BIOINFORMATICS ORIGINAL PAPER

Vol. 30 no. 24 2014, pages 3484–3490
doi:10.1093/bioinformatics/btu431

Sequence analysis

Advance Access publication July 14, 2014

Gustaf: Detecting and correctly classifying SVs in the NGS twilight zone

Kathrin Trappe[1,2,*], Anne-Katrin Emde[3], Hans-Christian Ehrlich[1] and Knut Reinert[1]

[1]Department of Computer Science, Freie Universität Berlin, 14195 Berlin, Germany, [2]Research Group Bioinformatics (NG4), Robert Koch Institute, 13353 Berlin, Germany and [3]New York Genome Center, New York, NY 10013, USA

Associate Editor: Gunnar Ratsch

ABSTRACT

Motivation: The landscape of structural variation (SV) including complex duplication and translocation patterns is far from resolved. SV detection tools usually exhibit low agreement, are often geared toward certain types or size ranges of variation and struggle to correctly classify the type and exact size of SVs.

Results: We present Gustaf (Generic mUlti-SpliT Alignment Finder), a sound generic multi-split SV detection tool that detects and classifies deletions, inversions, dispersed duplications and translocations of $\geq$30 bp. Our approach is based on a generic multi-split alignment strategy that can identify SV breakpoints with base pair resolution. We show that Gustaf correctly identifies SVs, especially in the range from 30 to 100 bp, which we call the next-generation sequencing (NGS) twilight zone of SVs, as well as larger SVs >500 bp. Gustaf performs better than similar tools in our benchmark and is furthermore able to correctly identify size and location of dispersed duplications and translocations, which otherwise might be wrongly classified, for example, as large deletions.

Availability and implementation: Project information, paper benchmark and source code are available via http://www.seqan.de/projects/gustaf/.

Contact: kathrin.trappe@fu-berlin.de

Received on April 11, 2014; revised on June 25, 2014; accepted on June 29, 2014

1 INTRODUCTION

Variation in the human genome remains only partially characterized. Theoretically, whole-genome sequencing (WGS) data carry the potential to identify all genomic variation, including small variants such as single nucleotide variants and small indels, as well as larger structural variants (SVs; typically defined as >50 bp), such as large deletions, inversions, duplications and translocations. However, the difficulty lies in computationally analyzing the large-scale data obtained from WGS and reliably identifying the whole range of genomic variation.

With current algorithms and methods, only small variants, up to 30 bp, can be identified rather confidently, while larger variation still poses a major challenge (Alkan *et al.*, 2011). Large and complex rearrangements are often accompanied by micro-homologies or microindels (Onishi-Seebacher and Korbel, 2011) around their breakpoints, which makes them even harder to detect, particularly with base pair resolution.

Many bioinformatics tools have been developed in recent years that address SV detection through identification of certain signatures in the sequencing data (Alkan *et al.*, 2011). Mainly, they rely on one or multiple of the following four approaches: (i) identifying discordant read pairs that span SV breakpoints (Chen *et al.*, 2009; Hormozdiari *et al.*, 2009; Marschall *et al.*, 2012; Tuzun *et al.*, 2005), (ii) detecting regions of unexpectedly low or high read depth (Abyzov *et al.*, 2011; Xi *et al.*, 2011; Yoon *et al.*, 2009), (iii) identifying split reads that span SV breakpoints (Emde *et al.*, 2012; Ye *et al.*, 2009) or (iv) local reassembly of SV candidate regions (Chen *et al.*, 2013). Typically, overlap between different tools is low (Alkan *et al.*, 2011). This is partially due to the fact that most tools are geared toward certain types of data, e.g. for specific read lengths, or toward certain SV size ranges and types, e.g. mid-size indels. Also, the different strategies suffer from various biases and sources of errors. Read depth methods are vulnerable to read depth fluctuations leading to non-uniform coverage, e.g. due to GC content and mappability issues. Even after successfully normalizing for coverage biases, the spectrum of SVs that can be identified through read depth signatures is limited to copy number variable regions, i.e. deletion and amplification. Read pair methods are vulnerable to mis-mappings caused by repetitive regions in the genome and are furthermore susceptible to chimeric read pairs (Maher *et al.*, 2009).

Both read depth and read pair methods have problems identifying small SVs, i.e. less than a few hundred base pairs, which we here term the next-generation sequencing (NGS) twilight zone of SVs. For read depth methods, it is hard to discern coverage changes in small regions from natural read depth fluctuations. For read pair methods, such relatively small deletions are difficult to identify because paired read spacing may still lie within the variance of the insert size distribution.

Of the four strategies, only split-reads- and assembly-based methods yield single nucleotide resolution, by identifying reads or reconstructing contigs that directly span SV breakpoints. When correctly mapped onto the reference genome, the read- (or contig-)to-reference alignment of an SV-spanning read (contig) will be split into partial alignments at the breakpoint positions, hence yielding base pair resolution.

The split-read approach has been primarily used in conjunction with read pair methods that identify potential SV-spanning reads through abnormal paired read distance or orientation (Marschall *et al.*, 2013; Rausch *et al.*, 2012; Ye *et al.*, 2009). This significantly reduces the search space for split mapping, which is computationally expensive to apply to the whole genome.

*To whom correspondence should be addressed.

However, with increasing read lengths (and improved local reassembly approaches that generate contigs), the split-mapping approach becomes more and more powerful, as it yields highest confidence and base pair resolution.

Therefore, we aimed to develop a method that can generically split-map contigs or reads of arbitrary length. Our tool Gustaf (Generic mUlti-SpliT Alignment Finder) allows for multiple splits at arbitrary locations in the read, is independent of read length and sequencing platform and supports both single-end and paired-end reads.

