# TIGRA: A targeted iterative graph routing assembler for breakpoint assembly

Ken Chen,[1,2,6,7] Lei Chen,[3,6] Xian Fan,[1,2,6] John Wallis,[3] Li Ding,[3,4] and George Weinstock[3,5]

[1]*Department of Bioinformatics and Computational Biology, The University of Texas MD Anderson Cancer Center, Houston, Texas 77030, USA;* [2]*Department of Computer Science, Rice University, Houston, Texas 77005, USA;* [3]*The Genome Institute, Washington University School of Medicine, St. Louis, Missouri 63110, USA;* [4]*Department of Medicine, Washington University School of Medicine, St. Louis, Missouri 63110, USA;* [5]*Department of Genetics, Washington University School of Medicine, St. Louis, Missouri 63110, USA*

**Recent progress in next-generation sequencing has greatly facilitated our study of genomic structural variation. Unlike single nucleotide variants and small indels, many structural variants have not been completely characterized at nucleotide resolution. Deriving the complete sequences underlying such breakpoints is crucial for not only accurate discovery, but also for the functional characterization of altered alleles. However, our current ability to determine such breakpoint sequences is limited because of challenges in aligning and assembling short reads. To address this issue, we developed a targeted iterative graph routing assembler, TIGRA, which implements a set of novel data analysis routines to achieve effective breakpoint assembly from next-generation sequencing data. In our assessment using data from the 1000 Genomes Project, TIGRA was able to accurately assemble the majority of deletion and mobile element insertion breakpoints, with a substantively better success rate and accuracy than other algorithms. TIGRA has been applied in the 1000 Genomes Project and other projects and is freely available for academic use.**

**[Supplemental material is available for this article.]**

Genomic structural variations (SVs), such as insertions, deletions, duplications, inversions, and translocations, are important genetic drivers for human diseases. Identifying these SVs, however, has been difficult due to the complexity of the human genome and the limitation of current technologies. Progress in next-generation sequencing (NGS), such as whole genome sequencing, has greatly facilitated our study of SVs. Many SVs have since been identified through aligning paired-end reads or long reads for human reference assembly. Several popularly used SV prediction algorithms (Chen et al. 2009; Hormozdiari et al. 2009; Ye et al. 2009; Rausch et al. 2012) have predicted 10 times more SVs in an individual genome than in previous studies (Korbel et al. 2007; Kidd et al. 2008). However, due to limited physical coverage of many NGS libraries and the repetitiveness of SV regions, the alignment-based approaches often suffer from relatively high false discovery rates (FDRs) and lack of breakpoint precision. Some complex SVs, such as those in genomically instable cancer genomes, are particularly difficult to resolve precisely because of flanking repeats and sequence divergence from the reference assembly (Stephens et al. 2009).

Another way to identify SVs is through sequence assembly, which pieces together redundant short reads into longer sequence contigs. This approach is not biased by the reference and is generally more specific than alignment-based approaches because more reads are involved in determining a breakpoint. However, this approach is limited by our ability to assemble large and complex genomes from short reads and generally demands more coverage. Although progress has been made in assembling a complete human genome from short NGS reads using affordable computational resources (Zerbino and Birney 2008; Li et al. 2010; Simpson and Durbin 2010; Iqbal et al. 2012), assembler development is still at the stage of increasing assembly contiguity and reducing computational cost, instead of comprehensively and accurately representing all DNA sequences in the input data. Consequently, low-coverage alternative alleles in diploid or polyploid genomes tend to be ignored (Alkan et al. 2011). Despite recent efforts that have enriched the representation of alternative SNVs and indel alleles (Iqbal et al. 2012), breakpoint sequences underlying large SVs are still insufficiently represented in the assembly results (Alkan et al. 2011; Mills et al. 2011).

There is a pressing demand to obtain high-quality sequences underlying the SV breakpoints (breakpoint sequences) for the purpose of achieving accurate characterization and deriving biological understanding. In our early efforts, we found that applying TIGRA to predicted breakpoints can reduce the FDR to as low as 2%–3% for germline SVs (Chen et al. 2009; Ding et al. 2010) and 6% for somatic SVs (Chen et al. 2009). Similarly, low FDRs (3%–5%) have been observed in independent research (Zhu et al. 2012). Achieving a low FDR is valuable in large-scale studies because it alleviates the need to perform orthogonal experimental validation, which can be costly and time-consuming. Existing experimental techniques such as polymerase chain reaction, array hybridization, fluorescence in situ hybridization, and capillary-based sequencing have their own limitations in terms of capability and precision (Korbel et al. 2007; Kidd et al. 2008; Conrad et al. 2009, 2010). Therefore, it is highly desirable to achieve low FDRs when using

**[6]These authors contributed equally to this work.**
**[7]Corresponding author**
**E-mail kchen3@mdanderson.org**
Article published online before print. Article, supplemental material, and publication date are at http://www.genome.org/cgi/doi/10.1101/gr.162883.113.




24:310–317 Published by Cold Spring Harbor Laboratory Press; ISSN 1088-9051/14; www.genome.org

breakpoint assembly, which has been demonstrated by the 1000 Genomes Project Consortium (Mills et al. 2011).

Obtaining high-quality breakpoint sequences is also essential for understanding the molecular processes and mechanisms that produced the germline (Korbel et al. 2007; Gu et al. 2008; Kidd et al. 2008; Conrad et al. 2010; Lam et al. 2010) and/or the somatic SVs (Stephens et al. 2009; Malhotra et al. 2013). For example, the presence of microhomology and nontemplate sequences at the breakpoints is a signature for DNA double-strand breaking followed by microhomology-mediated repair. Precise characterization at nucleotide resolution provides resources for genomic evolutionary studies and sheds light on the etiology of genetic diseases (Yang et al. 2013).

Although it seems conceptually straightforward to perform local targeted assembly at any given location in an alignment assembly (e.g., BAM), it is actually nontrivial to establish a good practice. One major concern is the false negatives, i.e., an assembler fails to produce a valid breakpoint sequence, even with sufficient coverage. This concern is credible because almost all existing assemblers are developed to perform whole haploid genome assembly and are optimized to produce long contigs at the cost of ignoring local complex structures. In addition, they are not designed to process population sequencing data that contain multiple alleles, such as those collected from heterogeneous cancer tissues.

