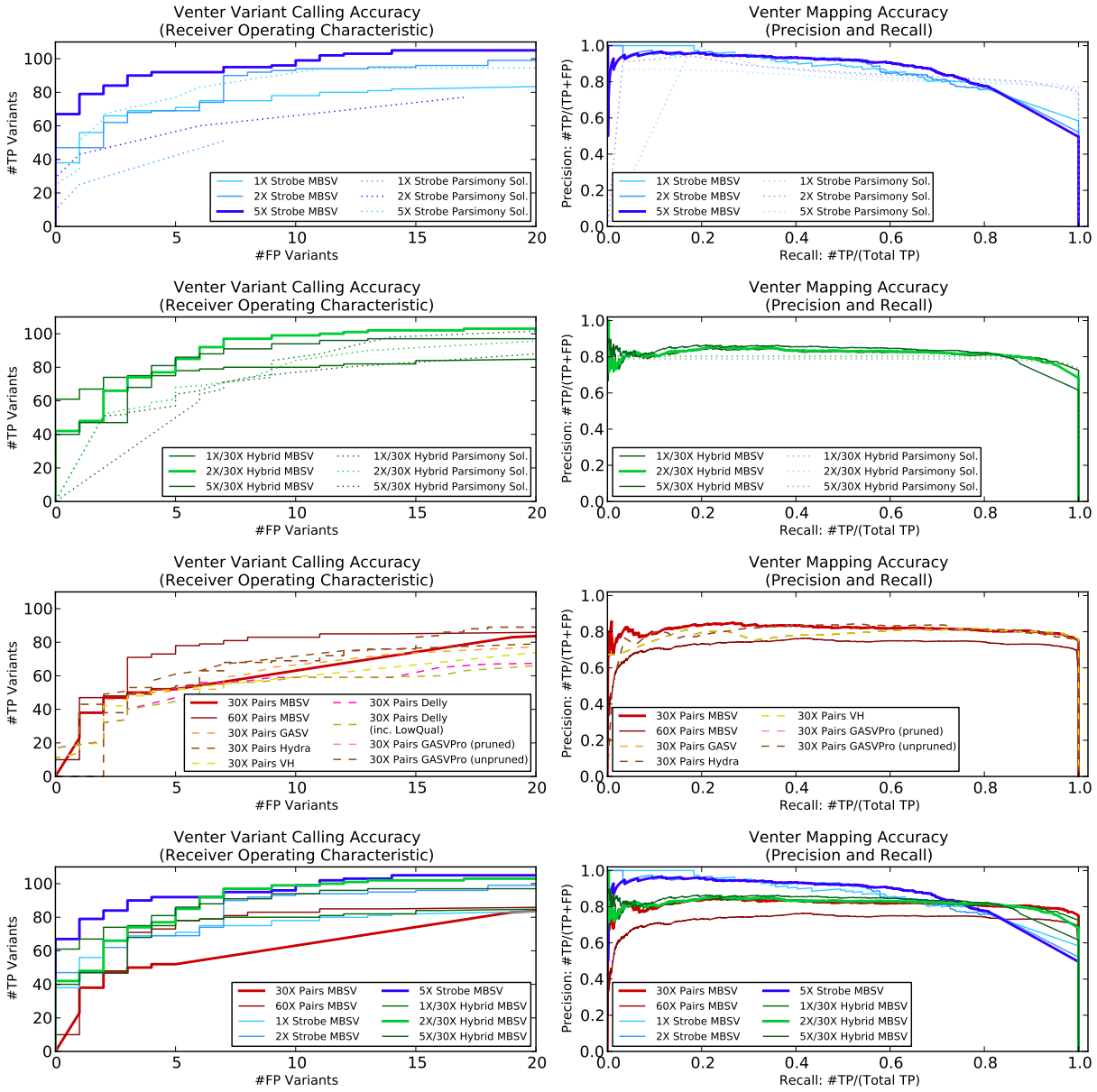

Figure 7: Venter Simulation.

Sub-Optimal mappings contribute to true positive novel adjacencies. We considered the set of sub-optimal multi-breakpoint-mappings that are true positives according to the mapping accuracy for deletions. Here, a mapping is sub-optimal if at least one of the reads is not the top hit by the BLASR score. For the 5X Strobe dataset, 13% (163 of 1069) discordant pairs that span a true positive deletion include at least one sub-optimal mapping. While this number is relatively modest, it shows that sub-optimal mappings span true deletions. For the 5X Long Read dataset, 97% (396 of 410) discordant pairs contain at least one sub-optimal mapping. Sub-optimal mappings from long reads may either come from sub-optimal read alignments *or* breakpoints identified within concordant alignments using the HMM. From these numbers, sub-optimal mappings are necessary for the long read data.

2.2 1000 Genomes Simulation

We generated a simulated dataset based on validated variants from the 1000 Genomes Project [1]. There are 734 deletions, 214 mobile element insertions, 26 tandem duplications, and 4 novel sequence insertions reported on chromosome 1 for individuals sequenced at low coverage and individual NA12878, which was sequenced to high coverage. We removed overlapping variants and inserted the remaining 794 rearrangements (557 deletions and 237 insertions) in the reference. Of these deletions, 219 (42.86%) have repetitive sequence spanning both of the novel adjacency coordinates.

We assessed the accuracy of MultiBreak-SV on the 1000Genomes data by generating Strobe, Pair, and Hybrid datasets similar to the Venter simulation. In general, all methods perform very well in terms of variant calling accuracy, predicting up to three false positives in order to recover over 450 true positive novel adjacencies (Figure 8 Left). The only outlier is the parsimony solution for strobe data, which requires about 50 false positives to achieve similar sensitivity as the other methods. All methods tend to also perform well in terms of mapping accuracy (Figure 8 Right); 30X Pairs MultiBreak-SV has slightly better precision (0.95) over the other methods (0.9) at all values of recall. The Hybrid dataset performs similarly since most of the mappings are from the 30X Pairs data.