Gustaf is based on finding local alignments of a read, and then essentially chaining local alignments into a semi-global read-to-reference alignment. Similar approaches are used in the context of whole-genome alignment, where large rearrangements between locally collinear blocks are maintained in graph or graph-like structures, e.g. the alignment graph (Kececioglu, 1993), the A-Bruijn graph (Pevzner *et al.*, 2004) or the Enredo graph (Paten *et al.*, 2008), or in the Shuffle-LAGAN glocal alignment algorithm (Brudno *et al.*, 2003).

Local alignments are identified with Stellar (Kehr *et al.* 2011; www.seqan.de/projects/stellar), an edit distance local aligner, which guarantees to find all local alignments of a given minimal length, maximal error rate and maximal X-drop. In theory, however, our approach can take local alignments from any aligner as input, making it versatile and adaptable. Because local alignments are allowed to be anywhere in the reference genome, it allows for non-collinearity of chained local alignments, and hence has the power to identify all types of SV.

In contrast to other SV callers, Gustaf furthermore attempts to correctly classify SV events leading to multiple breakpoints such as translocations and dispersed duplications including the actual length of the duplication (see also Fig. 1). These complex SV patterns incorporate pseudodeletions that make them harder to classify and that are often wrongly reported by other methods.

In the following, we introduce our method Gustaf and compare it with two other popular methods that are able to report SVs with base pair resolution, Pindel (Ye *et al.*, 2009) and Delly (Rausch *et al.*, 2012).

2 METHODS

Gustaf is an SV detection tool that uses a split-read approach to detect exact breakpoints of SVs including indels, inversions, duplications and translocations. Gustaf uses the local aligner Stellar to detect partial alignments of a read and stores compatibility information of these partial alignments in a graph data structure.

We will first give a brief overview of Gustaf's general approach and then define all important steps such as local alignments, adjacency of local alignments, breakpoints of different types and our split-read graph together with the necessary methods such as the split alignment in the following paragraphs.

2.1 Gustaf's workflow

Gustaf takes as input a set of reads and optionally a set of local alignments for these reads. If no local alignments are supplied, they will be computed using Stellar. Our approach is based on maintaining all local alignments of a read within a graph structure so that we can use standard graph algorithms to evaluate relationships of the alignments. We call this graph *split-read graph* and will completely define it below (see also Figs. 2 and 3). In this graph, each local alignment is represented by one vertex.

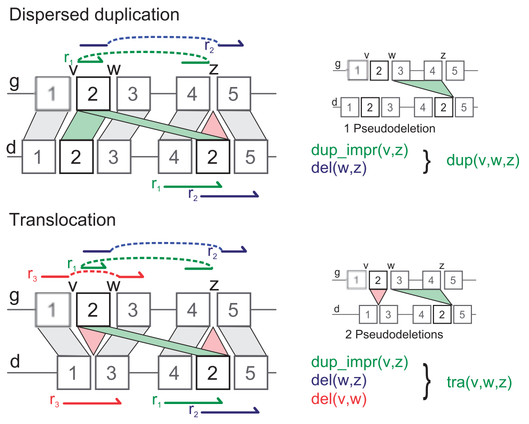

Fig. 1. Duplication and translocation alignment patterns: On the left side, we show alignment patterns of a reference *g* (upper sequence) with a donor genome *d* where sequence Block 2 is either duplicated (upper figure) or translocated within the donor genome. In a read-to-reference alignment, read r_1 indicates the duplication or translocation event (*dup_impr*(v, z)) through the different order of the read parts within the reference. We also observe pseudodeletions for both variants (highlighted on right side) through reads r_2 and r_3: an upstream duplication of Block 2 creates a pseudodeletion *del*(w, z), an upstream translocation pseudodeletions *del*(w, z) and *del*(v, w). From observing both *dup_impr*(v, z) and *del*(w, z), we infer the duplication *dup*(v, w, z). If we also observe *del*(v, w), we infer the translocation *tra*(v, w, z)

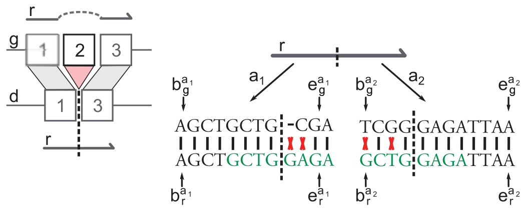

Fig. 2. Local alignments a_1 and a_2 of a read *r* spanning Blocks 1 and 3 in a donor genome *d* where Block 2 is deleted compared with the upper reference genome *g*. Alignment a_1 denotes the local alignment of the read prefix with Block 1, a_2 of the read suffix with Block 3. Denoted are begin (*b*) and end (*e*) positions of the alignment a_1 in the reference ($b_g^{a_1}$, $e_g^{a_1}$) and the read ($b_r^{a_1}$, $e_r^{a_1}$), and positions of a_2 analogous to a_1. Alignments a_1 and a_2 are adjacent in the read and overlap at their sloppy end within the read (bases GCTGGAGA). The correct split position of the deletion is indicated by the dotted line. Both a_1, a_2 have edit distance 2. The gained score d' of this split alignment is –4, as we get rid of all errors within the sloppy ends when cutting the alignments at the dotted line

Breakpoints (like positions *v*, *w*, *z* in Fig. 1) create new sequence adjacencies in a genome, e.g. Blocks 4 and 5 in Figure 1 are adjacent in the reference genome but formed new adjacencies with the duplicated (or translocated) Block 2 in the donor genome. Vertices of alignments that belong to the same variant, i.e. that span a breakpoint, are connected by an edge so that the edge represents the breakpoint. Each alignment has an edit distance. Roughly, each edge carries as weight the edit distance of its target vertex (see also Fig. 3, more details will follow below).