Since 2008, we have worked to establish a good practice for assembling breakpoints. After testing the available assemblers, we developed a new approach that uses a targeted iterative graph routing assembler (TIGRA), which we optimized to assemble breakpoints from population sequencing data with minimized false negatives and improved sequence accuracy. We have applied various versions of TIGRA to data compiled in the 1000 Genomes Project and other large-scale cancer genome sequencing projects and have demonstrated its efficacy in confirming alignment-based SV calls, pinpointing precise breakpoints, estimating SV genotypes and allele abundances, and facilitating breakpoint analysis for mechanistic understanding (The 1000 Genomes Project Consortium 2010, 2012; Ding et al. 2010; Mills et al. 2011). To our best knowledge, TIGRA is the only tool that has been specifically developed to assemble SV breakpoints and has been thoroughly examined in population-scale genome sequencing projects. In this manuscript, we describe the rationale and details behind our design and demonstrate the efficacy of TIGRA using publicly available data. To demonstrate the advantages of TIGRA, we compare it with alternative approaches that employ other widely used assemblers, such as Velvet (Zerbino and Birney 2008), SGA (Simpson and Durbin 2010), Phrap (http://www.phrap.org), and SPAdes (Bankevich et al. 2012).

We selected these assemblers to represent a wide spectrum of assembly algorithms. Both Velvet and SPAdes are based on de Bruijn graphs. Although Velvet was among the first published short read assemblers and has been widely used, SPAdes was developed more recently and includes a set of theoretical and algorithmic advances that target single-cell data and metagenomic data. SGA is the first string graph-based assembler that has achieved considerable advances in reducing computational expenses and has been applied to breakpoint assembly (Malhotra et al. 2013). Phrap is a classic assembler that implements the overlap-layout-consensus strategy. As reads become longer, such a strategy can potentially be revived for small-scale NGS assembly.

## Results

### Overview of TIGRA

The TIGRA program consists of two major steps: (1) read extraction, and (2) assembly (Fig. 1; Methods). The input is a list of putative breakpoints and a set of binary sequence alignment/map formatted (BAM) files that contain the sequence reads aligned to the reference (Li et al. 2009).

An input breakpoint can be specified with different levels of detail. At a minimum, it must have the genomic coordinates of the predicted breakpoint. In addition, it can include the type of SV (e.g., deletion or insertion) that produced the breakpoint, which is inferred from discordant paired-end alignment or split-read alignment (Chen et al. 2009; Ye et al. 2009; Rausch et al. 2012).

To increase the chance of successful assembly, TIGRA tries to include all reads that are likely associated with the breakpoint, as long as they have at least one end or a subsegment that is confidently mapped (e.g., mapping quality greater than 20) around the breakpoint. The other end (mate) could be either unmapped, soft-clipped, or mapped to a distal locus (e.g., a different chromosome). When the type of SV is known, TIGRA extracts reads selectively to reduce the representation of the reference allele. For example, for a deletion breakpoint, only reads near the start and the end coordinates are extracted; whereas reads within the deleted region are not extracted. When genotypes are known, TIGRA can selectively obtain reads from the individuals that contain the breakpoint.

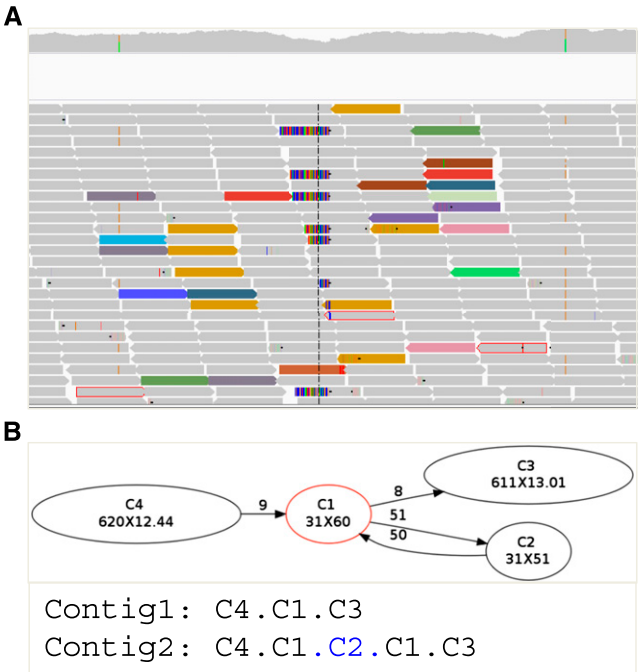


**Figure 1.** Schematic view of TIGRA. (*A*) Reads (arrow-shaped boxes) at a breakpoint (vertical dashed line in the center), including those normally mapped (gray), mate-unmapped (gray with red outline), soft-clipped (multicolored), and interchromosomally mapped (colored) are extracted from BAM files and sent to the assembly algorithm. (*B*) A de Bruijn graph is constructed using an iterative multiple-k-mer assembly algorithm. A contig (oval indexed node) with a specified length and average k-mer coverage (*x*) is connected to other contigs if it overlaps other contigs by k-1 bp (edge) in a particular orientation (arrow), and is of a particular coverage (weight). In this example, a mobile element insertion (of C2) with homology regions (C1) is successfully assembled. Two contig strings are decoded from the graph by TIGRA, representing two alternative alleles.

Once reads are extracted, TIGRA attempts to assemble them into longer contigs using a de Bruijn graph-based approach. To increase the chance of successful assembly, we adopted several innovations that differ from standard approaches such as Velvet. First, rather than using a single k-mer, TIGRA allows for the iterative use of multiple k-mers to increase the chance of assembling low-coverage alleles from short reads. This feature is particularly relevant in our setting of population sequencing because the alternative alleles, such as those in a subpopulation of tumor cells, can have substantially lower coverage than those of the dominant alleles. The coverage of alternative alleles can be further lowered in the BAM files due to alignment biases against nonreference sequences. Second, TIGRA records bubbles (alternative paths) in the contig graphs and outputs them to represent the alternative alleles with common flanking sequences. Third, it uses reads (instead of k-mers) to resolve repeats in the graph. This feature makes it possible to distinguish repeats that are longer than the k-mer length and at lengths up to the read length. Finally, TIGRA takes various measures to enhance the representation of the alternative structures. For example, TIGRA uses singleton k-mers (rather than ignoring them) if they facilitate further expansion of contigs from the previous iteration. More details are described in Methods.

Since we developed and applied TIGRA to the 1000 Genomes Project data in 2010 (Mills et al. 2011), similar techniques have been adopted in different contexts (e.g., single-cell genome assembly) (Bankevich et al. 2012; Peng et al. 2012; Iqbal et al. 2013), which have further supported the validity of our design.