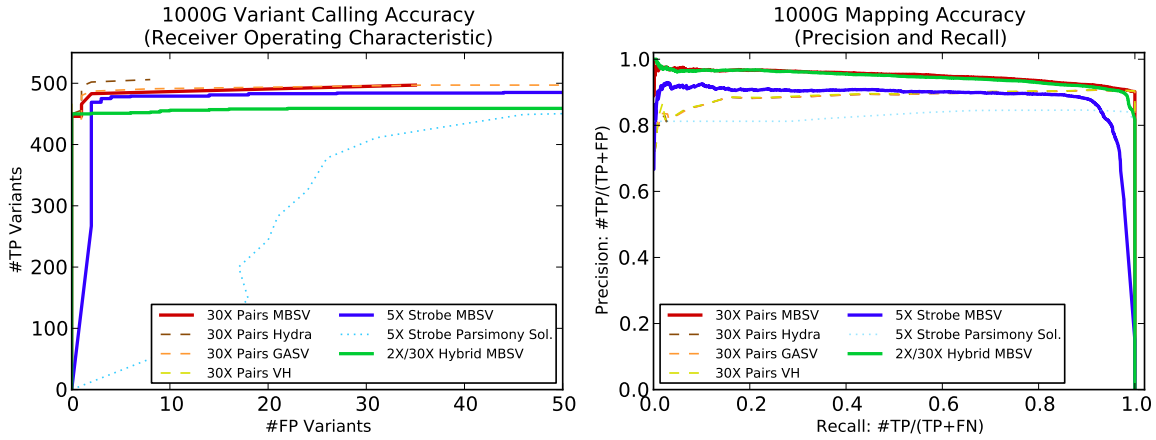

Figure 8: (Left) Variant calling accuracy and (Right) mapping accuracy for datasets from the 1000 Genomes simulation.

Plots for the 1000 Genomes Simulation for all methods and data coverages are in Figure 9. The curves are divided by (Row 1) MultiBreak-SV vs. Parsimony Solution for 1X,2X, and 5X strobes, (Row 2) MultiBreak-SV vs. Parsimony Solution for 1X,2X, and 5X hybrid datasets, (Row 3) the paired dataset run with a variety of methods, and (Row 4) all the MultiBreak-SV runs. GASVPro-HQ is the original predictions output by GASVPro on unique alignments, and GASVPro-HQ Pruned removes overlapping variants from the predictions. Hydra-HQ is the Hydra method applied to unique alignments. In addition, we also ran the probabilistic method on 60X coverage of pairs; strobe and hybrid datasets still outperform 60X pairs in terms of variant calling accuracy

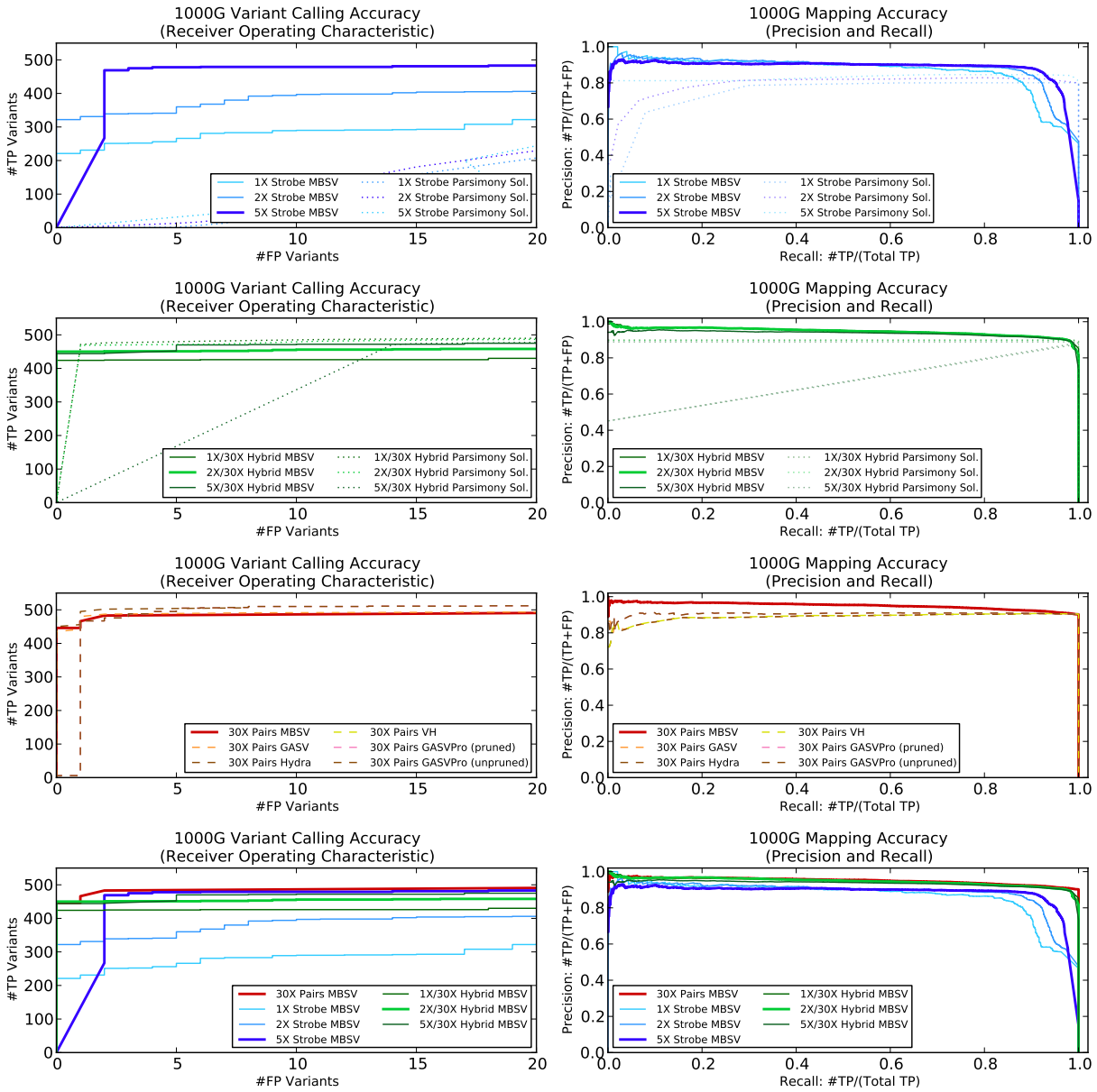

Figure 9: 1000 Genomes Simulation ROC and Precision-Recall curves.