All valid combinations of alignments are represented by their corresponding paths of vertices in the graph. The sum of the edge weights of each path will correspond to the edit distance of the individual split alignments, with additional penalties incurred for each split depending

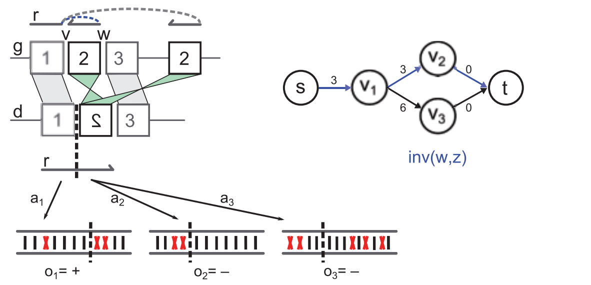

Fig. 3. Gustaf's workflow detecting an inversion of Block 2 in the donor genome d where another similar Block 2 is present in the reference g: A read spanning Blocks 1 and 2 results in alignments a_1, a_2 and a_3 (with $o_1 = +$ and $o_2 = o_3 = -$) where a_3 is an alignment to the region similar to Block 2 upstream in the reference. The corresponding split-read graph (top-right) has artificial start (s) and end (t) vertices, representing start and end of the read, and vertices v_1-v_3 representing alignments a_1-a_3. Ingoing edge labels are edit distances of corresponding alignments adjusted by inversion penalty 5 and the gained split-alignment score (4 for both v_2 and v_3). The shortest path from s over v_1 and v_2 to t represents the most likely alignment combination (a_1, a_2) of the inversion

on the type and size of the SV it indicates (see Fig. 3 for an inversion example $inv(w, z)$ for a_1, a_2 with edit distance $d^{a_2} = 2$, inversion penalty 5 and gained split-alignment score $d' = -4$). The path with the lowest total weight represents the most likely combination of local alignments for this read, and the edges give the breakpoints causing the split in the read-to-reference alignment.

Local alignments. Gustaf uses SeqAn's local aligner Stellar (Kehr *et al.*, 2011) to compute the set of local alignments $\mathcal{A}(r)$ of each read r. Stellar implements a seed and extend approach based on the SWIFT filter algorithm (Rasmussen *et al.*, 2006) and guarantees to find all local alignments between two sequences given a minimal match length and maximal error rate for these alignments. During the extension phase, the seeds are extended as long as the final alignment is still valid with respect to the allowed maximal error rate. This produces sloppy alignment ends. In case a read spans an SV, the alignment is thus likely to extend past the SV breakpoint. This is an important feature that we use to our advantage in the subsequent split-graph construction (see also Fig. 2).

Throughout the article, we use the following definitions: Let $s = (b, e)$ be a segment of a sequence starting in b and ending in e, i.e. $b < e$. An alignment $a = \{s_g, s_r, o\}$ between a reference g, and a read r aligns the segment $s_g = (b_g, e_g)$ of g to the segment $s_r = (b_r, e_r)$ of r. The orientation $o \in \{+, -\}$ indicates whether the read mapped to the forward (+) or reverse (–) strand of the reference. We denote the read segment positions b_r and e_r of an alignment a with b_r^a and e_r^a, and the reference positions b_g^a and e_g^a (see Fig. 2). Every alignment a has an edit distance d^a.

For two alignments a_1, a_2, we say $a_1 < a_2$ if $b_r^{a_1} < b_r^{a_2}$, i.e. we can sort all alignments of one read according to their start position in the read. Analogously, we can impose an ordering of the alignments according to their reference sequence positions.

A read that spans a breakpoint is split up in alignments, and we identify the alignments belonging to the same variant according to their adjacency, i.e. we say that two alignments span a breakpoint if they fulfill the following criteria of adjacency for alignments. Two alignments can be adjacent regarding their read or their reference sequence (or both). For read adjacency, two alignments a_1 and a_2 with $a_1 < a_2$ are adjacent if their read positions overlap such that $b_r^{a_2} < e_r^{a_1}$, or if the gap between the read segments is smaller than a predefined threshold T_g, i.e. $b_r^{a_2} - e_r^{a_1} < T_g$. The gap definition of adjacency accounts for possible microindels around a breakpoint and is per default 5 bp but can be adjusted by the user. For reference adjacency, a_1, a_2 are adjacent if their reference positions overlap such that either $b_g^{a_2} < e_g^{a_1}$ (and $b_g^{a_1} < b_g^{a_2}$) or $b_g^{a_1} < e_g^{a_2}$ (and $b_g^{a_2} < b_g^{a_1}$), or if the gap between the reference segments is smaller T_g.

The adjacency and relation of two alignments a_1, a_2, indicate a specific type of SV (see SV classification section). The positions of the SVs are determined by the begin and end positions of the reference segments in a_1 and a_2, refined by the split-alignment method described in the next paragraph.