## Deletion breakpoint assembly

To examine TIGRA, we curated two sets of previously sequenced deletion breakpoints as reference standards (Conrad et al. 2010; Kidd et al. 2010).

The first set contains 245 deletion breakpoints (Supplemental Table 1) that were previously characterized at breakpoint resolution (Conrad et al. 2010) and that are present (80% reciprocal overlap) in at least one of the 45 CEU samples in The 1000 Genomes pilot project (Mills et al. 2011). We downloaded the corresponding Burrows-Wheeler (Li and Durbin 2009) aligned BAM files from the 1000 Genomes Project, which consisted of paired-end short (35–50 bp) Illumina reads obtained from low-depth (2–5×) whole genome sequencing. To demonstrate the efficacy of TIGRA's assembly algorithm, the second step in the overall process, we examined the same set of reads assembled by TIGRA using Velvet, SGA, Phrap, and SPAdes (Methods).

In our first experiment, we used the available genotype data to guide the read extraction. At each breakpoint, we obtained a set of reads from the variant-containing samples and produced multiple sets of contigs using different assemblers or parameters (Table 1A). We aligned the assembled contigs to the reference sequence using cross_match (http://www.phrap.org/phredphrapconsed.html) and considered a breakpoint to be correctly assembled if one of its derivative contigs aligned in a configuration compatible with the ground truth, i.e., indicating a deletion of similar size (<10% difference) with similar breakpoint coordinates (<5 bp apart). Of the 245 breakpoints, TIGRA correctly assembled 83.7%, followed by SPAdes (69.0%), SGA (56.7%), Velvet (k = 31; 55.10%), and Phrap (35.9%). We followed a recently published procedure (Malhotra et al. 2013), which decoded all paths in the string graph (called SGA.walk), and merged it with standard string graph assembler (SGA) results (called SGA.expand). This improved the overall SGA result to 62.3%. The use of multiple k-mers (k = 15, 25) in TIGRA outperformed the use of a single k-mer (k = 25) by 11 (4.5%) breakpoints. Overall, TIGRA assembled 15% more breakpoints than any other assembler evaluated in this experiment.

**Table 1. Comparison of deletion breakpoint assembly using low-coverage population sequencing data from the 1000 Genomes Project based on a set of 245 known breakpoints in 45 CEU pilot samples (A) and a set of 562 known breakpoints in eight phase 3 samples (B)**

A

| | Genotype ignored | | Genotype aware | |
|---|---|---|---|---|
| Method | Number | Percentage | Number | Percentage |
| Conrad | 245 | 100.00% | 245 | 100.00% |
| Tigra-0.3.7 | 173 | 70.61% | 205 | 83.67% |
| Tigra-0.3.7, k = 25 | 173 | 70.61% | 194 | 79.18% |
| Velvet-1.2.09, k = 31 | 50 | 20.41% | 135 | 55.10% |
| SGA-0.9.17 | 130 | 53.06% | 138 | 56.33% |
| SGA-0.9.17 walks | 149 | 60.82% | 133 | 54.29% |
| SGA-0.9.17 expand | 156 | 63.67% | 152 | 62.04% |
| Phrap-1.080721 | 89 | 36.33% | 88 | 35.92% |
| SPAdes-2.5.0 | 143 | 58.37% | 169 | 68.98% |

B

| | Genotype ignored | | Genotype aware | |
|---|---|---|---|---|
| Method | Number | Percentage | Number | Percentage |
| Kidd | 562 | 100.00% | 562 | 100.00% |
| Tigra-sv-0.3.7, k = 15,25,35 | 317 | 56.41% | 276 | 49.11% |
| Velvet-1.2.09, k = 31 | 111 | 19.75% | 196 | 34.88% |
| SGA-0.9.17 | 245 | 43.59% | 90 | 16.01% |
| SGA-0.9.17 walks | 278 | 49.47% | 21 | 3.74% |
| SGA-0.9.17 expand | 307 | 54.63% | 95 | 16.90% |
| Phrap-1.080721 | 158 | 28.11% | 240 | 42.70% |
| SPAdes-2.5.0 | 201 | 35.77% | NA | NA |
| SPAdes-2.5.0 sc | 179 | 31.85% | 270 | 48.04% |

In our second experiment, we ignored the known genotypes and extracted reads from all 45 samples. This application is analogous to assembling subclonal breakpoints from population sequencing of clonally heterogeneous cancer or microbial samples that contain mixtures of low-abundance alleles. As expected, fewer breakpoints were successfully assembled. Still, TIGRA succeeded in 70.6% of the 245 cases (Table 1A) in contrast to the success of Velvet (20.4%), SGA.expand (63.7%), Phrap (36.3%), and SPAdes (58.4%). The expanded SGA process scored well in this experiment. However, it produced almost twice as many contigs as did TIGRA, which increases the load for downstream analysis. The success rate was clearly affected by variant allele frequency in the population (Fig. 2). TIGRA performed substantially better than the other assemblers at assembling low-frequency (less than 0.15) alleles, as expected from its use of multiple k-mers and more careful treatment of data.

In both experiments, results can be improved by adjusting TIGRA read extraction parameters (Supplemental Table 2; Supplemental Notes).

The second reference standard set contains 562 sequenced deletion breakpoints in eight cell-line samples obtained using long insert (~48 kb) fosmid end sequence profiling (Supplemental Table 3; Kidd et al. 2010). We thought that this set of breakpoints might provide a more comprehensive assessment because it includes breakpoints with longer (>100 bp) homology, resulting from