Differences Between Venter and 1000G Simulations For the VENTER simulated chromosome, the 30X PAIRS dataset performs worse than the STROBES and HYBRID datasets, regardless of the method used. In contrast, 30X PAIRS performed well on 1000G simulated chromosome. All adjacencies in 1000G were determined from several different high-throughput, short-read sequencing technologies, and primarily from Illumina and SoLID sequencing. Thus it is reasonable to assume that simulated short reads will predict a majority of the variants. VENTER structural variants were established using older, Sanger sequencing technology. Of the 124 deletions in VENTER, 112 (90.32%) have repetitive sequence spanning both of the novel adjacency coordinates. Of the 511 deletions in 1000G, 219 (42.86%) have repetitive sequence spanning both novel adjacency coordinates. When BWA aligns reads in paired-mode, it tries to find unique paired-end read alignments consistent with the expected fragment distribution. Due to the repetitive sequence at the novel adjacency coordinates in VENTER, BWA mistakenly reports a concordant alignment for 48% of the multi-reads from 30X PAIRS that support a deletion. Compare this to an incorrect concordant alignment for 10% of the multi-reads from 30X PAIRS in the 1000G chromosome. There are also more false positive alignments in VENTER (25%) compared to 1000G (10%), which originate from more varied sources of error (Table 4). For these reasons, the 1000G simulation is "easier" to predict the detectable deletions than the VENTER simulation.

2.3 Alignment Analysis

2.3.1 BLASR Alignment Analysis

	1000GChr1	**VenterChr17**
Individual Chromosome Construction		
Chr	1	17
# insertions	557	8752
# deletions	237	8801
# inversione	0	4
Total # of SVs	794	17377
# deletions $\geq$ 120bp (**D**)	511	124
Simulated Paired Data		
# of fragments that span a del. in **D** (**S**)	16679	3466
Avg. # of fragments that span a deletion	32.6	29.5
BWA Alignments		
# of deletion ESPs > 600bp and < 1Mb	15300	1791
– # with correct coordinates (**TP**)	13802 (90.2%)	1345 (75.1%)
– # with incorrect coordinates (**FP**)	1498 (9.8%)	446 (24.9%)
# in **D** supported by at least one **TP**	499	93
Where do FPs Come From?		
–# from split read	1440 (96.1%)	284 (63.7%)
–# from read in novel sequence	55 (3.7%)	123 (27.6%)
–# that span another SV	1	39 (8.7%)
–# other	1	0
BWA Alignments for Fragments in S		
# with a deletion ESP alignment	13606 (81.6%)	1351 (39.0%)
– # with correct coordinates	13570 (99.7%)	1304 (96.5%)
# with a non-del. ESP alignment	22	16
# that are unmapped/qual < 10	1392 (8.4%)	639 (18.4%)
# with a concordant alignment	1659 (10.0%)	1659 (47.9%)
–# that are $\leq$ 600bp (truly concordant)	1120 (67.5%)	663 (40.0%)
–# that are > 600bp	537 (32.4%)	716 (43.2%)
–# with read in novel sequence	2	81 (4.9%)

Table 4: Simulation Statistics for 1000 Genomes and Venter simulations. Percentages are shown in parentheses when the value is > 1%. The **TP** and **FP** alignments are determined by the mapping accuracy.

2.3.2 BWA Alignment Analysis

Using the Venter Chr17 simulation, we assessed the usefulness of incorporating ambiguous paired-read alignments. We tested three alignment pipelines utilized by different publications (Figure 10).

1. **Novoalign** We aligned fragments that correctly span a deletion greater than 20bp in the individual genome with Novoalign, a more sensitive aligner than BWA that returns multiple alignments [5]. After computing Novoalign alignments for each read (single-read mode), we paired all vs. all alignments for each paired-read, which determines the set of discordant ESPs for clustering.

2. **Hydra Pipeline (BWA+Novoalign+Novoalign)** We ran the alignment pipeline from Hydra [6], which involves running BWA once to find unique alignments and then Novoalign

twice with the remaining reads, gaining sensitivity in the alignments at each step. [10] also used this method of collecting ambiguous alignments from paired-end reads.

3. **BWA>10** We take all unique alignments with a BWA mapping quality greater than 10. Note that this is more generous than many default parameters for SV detection, which usually thresholds the alignments at a mapping quality of 30.

There are two ways to run Hydra: Hydra-HQ selects the best alignment for each read, and Hydra with the "all" option (Hydra-All) includes all alignments for each read. When we run Novoalign on the correct fragments, Hydra performs well at detecting the structural variants, but the number of alignments produced by this pipeline is very large (the number of false positive pairs approach 60,000). When we run the Hydra pipeline, the number of false positive alignments is appropriately reduced, but over half of the true positive variants are lost due to the paired mode of BWA and Novoalign. Thus, we take the set of BWA alignments for the paired read datasets.