Split alignment. Two adjacent alignments a_1, a_2 with $a_1 < a_2$ that overlap at their sloppy ends are realigned in their overlapping region to determine the exact breakpoint. This realignment is in principle similar to the split-alignment approach in AGE (Abyzov and Gerstein, 2011) and is implemented as an alignment algorithm in SeqAn (Döring *et al.*, 2008; Emde *et al.*, 2012). Similar to AGE, two alignment matrices for the overlapping parts of two adjacent alignments a_1, a_2 of a read r are computed simultaneously. That is, we have a segment $s_o = (b_r^{a_2}, e_r^{a_1})$ denoting the overlapping region in the read, and two alignments $a_1' = \{s_1', s_o, o_1\}$ and $a_2' = \{s_2', s_o, o_2\}$ where s_i' is the subsequence of s_g in a_i that aligns to s_o. The matrices of a_1', a_2' determine the best split point p as the position in the read with the best total edit distance score for both alignments a_1' and a_2'. The segments s_r, s_g of the alignments a_1 and a_2 are trimmed according to p, i.e. s_r^t of a_1^t is $s_r^t = (b_r^{a_1}, p)$ and s_r^t of a_2^t is $s_r^t = (p, e_r^{a_2})$, and s_g^t of a_1^t and a_2^t is the subsequence that aligns to s_r', respectively. The new edit distances of a_1^t and a_2^t are lower than the old ones d^{a_1} and d^{a_2} of the untrimmed alignments a_1, a_2, i.e. we have a gain d' of edit score when we split align a_1 and a_2.

All adjacency and split-alignment information is represented in a graph that we call split-read graph and that is defined in the following paragraph.

Split-read graph. All local alignments $\mathcal{A}(r)$ of a read r and their adjacency relation to each other are represented in a split-read graph $G = (V, E, s, t)$ (see Fig. 3). When building the graph, we start with an almost empty graph containing only two artificial vertices representing *start (s)* and *end (t)* of a read, and then add a vertex $v \in V$ for each local alignment $a \in \mathcal{A}$ of the read. We add directed edges $e_s = (s, v)$ and $e_e = (v, t)$ for every a (represented by v) that misses $t_i^a < T_i$ base pairs to either start or end of the read, and weight them with $w(e_s) = d^a + t_i^a$ and $w(e_e) = t_i^a$.

We then add directed edges $e = (v_1, v_2)$ between all pairs of alignments a_1, a_2 that fulfill the criteria of adjacency defined above and weight the edge with $d^{a_2} - d'$, the edit score of a_2 adjusted by the gained score of the split alignment of a_1 and a_2. Based on the type of split that an edge supports, an additional penalty is added to the weight of the edge. Splits that agree with a collinear alignment of adjacent local alignments, i.e. are suggesting a simple insertion or deletion event, receive a penalty of 0. All other splits, i.e. suggesting translocation, inversion or duplication, receive a higher penalty. We set all penalties to 5 here, i.e. for a 100 bp read, a non-collinear split will be weighted equivalently to five mismatches in a read-to-reference alignment, but these penalties can be adapted by the user depending on the application. Edges reflect adjacency of the alignments either within the read or within a reference genome. The direction of the edge depends only on the alignment order in the read, such that if a_1 and a_2 with $a_1 < a_2$ are adjacent, then there is an edge $e = (v_1, v_2)$, independent of whether the adjacency is in the read or reference. This definition guarantees that we have a directed acyclic graph (DAG) for which we use common graph algorithms like the DAG shortest path algorithm.

The adjacency edges create zero to multiple paths through the graph from *start* to *end* where the sum of the edge weights of each path correspond to the total edit distance of the alignments on the path plus penalties incurred for the indicated SV types (see Fig. 3 for an inversion example with two alternative paths). Adjusting thresholds T_g and T_i for gaps allowed at beginning or within reads influences sensitivity and specificity when adding edges and therefore also the number of paths. We identify the most likely path using a DAG shortest path algorithm.

SV classification. We define the different SV types according to the positions and sequence content affected by the variation. A deletion $del(b, e)$ is then a segment $s = (b, e)$ in the reference g that is absent in the donor genome (see absent Block 2 in Fig. 2). An inversion $inv(b, e)$ is a segment s that is inverted in g (see inverted Block 2 in Fig. 3). A dispersed duplication $dup(b, e, t)$ is a segment $s = (b, e)$ that appears again at position t in g (see duplicated Block 2 at position z in Fig. 1). If position e of s could not be inferred by the classification described below, we refer to the duplication as imprecise, denoted by $dup\_impr(b, t)$ (see imprecise duplication indicated by Read 1 in Fig. 1). A translocation is usually annotated by the new adjacencies formed by the translocation process. We denote a translocation $tra(b, m, t)$ by the three positions b, m, t involved in the new adjacencies, i.e. there is either a segment $s_1 = (b, m)$ translocated to positions t or a segment $s_2 = (m, t)$ translocated to position b (see translocated Block 2 at position z in Fig. 1).

The adjacency and relation of two alignments a_1 and a_2 indicate a specific type of SV. If g_1, g_2 of a_1, a_2 are different, both alignments are from different chromosomes indicating an inter-chromosomal translocation. When the orientation of both alignments is different, i.e. $o_1 \neq o_2$, a_1 and a_2 are spanning an inversion breakpoint (like a_1 and a_2, or a_1 and a_3, in Fig. 3). If a_1, a_2 are adjacent in the read and there is a gap $b_g^{a_2} - e_g^{a_1} > T_g$ between the reference positions, then the donor genome is missing sequence content at this breakpoint caused by a deletion event $del(e_g^{a_1}, b_g^{a_2})$. After an insertion event, a_1, a_2 are adjacent in the reference but not the read. Adjacency in both reference and read can indicate small indels or tandem duplications. Tandem duplications, dispersed duplications and intra-chromosomal translocations cause a change in the order of the alignments between read and reference, i.e. $b_r^{a_1} < b_r^{a_2}$ and $b_g^{a_2} < b_g^{a_1}$ (like alignments for Read 1 in Fig. 1). From this observation, we can only infer an imprecise duplication $dup\_impr(b_g^{a_2}, e_g^{a_1})$. Therefore, we cross-validate observed deletions and possible duplications to infer the missing middle coordinate of the duplication or to distinguish a duplication from an intra-chromosomal translocation. The difference between a duplication and an intra-chromosomal translocation event is an additional deletion pattern of the translocated region (see pseudodeletions and reads r_2 and r_3 in Fig. 1), upstream or downstream of the read containing the duplication pattern, which can usually not be observed by a single read. So given a $dup\_impr(b_g^{a_2}, e_g^{a_1})$, if we observe a $del_1(b_g^{a_2}, m)$, we infer a $dup(m, e_g^{a_1}, b_g^{a_2})$. If we observe a $del_2(m, e_g^{a_1})$, we infer $dup(b_g^{a_2}, m, e_g^{a_1})$. Only if we see both del_1 and del_2, we know one of them is the upstream or downstream deletion marking the event as a translocation $tra(b_g^{a_2}, m, e_g^{a_1})$ (see also Fig. 1).