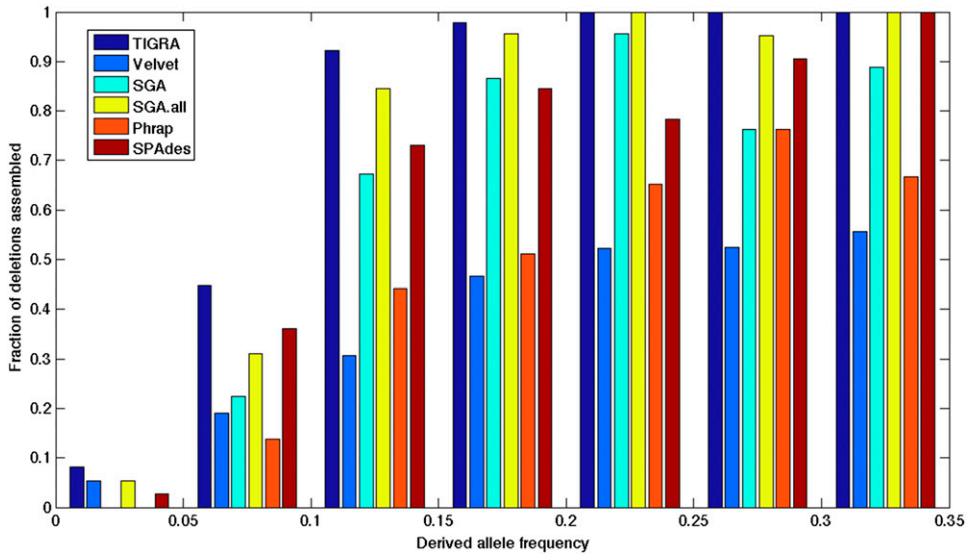


**Figure 2.** Comparison of assembly success rate at various allele frequencies in 45 CEU samples. Six assemblers are plotted: TIGRA (purple), Velvet (blue), SGA (cyan), SGA.all (yellow), Phrap (red), and SPAdes (brown). Allele frequencies (*x*-axis) are derived from the deletion genotypes released by The 1000 Genomes Project Consortium, and the fraction of success (*y*-axis) is estimated from 245 control deletion sites.

nonallelic homologous recombination. We downloaded the BWA aligned BAM files of these eight samples from the 1000 Genomes phase 3 project, which consisted of paired-end 100-bp Illumina reads obtained from low-depth (2–7×) whole genome sequencing. Similar to the Conrad set, we performed the following two experiments.

In the first experiment, we extracted 562 sets of reads independently from the corresponding BAM files using TIGRA and then assembled them using each of the five assemblers (genotype-aware) (Table 1B). TIGRA was able to correctly assemble 276 (49%) breakpoints, outperforming the other assemblers. The low coverage (three- to sevenfold haploid) made it challenging for SGA and Velvet. SPAdes performed nearly as well as TIGRA (in its single-cell mode but not in the whole genome mode). However, it was considerably slower (>100-fold) than TIGRA. As expected, the short insert size (ranging from 126 to 449 bp) and the short read length in the NGS data made it difficult to resolve long repeats. In the results from both TIGRA and SPAdes, we observed a monotonic reduction of correctly assembled breakpoints as the size of the breakpoint homology increased (Supplemental Fig. 1).

In the second experiment, we pulled reads from all eight samples at each breakpoint (genotype-ignored) (Table 1B). TIGRA correctly assembled 317 (56.4%) breakpoints, which was the best among all the assemblers. Interesting, 41 more breakpoints were assembled by TIGRA in this experiment than in the first experiment, which indicates that those breakpoints are present in more samples than originally reported.

## Mobile element insertion breakpoint assembly

Unlike deletions, insertions are generally difficult to assemble because reads from inserted sequences may not be as clearly associated with the insertion breakpoint in the BAM files. Assembling the mobile element insertions (MEIs) is even more difficult due to their nonunique sequence content and their relatively large sizes. Major MEI subtypes, such as *Alu,* LINE, and SVA, have many subfamilies that are distinct in their lineages yet are highly homologous. For example, among *Alu* subfamilies, *Alu*Ya4 differs from *Alu*Ya5 by only one substitution and from ancestral *Alu*Y by five substitutions. Distinguishing these subfamilies thus requires the accurate assembly of the inserted elements.

To examine how well TIGRA assembles MEIs, we applied it to a set of 442 experimentally validated MEI breakpoints in NA12878, a cell-line genome (Mills et al. 2011; Stewart et al. 2011). The BAM file, which we downloaded from the 1000 Genomes Project, consists of short insert (~400 bp) paired-end Illumina HiSeq reads (101 bp) with 60-fold haploid sequence coverage. Again, to compare the assembly algorithms, we assembled the same sets of reads extracted by TIGRA using Velvet, SGA, Phrap, and SPAdes.

All five assemblers produced results at each of the 442 breakpoints. The lengths of the resulting contigs varied widely across the various approaches: Phrap produced the longest contig (~1355 bp), followed by SPAdes (~1288 bp), Velvet (k = 31; ~858 bp), TIGRA (~805.5 bp), and SGA (~397 bp).

To examine the accuracy of the assembled sequences, we aligned the obtained MEI contigs against 52 MEI consensus sequences derived from Repbase 14.02 (Stewart et al. 2011) (http://www.girinst.org/repbase/index.html) using the BLAST-like alignment tool (Kent 2002). Based on the alignment, we found that contigs assembled from TIGRA contained the longest MEI sequences (average 280 bp) (Supplemental Table 4). This result indicates that the majority of these insertions (predominately *Alu*) were assembled near their full length (~300 bp). Manual inspection (Methods) of TIGRA contigs confirmed that many *Alu* elements were completely assembled not only with polyA tails but also with flanking sequences attached on both ends. Such results allow TIGRA to precisely pinpoint the breakpoints as well as accurately classify the lineages of the inserted elements. The average length of MEIs in Phrap contigs was 276 bp, which was almost as long as those obtained by TIGRA. In Velvet contigs, the average MEI sequence length was 269.6 bp, despite the longer overall contig lengths. Even shorter were those in the SPAdes contigs (260 bp) and SGA contigs (235.6 bp). Similar to the process for deletions, we expanded the SGA contig collection by including all paths in

the SGA graph (Methods). This increased the average length of its MEI-containing contigs to 1514 bp, at the cost of producing many (3.6 times) more contigs than TIGRA. The lengths of the MEIs in these contigs also increased to 266.4 bp, but were still shorter than those assembled by TIGRA, Phrap, or Velvet.

We classified each MEI breakpoint to a subtype (*Alu*, L1, and SVA) and a specific subfamily based on the best alignment of its assembly with the 52 Repbase sequences (Supplemental Table 4). Subtype classification based on TIGRA achieved ~96% overall concordance with those published by the 1000 Genomes Project Consortium (Stewart et al. 2011). Among the 442 breakpoints, TIGRA found 393 (88%) *Alu*, 44 (10%) L1, and 5 (1%) SVA. The subfamilies most frequently found were *Alu*Ya5 (136/442; 30.8%) and *Alu*Yb8 (96/442; 21.7%). These results were highly consistent with the findings reported by Stewart et al. (2011), which were derived from independent alignment-based approaches that utilized different technologies.