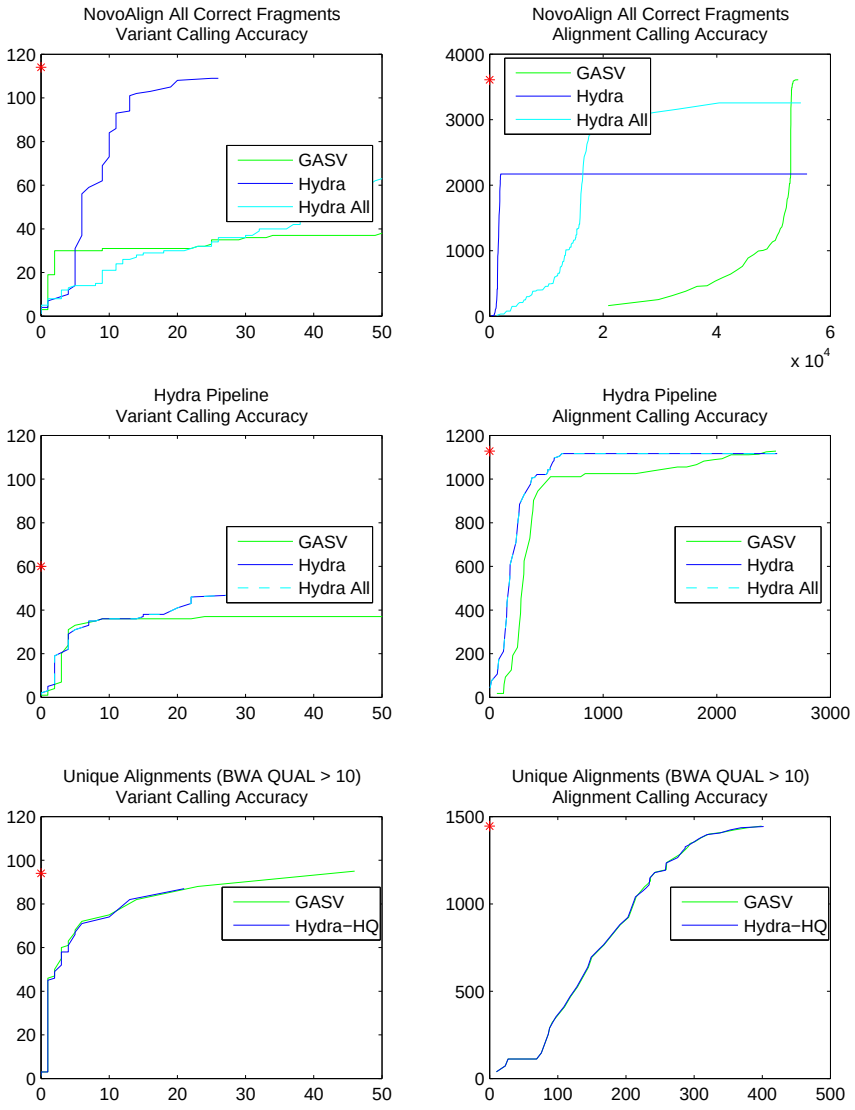

Figure 10: Recovery of true positive variants and alignments with various alignment pipelines on paired data. For each pipeline tested, we plot the variant calling accuracy (Left) and the alignment calling accuracy (right) for GASV, Hydra, and Hydra-All.

2.4 Sequenced Fosmids

2.4.1 Fosmid Selection Simulation

We simulated 3-strobes uniformly from the constructed individual chromosomes with read and advance lengths that mimic the real Fosmid data from D1. Table 5 shows the detectability of reported adjacencies for simulated strobes and simulated paired-read data.

We called a deletion or inversion *detectable* if a novel adjacency prediction with at least five supporting discordant pairs lies within fosmid's coordinates when aligned to the reference genome. Since the fosmids harbor adjacencies, a full alignment to the reference is not always obtainable. Thus, we find the best-scoring partial alignment of the fosmid to the reference and add a buffer of 100Kb. Since this region is much larger than the fosmid length, the detectability counts are conservative in the sense they may include false positive calls. We focused on two deletions (D1 and D2) that were detectable by both strobes and pairs (Figure 11) and two inversions (I1 and I2) that were detectable by the strobe simulation but *not* the paired simulation (Figure 12).

	Simulations for 44 Deletions			**Simulations for 19 Inversions**		
	Paired-End Detected	Paired-End Undetected	Total	Paired-End Detected	Paired-End Undetected	Total
Strobe Detected	25	10	35	4	9	13
Strobe Undetected	2	7	9	0	6	6
Total	27	17	44	4	15	19

Table 5: Detectability of Reported Adjacencies on Simulated Strobe and Paired-Read Data. (Left) Over half of the reported deletions are detectable by both paired-end and strobe datasets; we selected two of the fosmids harboring these deletions as controls. (Right) Strobes are able to detect nine more inversions than the paired-end dataset; we selected two of the fosmids harboring these deletions as "difficult" cases.

2.4.2 Fosmid MCMC Results

Figure 13 shows the distribution of sequence coverage for the four fosmids selected for sequencing. Figure 14 shows the results of MultiBreak-SV applied to the sequenced fosmids for different parameter values.

Name	Accession	Chr	Reported Start [4]*	Reported End [4]*	Reported Junction [4]	Detectable with Pairs?	Detectable with Strobes?
D1	AC158335	3	68739688	68747866	2	✓	✓
D2	AC153483	16	78371638	78384899	0	✓	✓
I1	AC195776	19	39264278	39280958	1236		✓
I2	AC193137	14	35017063	35031477	7063		✓

*Coordinates lifted over from hg18 to hg19.

Table 6: Fosmids Selected for Pacific Biosciences Sequencing. The reported junction is the length of unmatched sequence found across the adjacency [4]. The last two columns report whether simulations indicate that the variant is detectable with paired-read sequencing or strobe sequencing.