Sometimes a read reaches over a breakpoint, with one end being mappable, whereas the other is not, e.g. if the non-mappable part belongs to a large insertion or is simply too short for a mapper to be found. We refer to these breakpoints, which can only be observed from one end as *breakends*. Breakends that do not support already observed and classified SVs are reported as unrefined (due to the missing second alignment) single breakends.

Paired-end data. We described the workflow for single-end reads so far. The paired-end version is an extension to the described approach. By joining both mates and treating them as a single read, we obtain a graph with an expected split at the joining position of the mates. Edges in the graph that stem from this artificial split and are in agreement with the library insert size, receive an edge weight bonus that makes any path through this edge more likely to be the best path, i.e. the shortest path. The benefit from this version lies in the higher probability of choosing the correct alignments of a split read and thereby increasing the specificity of the SV calling.

3 RESULTS AND DISCUSSION

We benchmarked both the single-end (se) and paired-end (pe) version of Gustaf on a set of simulated data to evaluate the performance in terms of sensitivity (S) and positive predictive value (PPV) and to compare Gustaf with other state-of-the-art SV detection tools, namely, Delly (Rausch *et al.*, 2012) and Pindel (Ye *et al.*, 2009). Suitable real datasets with sufficient annotations are rarely available. Especially, sets with confident annotations of duplications and translocations are rare. The detection of these two SVs is one of the strengths that distinguish Gustaf from other SV detecting tools. Therefore, we use a simulated dataset that mimics real-world data to show Gustaf's ability to correctly identify dispersed duplications and intra-chromosomal translocations.

Gustaf works with both single-end and paired-end reads, whereas Delly and Pindel run only on paired-end data. Therefore, we test both the single-end and the paired-end versions of Gustaf. Using the paired-end version, we would expect higher precision rates because the paired-end information can help to identify the correct alignments of a split compared with the single-end version.

We expect good recovery and precision rates of Gustaf for an average parameter set of a typical Illumina run where the coverage is high enough (coverage of 15), for both single-end and paired-end data. Our approach should also work for low coverage runs (coverage of 5 or 10) and become even more sensitive for high coverage (coverage of 30). The results should also be independent of the insert size distribution of the paired-end library, which we test using simulated insert sizes of $(\mu, \sigma) = (300, 30)$, $(\mu, \sigma) = (600, 60)$ and $(\mu, \sigma) = (1000, 100)$.

We test all methods on a set of variants from 30 to 500 bp, including the NGS twilight zone from 30 to 100 bp, a size range usually not in the focus of other SV detection tools. In addition, we also investigate a set of large variants ranging from 500 to 5000 bp, a size range most other tools operate on because they often rely on discordant read pairs.

Simulated dataset. We used SeqAn's Mason (Holtgrewe (2010); www.seqan.de/projects/mason) to create a simulated dataset out of a random 1 Mbp region of the human chromosome 22 from the latest reference genome *hg19*.

The newest version of Mason (Mason2, Holtgrewe, unpublished) includes a variant simulator that can simulate single nucleotide polymorphisms (SNPs), small indels and SVs including large indels, inversions, dispersed duplications and intra-chromosomal translocations at specified size ranges and simulation rates in a non-overlapping manner. To have a sufficient number and distribution of SVs, we used a simulation rate of 10^{-4} for indels, inversions and dispersed duplications, and 5.0×10^{-5} for intra-chromosomal translocations, resulting in a set of 330 SVs including 48 deletions, 93 inversions, 99 duplications and 50 translocations for the SV range of 30–500 bp. For the large variants, the set consists of 159 variants including 23 deletions, 44 inversions, 43 duplications and 28 translocations.

The simulated genome includes SNPs and small indels (1–6 bp) simulated with Mason's default simulation rate to account for natural variation of the sequence content, although these SNPs and small indels are not part of the evaluation.

Single-end and paired-end Illumina reads, created from fragments whose sizes follow a normal distribution, are obtained by using Masons's read simulator for both variant sets. For the default run, the reads have a length of 100 bp with the default Illumina error probabilities (including a quality decrease at the end of a read) set in Mason, a coverage of 15, and the paired-end reads have a mean fragment size $\mu = 300$ and standard deviation $\sigma = 30$.

In addition, simulated reads for the set of small variants are generated with coverages of 5, 10 and 30, paired-end library sizes of $(\mu, \sigma) = (600, 60)$ and $(\mu, \sigma) = (1000, 100)$ and a read set with read length 150 bp where we set the library size to $(\mu, \sigma) = (600, 60)$ to avoid mate overlaps.

Configurations. Stellar (Kehr *et al*., 2011) runs with a minimal match length of 30 bp, maximal error rate 0.03 and the default X-drop of 5 bp.

Gustaf requires a support of minimal three reads for an SV to call for a coverage >5, and a support of minimal two reads for a coverage = 5. The threshold T_i for the allowed number of missing base pairs at start and end of a read is set to 30, which equals Stellar's minimal match length. This way, we can bridge missing alignments smaller than the minimal match length at start and end of a read.