A subset of 158 MEIs has independently characterized subfamilies in dbRIP (Wang et al. 2006). This provided us with an opportunity to further examine the accuracy of the assembled insertion sequences based on the subfamily classification. Comparing our subfamily lineages with those in dbRIP (Supplemental Table 4), we found a concordance rate of 69.6% under stringent criteria that require an exact lineage match (Table 2). This percentage is nearly identical to that reported previously (70%), which was derived from assembling longer 454 reads (Stewart et al. 2011). In comparison, the SGA expanded approach achieved a concordance rate of 62.0%; whereas the concordance rates of Velvet (k = 31; 44.3%), and SPAdes (41.1%) indicated worse accuracy. Interestingly, Phrap's concordance rate (68.35%) was nearly as accurate as TIGRA's. The majority of discordant classifications between TIGRA and dbRIP (48 total) are among closely related lineages: 17 were among the *Alu*Ya lineage (e.g., between Ya4 and Ya5), 7 among the *Alu*Yb lineage (e.g., between Yb7 and Yb8), and 15 were between ancestral *Alu*Y and more recent lineages (e.g., *Alu*Ye5).

## Discussion

Our comparative analyses demonstrated clear advances that TIGRA has achieved in assembling diversified classes of breakpoint sequences from NGS data. To our best knowledge, TIGRA is the only algorithm that has been specifically developed to perform large-scale breakpoint assembly. We emphasize that existing assemblers such as Velvet, SGA, and SPAdes are generic purpose assemblers that are not specifically designed to assemble breakpoints. Therefore, these comparisons have demonstrated the differences among only the assembly algorithms, instead of among the entire breakpoint assembly process. The specific ways that TIGRA extracts reads are also critical to the overall success. Taken together, TIGRA represents the current best practice for performing SV breakpoint assembly from NGS data.

**Table 2. Comparison of assembler accuracy based on mobile element subfamily classification of 158 mobile element insertion breakpoints in NA12878**

| Method | Matches dbRIP | Percentage |
|---|---|---|
| Tigra | 110 | 69.62% |
| Velvet, k=31 | 70 | 44.30% |
| SGA-0.9.17 | 87 | 55.06% |
| SGA-0.9.17.walk | 98 | 62.03% |
| Phrap-1.080721 | 108 | 68.35% |
| SPAdes-2.5.0 | 65 | 41.14% |

TIGRA is computationally efficient because it typically involves only thousands or tens of thousands of reads per breakpoint, which requires negligible memory. It is efficiently implemented in C++ using SAMtools C libraries to perform fast extraction of individual reads from BAM files. We have tested it on both personal computers and high-performance clusters. For example, it took <10 h using 24 Linux 64-bit CPUs to assemble all (over 20,000) predicted deletions in the 1000 Genomes phase 1 project from more than 1000 low-depth (~5×) BAM files.

The success of TIGRA depends on whether it can include reads spanning breakpoints. Therefore, it is important to provide input that has nucleotide or near nucleotide-level precision (smaller than a few hundred base pairs). Breakpoints determined from discordant paired-end alignments or split-read alignments, and by predictors such as BreakDancer, Delly (Rausch et al. 2012), GenomeSTRiP (Handsaker et al. 2011), and Pindel (Ye et al. 2009), are well suited as input to TIGRA. In the 1000 Genomes Project, the success rates in this category ranged between 50%–80% for the low-coverage (2–7×) short (100–500 bp) insert data. Breakpoints determined based on only the read depth such as CNVnator (Abyzov et al. 2011) and RDXplorer (Yoon et al. 2009) had substantially lower success rates (<20%). Breakpoints determined from the long (>1.5 kbp) insert data may be similarly limited, but more data are required to estimate the extent of this limitation. In addition, breakpoints in repetitive regions are demonstratively more difficult to assemble than those in unique regions due to the limitation of short reads.

In our previous efforts (Mills et al. 2011), we primarily applied TIGRA to assemble deletions and tandem duplications. Assembling MEIs represents a recent upgrade, which we have demonstrated in this manuscript. We have also programmed TIGRA to assemble other types of breakpoints, such as reciprocal translocations and inversions. Those types of breakpoint assemblies have been shown to be useful, but have not been examined as thoroughly due to a lack of ground truth data (Ding et al. 2010; Welch et al. 2011). As more data become available, it is likely that we will need to make additional improvements to the assembly procedures. In addition to simple breakpoints, TIGRA may prove valuable in constructing complex alleles in cancer genomes, which may contain clusters of SV breakpoints and other types of mutations in near distances (Stephens et al. 2009). Our existing implementation allows de novo assembly of any breakpoints, including those that cannot be easily characterized by canonical SV classes. In addition to genomic breakpoints, we have successfully employed TIGRA to assemble novel splice junctions in RNA-seq data and have shown its usefulness in improving the accuracy of gene fusion detection (Chen et al. 2012).

## Methods

### Read extraction

Given a breakpoint, TIGRA obtains reads from a set of binary sequence alignment/map formatted (BAM) files that contain the aligned sequence reads relative to the reference (Li et al. 2009). For a breakpoint produced by an intra-chromosomal SV, such as a deletion, tandem duplication, insertion, or inversion in the genomic interval $[a, b]$, TIGRA obtains reads that map to $[a\text{-}w, a+\theta]$ and $[b\text{-}\theta, b+w]$. We chose the default value of $w$ (500 bp) such that the assembled contig sequences are long enough to unambiguously map the variant. The inclusion of reads that did not originate from the

variant allele increases the complexity and degrades the assembly of variant alleles. Therefore, $\theta$ should be chosen so as to maximally include reads that are unique to the variant allele, especially those that span the breakpoints, while maximally excluding those that are unique to the reference allele. In practice, $\theta$ can be determined based on the confidence intervals of the predicted breakpoints, which are usually specified from the front-end prediction algorithms. Alternatively, it can be set to three times the standard deviation insert size, which typically ranges from 50 to 100 bp for Illumina paired-end libraries. In our practice, the assembly results are sensitive to $\theta$ but not as sensitive to $w$. For a breakpoint produced by a putative inter-chromosomal translocation ($c_1$, $a$), ($c_2$, $b$), where $c_1$ and $c_2$ represent two different chromosomes, we considered four possible fusion configurations and accordingly extracted four different sets of reads (Supplemental Fig. 2).

We extracted all the reads mapped within the above window for assembly. We included their mates (in paired-end sequencing), regardless of their mapping status and location. This could be an expensive operation for certain classes of breakpoints. For example, the inserted element of an MEI may have multiple homologs on the reference, causing the MEI-containing mates to be mapped incorrectly to distal loci. Extracting mates that are scattered in the genome is expensive, even with the random accessibility of BAM files. Nonetheless, the SAMtools C API that TIGRA utilizes has maximally alleviated this issue.