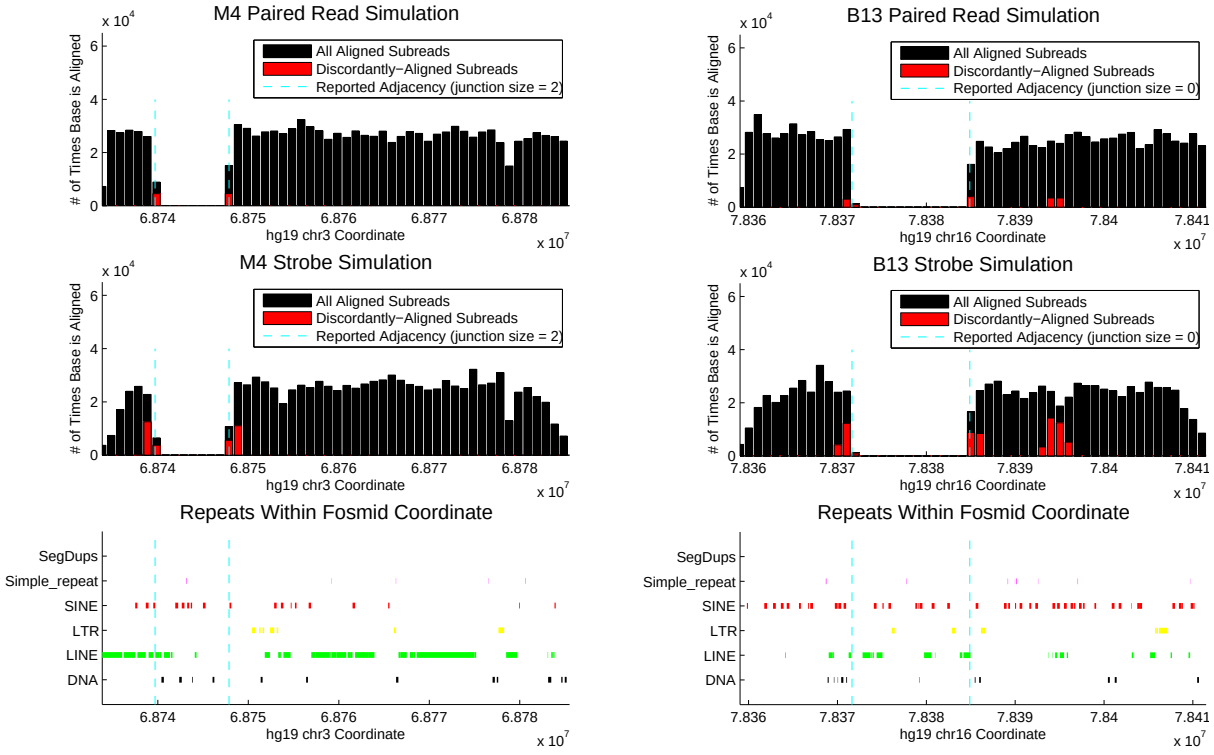

Figure 11: Paired Read and Strobe Simulations for Deletions D1 (M4) and D2 (B13). Both pairs and strobes have the ability to detect the deletions in simulation. (Top) Read Coverage for a paired read simulation. (Middle) Read Coverage for a strobe simulation. (Bottom) Repeats within the fosmid coordinate (hg19).

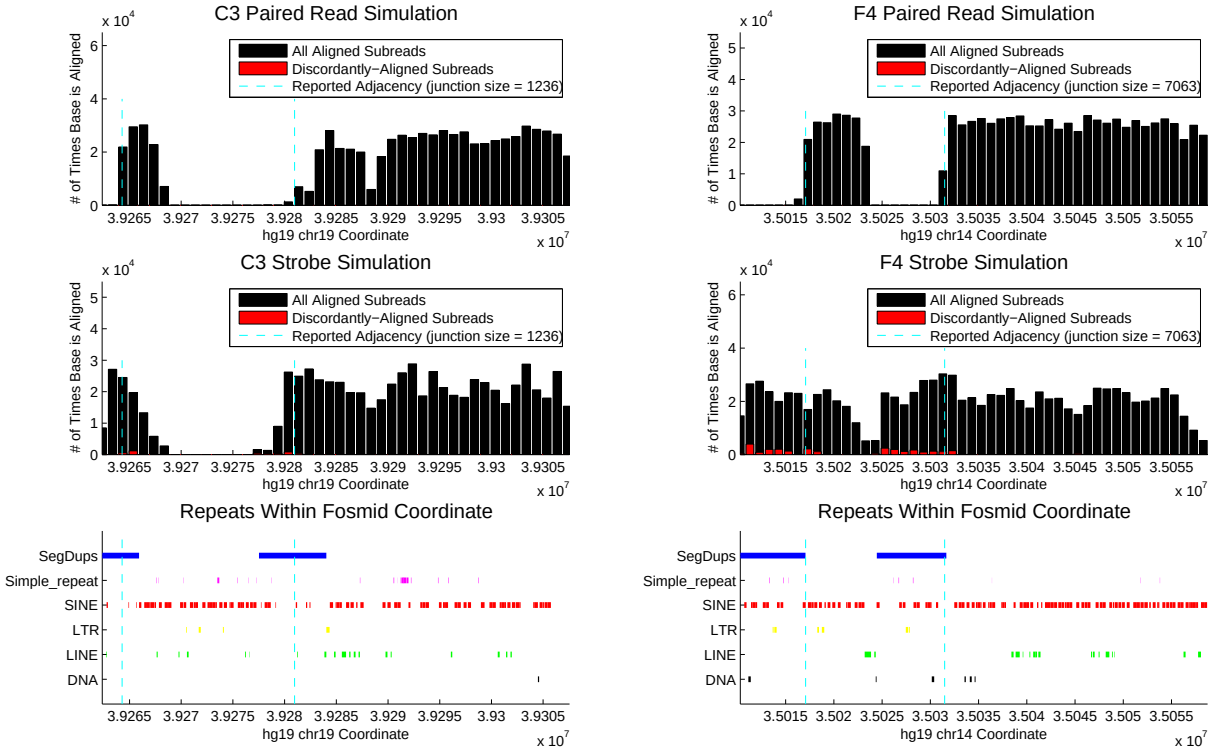

Figure 12: Paired Read and Strobe Simulations for Inversion I1 (C3) and I2 (F4). Pairs could not detect the inversions in simulation. See Figure 11 for plot description.