The Burrows–Wheeler Aligner (Li and Durbin, 2010) using default parameter settings constructs the mapping of the simulated paired-end reads for Delly and Pindel. Both were used with their default parameters and in default mode using the latest available versions on their Web sites (Delly: https://github.com/tobiasrausch/delly; Pindel: https://github.com/genome/pindel, March, 2014). The default configuration is most likely optimized by the author to detect large SVs rather than small ones, and therefore, we think that the obtained results have some room for improvement. Nevertheless, we believe that a comparison yields valuable insights into how Delly and Pindel perform when searching for small SVs.

Benchmark Analysis. The analysis of the benchmark results consists of the following three parts. In a first stage, we want to compare all tools on even ground independent of their SV classification. Therefore, we compare all predicted single breakpoints, i.e. all novel adjacencies in the donor genome, of all three tools with the simulated variants from Mason without considering the called SV type. A single breakpoint here is a single position of an SV, e.g. for a called deletion $del(b, e)$, which starts in position b and ends in position e, we would compare the positions b and e separately to the set of all single positions simulated by Mason and for now also ignore that both positions belong to a deletion. We count a predicted single breakpoint as true positive (TP) if the difference to the simulated single breakpoint is at most 10 bp.

Next, we want to evaluate the SV classification of all tools and therefore compare the called SVs according to their SV type including a classification for deletions, inversions and duplications, i.e. we compare a called deletion $del_1(b_1, e_1)$ to a simulated deletion $del_2(b_2, e_2)$ considering both positions b_i and e_i. Here, we consider all predicted duplications as imprecise denoted as $dup\_impr(b, t)$. This way, we compare the begin and target positions b, t predicted by all tools with the begin and target position of the dispersed duplications $dup(b, m, t)$ simulated by Mason but disregarding the known length so that we can directly compare Gustaf with Pindel and Delly.

In contrast to Delly and Pindel, Gustaf classifies and fully annotates dispersed duplications and intra-chromosomal translocations. In the last part, we therefore compare the dispersed duplications $dup(b, m, t)$ and translocations $tra(b, m, t)$ predicted by Gustaf considering also the predicted length of the duplication.

For the second and third analysis part, we require a reciprocal overlap of 80% of the simulated SV length and the predicted SV length to count the predicted SV as a TP. All predicted SVs not in the simulated set are counted as false positive (FP), all not recovered simulated SVs as false negative (FN). We compare the results using sensitivity values $S = TP/(TP + FN)$ and PPVs (or precision) $PPV = TP/(TP + FP)$. The results for coverages of 5, 10, 15 (default run) and 30, insert sizes for paired-end data $(\mu, \sigma) = (600, 60)$ and $(\mu, \sigma) = (1000, 100)$, read length 150 bp (with $(\mu, \sigma) = (600, 60)$ to avoid read overlaps) and different SV size ranging from 500 to 5000 bp are reported in Table 1.

Benchmark results. In general, we want a well-performing tool to have both a high sensitivity and precision for a given set of parameters. We will first evaluate the results of all methods on the set of small SVs with the tested parameters settings for coverages, insert size and read length, and then evaluate the set of large SVs below.

Gustaf has the highest sensitivity for the small SVs usually >95% over all tested parameter settings usually with a PPV >90%. Note that Gustaf requires a support of three reads for a coverage of 10, and only two for a coverage of 5. Therefore, Gustaf's sensitivity for a coverage of 5 is higher (97.9–70.8%) but at the cost of a lower PPV (26.3–35.4%).

For high enough coverage, Gustaf yields a perfect sensitivity with a PPV of ~92.1%, and Pindel and Delly improve toward their highest single breakpoints sensitivities of 92.8 and 71.8%, respectively. Otherwise, Delly's and Pindel's sensitivity values vary with coverage, insert size and read length. We observe a slightly higher precision for Pindel for the single breakpoints (>99.2%) and inversions (>97.6%) compared with Gustaf (>92.1, >92%) and Delly (>57.8, >59.7%), but almost always at the cost of a lower sensitivity. Regarding the single breakpoint evaluation, Gustaf always has both high sensitivity and precision for all tested parameter settings, whereas Delly and Pindel have a high variability with varying parameters and between sensitivity and precision.

For deletions, Gustaf has also high sensitivity values (reaching 100% for a coverage of 30), although at lower PPVs (usually >80%). Delly and Pindel have moderate sensitivities of up to 68.8% (Delly) and up to 95.8% (Pindel) for sufficient coverage. The drastically low PPVs of Delly and Pindel were surprising. A deeper analysis showed that Delly and Pindel call the aforementioned pseudodeletion patterns as deletions. Therefore, we validated the assumption of called pseudodeletions by including all pseudodeletions into the set of simulated deletions for which Delly and Pindel reached much higher precisions (Delly up to 77.6%, Pindel up to 99.5%, data not shown). This supports our approach of using these deletions to resolve dispersed duplication and translocation patterns. It also explains why the PPVs for the single breakpoints are so much higher (>91%) and why Rausch *et al*. (2012) report much higher precision values (>95%) for Delly and Pindel in their own benchmark where they only simulate deletions, tandem duplications and inversions. As one conclusion from these observations, we can say that especially deletions should be treated with care and be cross-examined in case they are part of

Table 1. Sensitivity (S) and PPV for variants ($\geq$30 bp, $\leq$500 bp) simulated onto chr22, shown for different coverages