### The iterative graph routing assembler (TIGRA)

We considered the assembly problem as a graph routing (path selection) problem on the sequence graphs derived from the de Bruijn graphic representations of sequencing reads. The main algorithm and implementation are similar in principle to other de Bruijn graph-based assemblers (Pevzner et al. 2001; Zerbino and Birney 2008; Chaisson et al. 2009; Li et al. 2010; Iqbal et al. 2012). Briefly, the de Bruijn graph records k-mers (sequence of length k) in the reads as nodes, and the k-1 bp overlaps between them as edges. The assembly process can be viewed as navigating through this graph, finding a path or paths that represent the target genome sequence(s). During this multistep process, the original de Bruijn graph nodes, if unambiguously connected, are merged to form super nodes representing longer sequences.

### Multiple k-mers and iterative assembly

The k-mer size is one of the most important parameters for a de Bruijn graph-based assembler. The length of the k-mer represents a tradeoff between complexity and redundancy (coverage). For two reads to be assembled together, they must have an exact overlap of k-1 bp or more; thus, shorter k-mers do not require as much redundancy as longer k-mers but are more limited in representing complex sequences. When the coverage is low, as often occurs for low-abundance alleles, using a short k-mer can be critical for successful assembly (Supplemental Fig. 3). However, a shorter k-mer has a limited ability to resolve repeats, causing homologous sequences to collapse together. In contrast, a longer k-mer can better resolve repeats but may result in fragmented assembly due to insufficient coverage. Given the randomness of sequence complexity and coverage, it is conceivable that incrementally iterating through multiple different k-mers can produce improved assemblies (Chaisson et al. 2009).

One of most distinctive features of TIGRA is the implementation of such an iterative k-mer. TIGRA starts with a relatively small k-mer size when it uses multiple k-mers. The assembled contigs that are longer than the next k-mer size are included as pseudo reads in the next assembly iteration. Similar to other assemblers, the choice of the k-mer size is empirical and depends on the complexity of the genome and the property of the data. Our default double k-mer setting (15 bp followed by 25 bp) appears to be more effective than single k-mer settings at assembling breakpoints based on the data we have examined. For reads longer than 100 bp, a third k-mer of size 35 bp can often provide further improvement, especially for assembling insertions.

This idea of the iterative use of multiple k-mers was first developed in the Euler-SR assembly package (Chaisson et al. 2009). The approach of Chaisson et al. is different in that they used every pair of potentially connected contigs (as connected in the graph) as pseudo reads. Including more pseudo reads may be advantageous in assembling large genomes. However, it can also increase the risk of misassembly. In our setting, we chose to include individual contigs as pseudo reads. Individual contigs usually add less than 1× the sequence coverage per iteration and thus have a limited possibility of introducing errors.

Most assemblers discard singleton k-mers (those observed only once) because they are very likely introduced by random sequencing errors (Zerbino and Birney 2008). TIGRA implemented a similar treatment in constructing the initial graph. After that, TIGRA launched a process to rescue the singleton k-mers if they are able to connect significant graph components that cannot be connected otherwise. We used the same k-mer-hashing algorithm that was used in constructing the initial graph except that the assembled contigs (instead of raw reads) and the singleton k-mers were assessed.

### Tips and bubbles

Two major obstacles in genome assembly are sequencing errors and repeats. Sequencing errors form "tips" on the graph and can be easily removed. Mutations or sequencing errors can also form "bubbles," which consist of two alternative alleles with identical flanking sequences. Most current de Bruijn graph assemblers collapse the bubbles in the sequence graph to produce a single consensus sequence. Without care, this frequently leads to the loss of the alternative alleles. TIGRA retains this bubble structure when the lengths of the alternative alleles differ more than 3 bp because most NGS platforms produce errors shorter than 3 bp, and we are primarily interested in assembling alleles that have lengths or structures that differ substantially from the reference.

### Resolving repeats using reads

The de Bruijn graph collapses repeats when they are longer than the k-mer size. Such repeats can be recognized and potentially resolved based on coverage analysis, as implemented in Velvet. However, this approach does not work in a region with an altered copy number and is not robust in a region with biased coverage. Instead of using coverage, we used read sequences to resolve repeats, which are usually longer than the longest k-mer size. TIGRA identifies repeats from a "multiple in, multiple out" structure (Supplemental Fig. 4) and resolves them if there are spanning reads indicating an unambiguous connection. We found this solution to be more suitable for assembling SV breakpoints, which are usually associated with copy number alterations and are near low-complexity regions that often have abnormal coverage.

Using pairs may lead to further improvement in resolving repeats. However, since our primary focus is local assembly in short breakpoint regions, we did not implement that feature in our current version of TIGRA.

### Highlighting the nonreference sequence

In a TIGRA contig, we use uppercase letters to highlight novel sequences that differ from the reference and that are assembled from unmapped or poorly mapped reads. This feature makes it easy to identify breakpoints and highlight the differences between the alternative alleles and the reference.

### Cross_match alignment and SV calling

In our experiments, we used cross_match (http://www.phrap.org/phredphrapconsed.html) to examine whether a contig contained an SV breakpoint. For a contig assembled from an intra-chromosomal variant, we prepared a local reference sequence excised from [*a-w, b+w*]. For a contig assembled from an inter-chromosomal rearrangement, we prepared two local reference sequences from [*a-w, a+w*] of chromosome $c_1$ and from [*b-w, b+w*] of chromosome $c_2$, respectively. We mapped each contig assembled by TIGRA to the corresponding reference sequences using cross_match. In the default setting, we used the following cross_match parameters: -bandwidth 20 -minmatch 20 -minscore 25 -penalty -10 -discrep_lists -tags -gap_init -10 -gap_ext -1. We removed contigs that had more than two hits to the reference and ignored alignments that had substitution rates >0.5%. If a contig differs substantially from the reference, cross_match returns multiple local alignments together with a set of statistics describing the quality of the alignments. A glocal alignment (combination of local and global alignment) was constructed from these local alignments (Brudno et al. 2003). We used that alignment as the basis for reporting the existence of breakpoints and detailed information on the type, size, orientation, and location of the breakpoints (Supplemental Fig. 5). For example, the local alignment that supports a deletion breakpoint contains two local 1-monotonic alignments to the reference (Chen et al. 2012). The gap between the end position of the first alignment and the start position of the second alignment corresponds to the size of the deletion, whereas the bases shared by both alignments correspond to breakpoint homology.