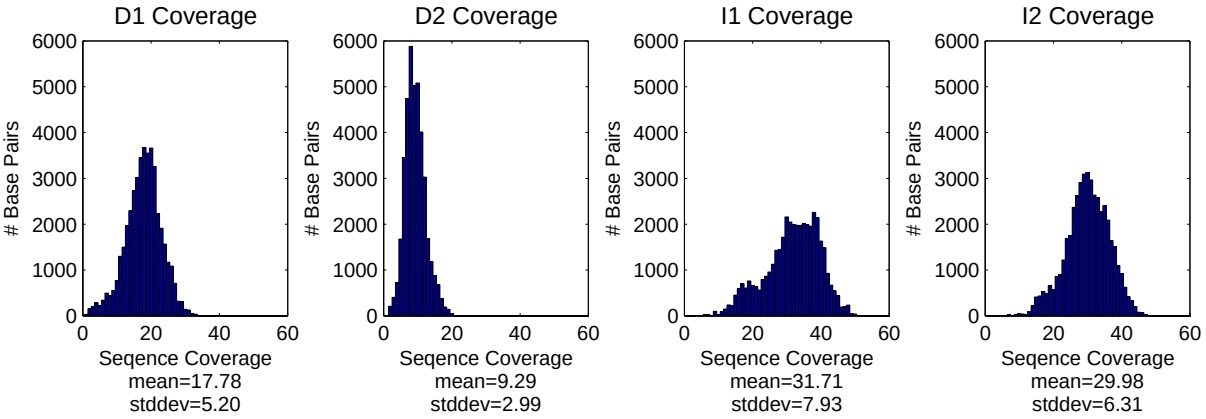

Figure 13: Sequence coverage for the four fosmids.

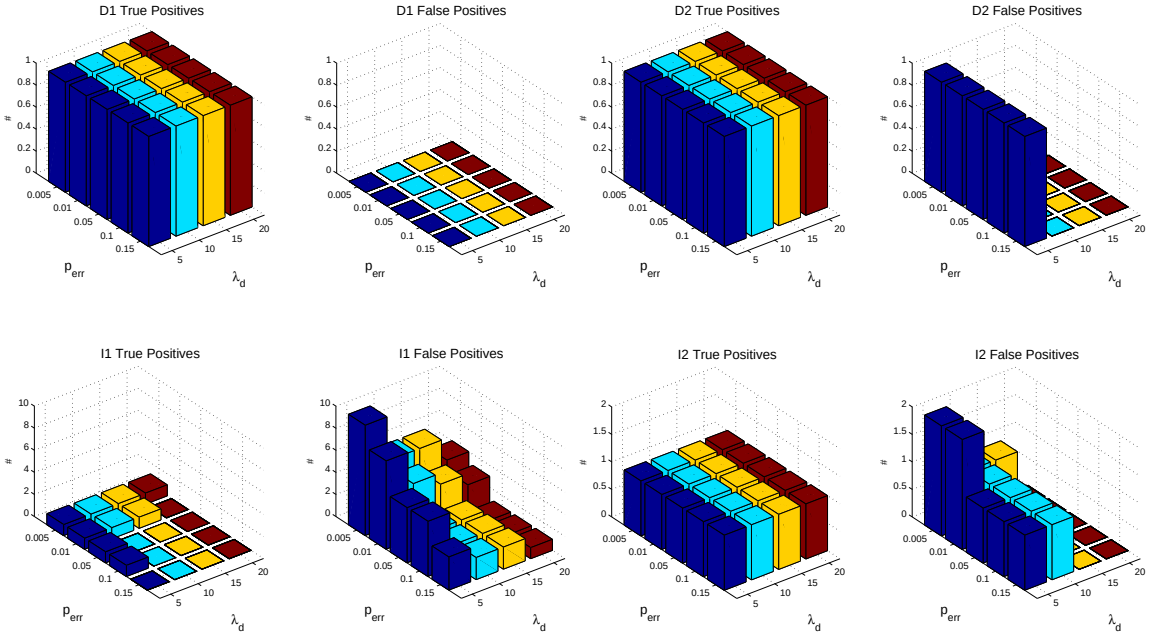

Figure 14: MCMC Results on Sequenced Fosmids. For each fosmid (Top row: D1, D2; Bottom row: I1, I2), the plot on the left indicates the number of true positives recorded and the plot on the right indicates the number of false positives for combinations of $\lambda_d = (5, 10, 15, 20)$ and $p_{err} = (0.005, 0.01, 0.05, 0.1, 0.15)$. The true positive and false positive plots are scaled by the same range on the Z-axis for each fosmid.

2.5 Sequenced CHM1TERT Genome

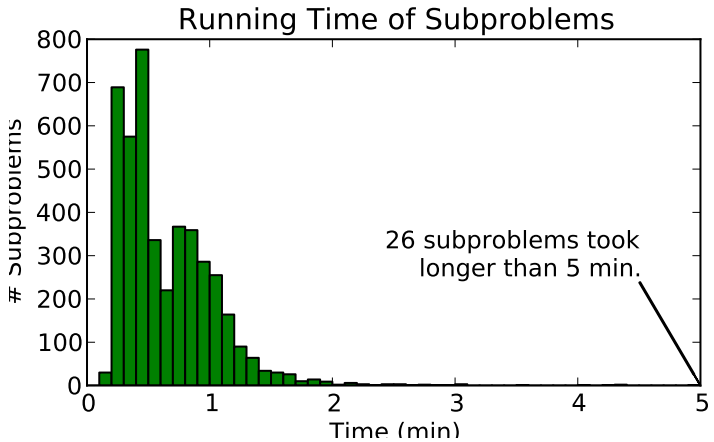

Figure 15: Histogram of times for each sampled run.