	Coverage				Insert size		Read length	Large SVs
Tool	5	10	15	30	600,60	1000,100	150 bp	$\geq$500, $\leq$5000 bp
	S/PPV	S/PPV	S/PPV	S/PPV	S/PPV	S/PPV	S/PPV	S/PPV
Single BP								
Gustaf (se)	0.988/0.955	**0.961**/0.971	**0.988**/0.966	0.999/0.921	**0.988**/0.966	**0.988**/0.966	**0.996**/0.963	0.991/0.948
Gustaf (pe)	**0.993**/0.946	0.948/0.961	0.986/0.962	**1.000**/0.929	0.983/0.962	0.983/0.965	0.993/0.966	0.989/0.966
Delly	0.627/0.578	0.706/0.611	0.715/0.603	0.718/0.578	0.374/0.353	0.191/0.266	0.430/0.414	**0.994**/0.913
Pindel	0.349/**0.992**	0.717/**0.996**	0.837/**0.997**	0.928/**0.981**	0.829/**0.997**	0.830/**0.997**	0.626/**0.987**	0.899/**0.991**
Deletion								
Gustaf (se)	0.938/0.804	0.792/0.864	**0.938**/0.849	**1.000**/0.774	**0.938**/0.849	**0.938**/0.849	**1.000**/**0.923**	**1.000**/**0.920**
Gustaf (pe)	**0.979**/**0.855**	**0.833**/**0.909**	**0.938**/**0.865**	**1.000**/**0.828**	**0.938**/**0.882**	0.917/**0.880**	0.979/0.870	0.957/0.846
Delly	0.583/0.173	0.688/0.205	0.688/0.204	0.646/0.190	0.583/0.204	0.438/0.231	0.458/0.172	**1.000**/0.187
Pindel	0.208/0.164	0.625/0.200	0.833/0.208	0.958/0.198	0.729/0.183	0.812/0.197	0.562/0.205	0.696/0.163
Inversion								
Gustaf (se)	**0.978**/0.968	**0.978**/0.978	**0.978**/0.978	**1.000**/0.920	0.978/0.978	**0.978**/**0.978**	**1.000**/0.968	0.977/0.935
Gustaf (pe)	**0.978**/0.978	0.967/0.967	**0.978**/0.978	**1.000**/0.929	**0.989**/0.978	**0.978**/0.947	**1.000**/0.968	**1.000**/0.978
Delly	0.707/0.596	0.783/0.610	0.772/0.597	0.793/0.598	0.293/0.172	0.011/0.007	0.370/0.239	**1.000**/0.880
Pindel	0.467/**1.000**	0.696/**1.000**	0.783/**0.986**	0.880/**0.976**	0.739/**0.986**	0.707/0.970	0.576/**1.000**	0.955/**1.000**
Imprecise duplication								
Gustaf (se)	0.990/**0.866**	**0.959**/**0.887**	0.990/0.890	**1.000**/0.790	**0.990**/**0.890**	**0.990**/**0.890**	**0.980**/**0.881**	**1.000**/0.796
Gustaf (pe)	**1.000**/0.838	0.939/0.860	**1.000**/**0.891**	**1.000**/**0.803**	0.969/0.856	**0.990**/0.866	**0.980**/0.873	**1.000**/**0.896**
Delly	0.724/0.568	0.806/0.560	0.806/0.534	0.786/0.513	0.245/0.273	0.000/0.000	0.388/0.384	**1.000**/0.589
Pindel	0.265/0.788	0.704/0.726	0.786/0.694	0.898/0.667	0.827/0.692	0.806/0.664	0.551/0.659	0.930/0.645
Duplication								
Gustaf (se)	0.980/**0.941**	**0.939**/**0.920**	0.980/**0.941**	**1.000**/**0.867**	**0.980**/**0.941**	0.980/**0.941**	**0.969**/0.905	**0.977**/0.857
Gustaf (pe)	**0.990**/0.874	0.918/0.882	**0.990**/0.907	**1.000**/0.860	0.949/0.903	**0.990**/0.907	**0.969**/**0.922**	**0.977**/**0.894**
Translocation								
Gustaf (se)	**0.980**/**0.961**	**0.940**/**0.940**	**0.980**/**0.961**	0.960/**0.889**	**0.980**/**0.961**	**0.980**/**0.961**	**0.980**/0.925	**0.929**/**0.963**
Gustaf (pe)	0.920/0.885	0.880/0.898	0.920/0.902	**0.980**/0.875	0.920/0.902	0.920/0.920	0.960/**0.941**	**0.929**/0.897

Note: Bold values denote best sensitivity/precision values per variant category in that column. Variants $\geq$500 bp are shown separately (last column). Read length is 100 bp and mean insert size 300, except in the 'Insert Size' and 'Read Length' columns where these parameters are varied separately. The 'Single Breakpoint (BP)' category measures how well the tools can identify individual breakpoints relative to the reference genome with nucleotide precision. Predicted breakpoints are allowed to vary by up to 10 bp from simulated breakpoints. For all other categories, the predicted variant is required to have at least 80% reciprocal overlap with the simulated variant.

more complex variants. Maybe this observation even raises an issue with already annotated and not cross-validated deletions.

Inversions are generally recovered well by all tools with Gustaf always having the highest sensitivity of 97–100% with very high PPVs (92–97.8%), and Pindel usually having the highest precision of 97–100%, although with comparatively low sensitivity (46–88%) depending on the coverage. Delly has robust sensitivities (70–79.3%) and PPVs ($\sim$60%) over the tested coverages, but these values decrease with increasing insert size or read length (sensitivities <40%, PPVs <25%).

For high coverage of $\geq$ 15, Gustaf always has the highest sensitivity and precision for duplications, even when considering the precise duplications. Pindel can recover almost 90% of the duplications as an imprecise duplication, Delly recovers 80.6%. The precision for both tools, however, is <78%.