### Running the assemblers

We ran TIGRA-0.3.7 under the default parameters, unless otherwise stated. Detailed usage is available in the Supplemental Notes and on the TIGRA website. We used Velvet version 1.2.09 for our analysis, with the following commands: "velveth 31 -short" and "velvetg -exp_cov auto." We ran both SGA-0.9.43 and SGA-0.9.17 using the following commands and parameters: "sga preprocess; sga index -a ropebwt; sga filter -x 2; sga overlap -m 15; sga assemble -m 15 -d 0 -g 0 -b 0 -l 100." We ran the expanded SGA assemblies (SGA.walk) with a modified 0.9.17 version using code from (Malhotra et al. 2013; Ira Hall, pers. comm.), with the following commands and parameters: "sga preprocess; sga index; sga filter -x 2; sga overlap -m 15; sga assemble -m 15 -d 0 -g 0 -b 0 -l 100; sga walk -d 10000–component-walks." We ran Phrap-1.080721 without any parameters. We ran SPAdes-2.5.0 in the single-cell mode on low-coverage data using "spades.py–sc–only-assembler -s –o." We ran it in the whole genome mode on high-coverage data using "spades.py–only-assembler -s –o."

### Manual inspection of TIGRA contigs

To manually inspect TIGRA contigs, we used the human BLAT search tool (http://genome.ucsc.edu/cgi-bin/hgBlat?command=start). We copy-pasted the assembled sequence onto the textbox and submitted it to the server. We reviewed the alignment results to assess the number, quality, and difference between the best and the rest, and the annotation of the alignments in the UCSC Genome Browser (e.g., whether they overlapped with any mobile elements or previously reported variants). For an MEI, the inserted sequence within the contig will align to its homologs on the reference, based on which we can assess its origin and sequence divergence. We considered this information collectively to infer how likely it was that a contig contained a valid breakpoint.

### Data

We downloaded the HiSeq NA12878 BAM file we used in this study from ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/technical/working/20120117_ceu_trio_b37_decoy/. We downloaded the low-depth CEU BAM files from http://ftp-trace.ncbi.nih.gov/1000genomes/ftp/pilot_data/data/ and from http://ftp-trace.ncbi.nih.gov/1000genomes/ftp/data/.

We obtained the 245 deletion breakpoints by intersecting the 1000 Genomes Pilot 1 deletion genotype file (http://ftp-trace.ncbi.nih.gov/1000genomes/ftp/pilot_data/release/2010_07/low_coverage/sv/CEU.low_coverage.2010_06.deletions.genotypes.vcf.gz) with the data in Supplemental Table 2 (ng.564-S2), which we downloaded from Conrad et al. (2010).

We obtained the 442 MEI breakpoints by extracting PCR-validated NA12878 sites from ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/pilot_data/paper_data_sets/companion_papers/mapping_structural_variation/union.2010_06.MobileElementInsertions.genotypes.vcf.gz.

We downloaded the dbRIP files from http://dbrip.brocku.ca/dbRIPdownload/.

### Software availability

TIGRA is available at http://bioinformatics.mdanderson.org/main/TIGRA. It is licensed under GPL-v3 and is free for academic use. Also provided is an example data set with a README file that explains the details of using TIGRA and the expected outcomes.

## Acknowledgments

We thank members of the 1000 Genomes Project SV group, Matt Hurles, Klaudia Walter, Chip Stewart, Bob Handsaker, Alexej Abyzov, Mark Gerstein, Jan Korbel, Can Alcan, and Evan Eichler for suggestions; Ira Hall for providing SGA expansion code; and Jared Thompson and Zamin Iqbal for their discussions. This work was supported by the National Cancer Institute through Grant R01-CA172652 to K.C., the Cancer Center Support Grant, P30-CA016672, and grant U24-CA143858; and the National Human Genome Research Institute through grants U01-HG006517, U54-HG003079, and U41-HG007497.

*Author contributions:* K.C., G.M.W., L.D.: project management; L.C., K.C.: assembly algorithm design; X.F., K.C., L.C.: software development; K.C., L.C., and J.W.: analysis; K.C., L.C., L.D., and G.M.W.: manuscript.

## References

The 1000 Genomes Project Consortium. 2010. A map of human genome variation from population-scale sequencing. *Nature* **467:** 1061–1073.

The 1000 Genomes Project Consortium. 2012. An integrated map of genetic variation from 1,092 human genomes. *Nature* **491:** 56–65.

Abyzov A, Urban AE, Snyder M, Gerstein M. 2011. CNVnator: An approach to discover, genotype, and characterize typical and atypical CNVs from family and population genome sequencing. *Genome Res* **21:** 974–984.

Alkan C, Sajjadian S, Eichler EE. 2011. Limitations of next-generation genome sequence assembly. *Nat Methods* **8:** 61–65.

Bankevich A, Nurk S, Antipov D, Gurevich AA, Dvorkin M, Kulikov AS, Lesin VM, Nikolenko SI, Pham S, Prjibelski AD, et al. 2012. SPAdes: A new genome assembly algorithm and its applications to single-cell sequencing. *J Comput Biol* **19:** 455–477.

Brudno M, Malde S, Poliakov A, Do C, Couronne O, Dubchak I, Batzoglou S. 2003. Glocal alignment: Finding rearrangements during alignment. *Bioinformatics* (Suppl 1) **19:** i54–i62.

Chaisson MJ, Brinza D, Pevzner PA. 2009. De novo fragment assembly with short mate-paired reads: Does the read length matter? *Genome Res* **19:** 336–346.

Chen K, Wallis J, McLellan M, Larson D, Kalicki J, Pohl C, McGrath S, Wendl M, Zhang Q, Locke D, et al. 2009. BreakDancer: An algorithm for high-resolution mapping of genomic structural variation. *Nat Methods* **6:** 677–681.

Chen K, Wallis JW, Kandoth C, Kalicki-Veizer JM, Mungall KL, Mungall AJ, Jones SJ, Marra MA, Ley TJ, Mardis ER, et al. 2012. BreakFusion: Targeted assembly-based identification of gene fusions in whole transcriptome paired-end sequencing data. *Bioinformatics* **28:** 1923–1924.

Conrad D, Pinto D, Redon R, Feuk L, Gokcumen O, Zhang Y, Aerts J, Andrews D, Barnes C, Campbell P, et al. 2009. Origins and functional impact of copy number variation in the human genome. *Nature* **464:** 704–712.

Conrad DF, Bird C, Blackburne B, Lindsay S, Mamanova L, Lee C, Turner DJ, Hurles ME. 2010. Mutation spectrum revealed by breakpoint sequencing of human germline CNVs. *Nat Genet* **42:** 385–391.