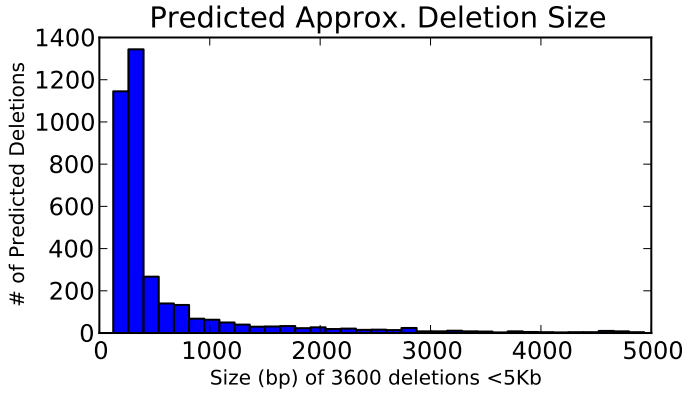

Figure 16: CHM1TERT Structural Variants Predicted by MultiBreak-SV.

2.5.1 Comparing Predictions to an Assembly

We used nucmer to align each chromosome in the NCBI CHM1TERT assembly to hg19. We then identified alignments larger than 7Kb and the top hit when aligning the assembly to the reference *and* the reference to the assembly using `delta-filter -l 7000 -1` provided by the nucmer software distribution. We then reported the coordinates using the `show-coords` program provided by nucmer.

For the high-probability deletions, we tested whether they were *confirmed* by the assembly. For every high-probability deletion with coordinates in the reference, we found the two flanking alignments a_{left} and a_{right} from the assembly that have the closest coordinates to the left and right breakpoints of the predicted deleted region in the reference. If a_{left} is within 100bp of the left predicted breakpoint, a_{right} is within 100bp of the right predicted breakpoint, and the leftmost coordinate of a_{right} in the assembly is larger than the rightmost coordinate of a_{left} by no more than 10bp, then we call the prediction confirmed by the assembly.

For those deletions that were not confirmed by the assembly, we examined whether they could *propose* a deletion in the assembly; that is, the deletion is not represented in the Illumina-based assembly. To do this, we found alignments a_{left} and a_{right} from the assembly that span the left and right breakpoints of the predicted deleted region in the reference. For a predicted

deleted region in the reference with coordinates (x, y), we mapped x to the assembly coordinates using a_{left} and mapped y to the assembly coordinates using a_{right} to produce coordinates (x', y') in the assembly. If the assembly does not reflect the deleted region, then $y - x$ should be about the same as $y' - x'$. Thus, we call the deletion *proposed* if the length $y - x$ in the reference is within 80% (between 80% and 120%) of the length $y' - x'$ in the assembly. The 128 deletions that are proposed in the Illumina assembly (Prob > 0.9; $k = 5$) are shown in Tables 7-9.

References

[1] 1000 Genomes Project Consortium. A map of human genome variation from population-scale sequencing. *Nature*, 467(7319):1061–1073, 2010.

[2] M J Chaisson and Glenn Tesler. Mapping single molecule sequencing reads using basic local alignment with successive refinement (blasr): application and theory. *BMC bioinformatics*, 13(1):238, 2012.

[3] F Hormozdiari et al. Combinatorial algorithms for structural variation detection in high-throughput sequenced genomes. *Genome Res*, 19(7):1270–1278, 2009.

[4] J. M. Kidd et al. A human genome structural variation sequencing resource reveals insights into mutational mechanisms. *Cell*, 143:837–847, 2010.

[5] Novocraft. Novoalign. http://www.novocraft.com/main/index.php.

[6] A R Quinlan et al. Genome-wide mapping and assembly of structural variant breakpoints in the mouse genome. *Genome Res*, 20(5):623–635, 2010.

[7] T Rausch et al. Delly: structural variant discovery by integrated paired-end and split-read analysis. *Bioinformatics*, 28(18):i333–i339, 2012.

[8] A Ritz, A Bashir, and B J Raphael. Structural variation analysis with strobe reads. *Bioinformatics*, 26(10):1291–1298, 2010.

[9] S Sindi et al. A geometric approach for classification and comparison of structural variants. *Bioinformatics*, 25(12):i222–i230, 2009.

[10] S S Sindi et al. An integrative probabilistic model for identification of structural variation in sequencing data. *Genome Biol*, 13(3):R22, 2012.

ClusterID	Probability	# Multi-Breakpoint-Mappings	Predicted Deletion
c214623	1.000000	6	chr12:22129571-22131592
c215458	1.000000	9	chr12:123010130-123010705
c40909	1.000000	9	chr2:240958939-240959289
c142689	1.000000	9	chr6:33093369-33093754
c142687	1.000000	5	chr6:33009528-33009811
c143932	1.000000	6	chr6:158548266-158549415
c214386.4	1.000000	10	chr12:7968504-7978148
c123784.0	1.000000	9	chr5:141455486-141459550
c207558	1.000000	9	chr11:104010469-104010834
c143026	1.000000	12	chr6:57533106-57534178
c161019	1.000000	6	chr7:151223097-151223602
c229757	1.000000	7	chr15:75867300-75867638
c241376	1.000000	8	chr19:57680219-57681186
c143020.2	1.000000	8	chr6:57381814-57382114
c143021.1	1.000000	7	chr6:57449479-57449821
c207291	1.000000	7	chr11:74134201-74145429
c214531	1.000000	5	chr12:14151168-14151682
c243165	1.000000	9	chr21:30263561-30264318
c175248	1.000000	7	chr8:140475258-140475668
c3862	1.000000	9	chr1:152555494-152588089
c142385	1.000000	11	chr6:11903312-11903800
c225684	1.000000	5	chr14:77687204-77702477
c234282	1.000000	8	chr16:71164332-71164738
c73255.3	1.000000	12	chr3:75778031-75787828
c207453	1.000000	10	chr11:89932512-89932883
c173746	1.000000	9	chr8:2215426-2216367
c99646	1.000000	6	chr4:164077149-164077610
c1782	1.000000	8	chr1:72766304-72811873
c243686	1.000000	29	chr22:21574865-25012381
c234027	1.000000	23	chr16:46421134-46428212
c243131	1.000000	5	chr21:24828945-24829598
c98099	1.000000	5	chr4:31492757-31493310
c187968	1.000000	18	chr9:68434017-70835512
c238983	1.000000	10	chr18:27872407-27872915
c225193	1.000000	5	chr14:24486705-24487053
c215299	1.000000	5	chr12:97220952-97221403
c143690	1.000000	11	chr6:129737058-129737414
c228806	1.000000	15	chr15:21933591-21934783
c239203	1.000000	8	chr18:55177456-55178018
c4563	1.000000	6	chr1:226985308-226986122
c243644	1.000000	5	chr22:20658024-20659538
c232865	1.000000	5	chr16:12354864-12355825
c243054	1.000000	5	chr21:9483464-9484661
c40464.1	1.000000	12	chr2:181925818-181933366
c221277	1.000000	10	chr13:57975474-57975882
c221559	1.000000	8	chr13:98721210-98721586
c143015	1.000000	6	chr6:57239071-57239440
c98369	1.000000	7	chr4:58087334-58087758
c98579	1.000000	15	chr4:69790854-69791219
c243052	1.000000	8	chr21:9448717-9449089
c233648	1.000000	6	chr16:32337970-32338470
c40708	1.000000	5	chr2:209451666-209452064