Ding L, Ellis M, Li S, Larson D, Chen K, Wallis J, Harris C, McLellan M, Fulton R, Fulton L, et al. 2010. Genome remodelling in a basal-like breast cancer metastasis and xenograft. *Nature* **464:** 999–1005.

Gu W, Zhang F, Lupski JR. 2008. Mechanisms for human genomic rearrangements. *Pathogenetics* **1:** 4.

Handsaker R, Korn J, Nemesh J, McCarroll S. 2011. Discovery and genotyping of genome structural polymorphism by sequencing on a population scale. *Nat Genet* **43:** 269–276.

Hormozdiari F, Alkan C, Eichler EE, Sahinalp SC. 2009. Combinatorial algorithms for structural variation detection in high-throughput sequenced genomes. *Genome Res* **19:** 1270–1278.

Iqbal Z, Caccamo M, Turner I, Flicek P, McVean G. 2012. *De novo* assembly and genotyping of variants using colored de Bruijn graphs. *Nat Genet* **44:** 226–232.

Iqbal Z, Turner I, McVean G. 2013. High-throughput microbial population genomics using the Cortex variation assembler. *Bioinformatics* **29:** 275–276.

Kent WJ. 2002. BLAT—The BLAST-like alignment tool. *Genome Res* **12:** 656–664.

Kidd J, Cooper G, Donahue W, Hayden H, Sampas N, Graves T, Hansen N, Teague B, Alkan C, Antonacci F, et al. 2008. Mapping and sequencing of structural variation from eight human genomes. *Nature* **453:** 56–64.

Kidd JM, Graves T, Newman TL, Fulton R, Hayden HS, Malig M, Kallicki J, Kaul R, Wilson RK, Eichler EE. 2010. A human genome structural variation sequencing resource reveals insights into mutational mechanisms. *Cell* **143:** 837–847.

Korbel JO, Urban AE, Affourtit JP, Godwin B, Grubert F, Simons JF, Kim PM, Palejev D, Carriero NJ, Du L, et al. 2007. Paired-end mapping reveals extensive structural variation in the human genome. *Science* **318:** 420–426.

Lam H, Mu X, Stütz A, Tanzer A, Cayting P, Snyder M, Kim P, Korbel J, Gerstein M. 2010. Nucleotide-resolution analysis of structural variants using BreakSeq and a breakpoint library. *Nat Biotechnol* **28:** 47–55.

Li H, Durbin R. 2009. Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics* **25:** 1754–1760.

Li H, Handsaker B, Wysoker A, Fennell T, Ruan J, Homer N, Marth G, Abecasis G, Durbin R. 2009. The Sequence Alignment/Map format and SAMtools. *Bioinformatics* **25:** 2078–2079.

Li R, Zhu H, Ruan J, Qian W, Fang X, Shi Z, Li Y, Li S, Shan G, Kristiansen K, et al. 2010. De novo assembly of human genomes with massively parallel short read sequencing. *Genome Res* **20:** 265–272.

Malhotra A, Lindberg M, Faust GG, Leibowitz ML, Clark RA, Layer RM, Quinlan AR, Hall IM. 2013. Breakpoint profiling of 64 cancer genomes reveals numerous complex rearrangements spawned by homology-independent mechanisms. *Genome Res* **23:** 762–776.

Mills R, Walter K, Stewart C, Handsaker R, Chen K, Alkan C, Abyzov A, Yoon SC, Ye K, Cheetham K, et al. 2011. Mapping copy number variation by population-scale genome sequencing. *Nature* **470:** 59–65.

Peng Y, Leung HC, Yiu SM, Chin FY. 2012. IDBA-UD: A *de novo* assembler for single-cell and metagenomic sequencing data with highly uneven depth. *Bioinformatics* **28:** 1420–1428.

Pevzner P, Tang H, Waterman M. 2001. An Eulerian path approach to DNA fragment assembly. *Proc Natl Acad Sci* **98:** 9748–9753.

Rausch T, Zichner T, Schlattl A, Stütz AM, Benes V, Korbel JO. 2012. DELLY: Structural variant discovery by integrated paired-end and split-read analysis. *Bioinformatics* **28:** i333–i339.

Simpson JT, Durbin R. 2010. Efficient construction of an assembly string graph using the FM-index. *Bioinformatics* **26:** i367–i373.

Stephens PJ, McBride DJ, Lin ML, Varela I, Pleasance ED, Simpson JT, Stebbings LA, Leroy C, Edkins S, Mudie LJ, et al. 2009. Complex landscapes of somatic rearrangement in human breast cancer genomes. *Nature* **462:** 1005–1010.

Stewart C, Kural D, Strömberg MP, Walker JA, Konkel MK, Stütz AM, Urban AE, Grubert F, Lam HY, Lee WP, et al. 2011. A comprehensive map of mobile element insertion polymorphisms in humans. *PLoS Genet* **7:** e1002236.

Wang J, Song L, Grover D, Azrak S, Batzer MA, Liang P. 2006. dbRIP: A highly integrated database of retrotransposon insertion polymorphisms in humans. *Hum Mutat* **27:** 323–329.

Welch J, Westervelt P, Ding L, Larson D, Klco J, Kulkarni S, Wallis J, Chen K, Payton J, Fulton R, et al. 2011. Use of whole-genome sequencing to diagnose a cryptic fusion oncogene. *JAMA* **305:** 1577–1584.

Yang L, Luquette LJ, Gehlenborg N, Xi R, Haseley PS, Hsieh CH, Zhang C, Ren X, Protopopov A, Chin L, et al. 2013. Diverse mechanisms of somatic structural variations in human cancer genomes. *Cell* **153:** 919–929.

Ye K, Schulz MH, Long Q, Apweiler R, Ning Z. 2009. Pindel: A pattern growth approach to detect break points of large deletions and medium sized insertions from paired-end short reads. *Bioinformatics* **25:** 2865–2871.

Yoon S, Xuan Z, Makarov V, Ye K, Sebat J. 2009. Sensitive and accurate detection of copy number variants using read depth of coverage. *Genome Res* **19:** 1586–1592.

Zerbino D, Birney E. 2008. Velvet: Algorithms for de novo short read assembly using de Bruijn graphs. *Genome Res* **18:** 821–829.

Zhu M, Need AC, Han Y, Ge D, Maia JM, Zhu Q, Heinzen EL, Cirulli ET, Pelak K, He M, et al. 2012. Using ERDS to infer copy-number variants in high-coverage genomes. *Am J Hum Genet* **91:** 408–421.

*Received July 1, 2013; accepted in revised form December 3, 2013.*