Table 7: CHM1TERT deletions that fill a gap in the Illumina assembly (Prob > 0.9; $k = 5$). Table 1 of 3.

ClusterID	Probability	# Multi-Breakpoint-Mappings	Predicted Deletion
c245147	1.000000	16	chrX:77696433-77697501
c199247	1.000000	9	chr10:104527817-104528521
c206934.5	1.000000	6	chr11:48928940-48934942
c160413	1.000000	8	chr7:90022129-90022508
c1885	1.000000	11	chr1:79392117-79401859
c214675	1.000000	7	chr12:25669994-25670524
c123285	1.000000	10	chr5:89860414-89861070
c242107	1.000000	8	chr20:4166113-4166631
c40912	1.000000	5	chr2:241047820-241048325
c1111	1.000000	7	chr1:28247753-28248121
c214573	1.000000	6	chr12:18317393-18317948
c3209	1.000000	7	chr1:145255746-145256019
c158930	1.000000	8	chr7:14449027-14449435
c123289	1.000000	8	chr5:90574678-90575002
c38987.1	1.000000	5	chr2:87623152-87626040
c2606	1.000000	5	chr1:142812683-142813916
c188662	1.000000	7	chr9:115937074-115937429
c197995	1.000000	6	chr10:18849888-18850416
c233177	1.000000	6	chr16:19506314-19506657
c243069	1.000000	8	chr21:9841703-9842091
c98801	1.000000	6	chr4:81991499-81991863
c237219	1.000000	5	chr17:43257773-43259144
c206445	1.000000	7	chr11:7716867-7717264
c206712	1.000000	8	chr11:32286610-32286993
c174993	1.000000	5	chr8:101904323-101904746
c72160	1.000000	6	chr3:2495660-2496474
c73151	1.000000	7	chr3:70415101-70415468
c98995	1.000000	10	chr4:97572089-97572449
c37837	1.000000	8	chr2:23803024-23803792
c143146.0	1.000000	7	chr6:74156152-74156783
c73756	1.000000	13	chr3:124890024-124891136
c198395	1.000000	6	chr10:47036090-47036344
c206915.1	1.000000	6	chr11:48880687-48882114
c188781	1.000000	5	chr9:139474544-139475441
c245793	1.000000	11	chrX:143206583-143206936
c242232	1.000000	5	chr20:18757019-18757495
c221337	1.000000	7	chr13:63643097-63643518
c175195.1	1.000000	8	chr8:132228122-132228613
c98462	1.000000	5	chr4:64194129-64194948
c221547.0	1.000000	7	chr13:96633419-96633849
c240938	1.000000	8	chr19:43167386-43167889
c39974	1.000000	8	chr2:133115523-133115842
c242290	1.000000	6	chr20:32043888-32045286
c143773.1	1.000000	6	chr6:139602655-139603007
c3765	1.000000	7	chr1:148854410-148855886
c40468	1.000000	8	chr2:182565234-182565598
c233577.0	1.000000	5	chr16:32109626-32109978
c243222	1.000000	8	chr21:41812220-41812739
c143023	1.000000	7	chr6:57422951-57429307
c188562.0	1.000000	7	chr9:101028705-101029115
c73746	1.000000	5	chr3:124086096-124087068

Table 8: CHM1TERT deletions that fill a gap in the Illumina assembly (Prob > 0.9; $k = 5$). Table 2 of 3.

ClusterID	Probability	# Multi-Breakpoint-Mappings	Predicted Deletion
c906	1.000000	5	chr1:19151766-19152500
c143161	1.000000	6	chr6:74865433-74866599
c237193	1.000000	5	chr17:40489893-40490604
c214389	1.000000	5	chr12:8132224-8133071
c98793	1.000000	10	chr4:81302519-81302937
c98545	1.000000	7	chr4:69307973-69308858
c215472	1.000000	7	chr12:125334367-125334740
c175011	1.000000	8	chr8:103940550-103940994
c4661	1.000000	7	chr1:241360528-241360955
c185724	1.000000	5	chr9:33423359-33424931
c72412	1.000000	5	chr3:18868173-18868578
c3154.1	1.000000	10	chr1:145026722-145027817
c40034.1	1.000000	7	chr2:138245282-138245708
c221260	1.000000	8	chr13:55634060-55634448
c236646	1.000000	8	chr17:18637452-18637888
c240475	1.000000	5	chr19:4885192-4886125
c143024	1.000000	11	chr6:57478147-57478871
c98740	1.000000	8	chr4:76948610-76949116
c38798.0	1.000000	5	chr2:83375213-83375687
c2084.1	1.000000	7	chr1:91914106-91914596
c40914	1.000000	7	chr2:241241730-241242599
c159387	1.000000	5	chr7:37998504-37998831
c1757	1.000000	5	chr1:70946193-70946600
c214691	1.000000	10	chr12:26937673-26938325
c99814	1.000000	5	chr4:187795674-187796042

Table 9: CHM1TERT deletions that fill a gap in the Illumina assembly (Prob > 0.9; $k = 5$). Table 3 of 3.