None of the duplications Delly found for an insert size of $(\mu, \sigma) = (1000, 100)$ could be confirmed by the set. Because the recovery rate for duplications is high for Delly given sufficient coverage and smaller insert sizes, this must be an artifact for this particular dataset given the high insert size.

Gustaf can recover over 94% of the small translocations over the whole tested parameter range with a precision value of almost always over 90%. Gustaf yields the best sensitivity (98%) and precision (94.2%) for the default setting with coverage of 15 and insert size $(\mu, \sigma) = (300, 30)$. On the set of small SVs, Gustaf almost always outperforms Delly and Pindel having equally high sensitivities and precisions. In addition, Gustaf called the dispersed duplications and translocations with high sensitivity and PPVs.

Gustaf can still compete for the large variant set of $500 \leq x \leq 5000$ bp SVs with sensitivity and PPV rates usually over 90%, showing Gustaf's ability to also handle larger SVs well. Compared with the small SV set, results for Pindel and Delly improve in terms of sensitivity and precision. Delly is geared toward larger SVs > 300 bp and has generally much higher sensitivity values and PPVs, having the highest sensitivity (100%) together with Gustaf for single breakpoints, deletions and inversions but with Gustaf having a higher PPV. If we include the pseudodeletions again for validation, both Delly and Pindel here reach a precision of up to 98% (data not shown). Pindel recovers 95.5% of the inversions with full precision (100%).

Delly and Gustaf recover all duplications when evaluated as imprecise where Pindel recovers 93.0% but with Gustaf having a much higher PPV (89.6%) than Delly (58.9%) and Pindel (64.5%). When evaluating duplications including the actual length, Gustaf still recovers 97.7% while keeping a high PPV of 85.7%. Gustaf recovers 92.9% of the large translocations with high precision (96.3%).

In summary, Gustaf compares favorably with Delly and Pindel. In addition, it is, to our knowledge, the only tool that can call dispersed duplications and translocations including their length. Those two types of SVs are detected with high sensitivity and precision for the tested SV size ranges and parameters, exceeding even the values for Delly and Pindel where both tools call duplications only as an imprecise type. Moreover, Delly and Pindel call the pseudodeletions of these complex variants resulting in a generally low precision for deletions. Considering this benchmark set, Gustaf is well suited for small and large SVs independent of the coverage or read length.

4 CONCLUSION

Compared with other state-of-the-art split-read-based methods, Gustaf improves detection of small SVs up to 500 bp, including the NGS twilight zone (SVs from 30 to 100 bp). For larger SVs from 500 to 5000 bp, Gustaf's results are comparable. On our simulated dataset, Gustaf consistently gives good results on the tested ranges of coverage, fragment size distribution and read length, with PPV and sensitivity mostly >90%.

One of Gustaf's unique strengths is its ability to detect SVs that are hard to classify including dispersed duplications and translocations with exact breakpoints. On our high coverage simulated dataset, Gustaf recovered up to 100% of the dispersed duplications and 98% of the translocations, both with high specificity.

Our approach is flexible in that it allows multiple splits per read. This feature will gain importance with increasing read lengths or when using Gustaf for mapping contigs. That flexibility even allows an application in mapping RNA-seq reads, which may span multiple exons [for preliminary evaluation, see Trappe (2012)].

Performing a local alignment search over a whole reference genome can get computationally expensive depending on the genome size and number of reads. However, local alignment computation can be easily parallelized and can furthermore be run independently of the core of the Gustaf algorithm.

In summary, the benefit of Gustaf is its generic multi-split-mapping approach, which makes it flexible and versatile in terms of SV types and sizes, and the length, protocol and technology of reads.

Funding: K.T. was funded by the DFG SPP1307 grants RE-1712/3-1 and RE-1712/3-2 during algorithm development and initial manuscript preparation and is currently funded by DFG grant RE3474/2-1.

Conflict of interest: none declared.

REFERENCES

Abyzov,A. and Gerstein,M. (2011) AGE: defining breakpoints of genomic structural variants at single-nucleotide resolution, through optimal alignments with gap excision. *Bioinformatics*, **27**, 595–603.

Abyzov,A. *et al.* (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. *et al.* (2011) Genome structural variation discovery and genotyping. *Nat. Rev. Genet.*, **12**, 363–376.

Brudno,M. *et al.* (2003) Glocal alignment: finding rearrangements during alignment. *Bioinformatics*, **19** (Suppl 1), i54–i62.

Chen,K. *et al.* (2009) BreakDancer: an algorithm for high-resolution mapping of genomic structural variation. *Nat. Methods*, **6**, 677–684.

Chen,K. *et al.* (2013) TIGRA: a targeted iterative graph routing assembler for breakpoint assembly. *Genome Res*, **24**, 310–317.

Döring,A. *et al.* (2008) SeqAn an efficient, generic C++ library for sequence analysis. *BMC Bioinformatics*, **9**, 11.

Emde,A.-K. *et al.* (2012) Detecting genomic indel variants with exact breakpoints in single- and paired-end sequencing data using SplazerS. *Bioinformatics*, **28**, 619–627.

Holtgrewe,M. (2010) Mason a read simulator for second generation sequencing data. In: *Technical Report TR-B-10-06*. Institut für Mathematik und Informatik, Freie Universität Berlin.

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

Kececioglu,J. (1993) The maximum weight trace problem in multiple sequence alignment. In: *Proceedings of the 4th Symposium on Combinatorial Pattern Matching (CPM)* Padova, Italy. volume 684 of *Lecture Notes in Computer Science*. Springer-Verlag, pp. 106–119.

Kehr,B. Weese,D. and Reinert,K. (2011) STELLAR: fast and exact local alignments. *BMC Bioinformatics*, 12 (Suppl 9), S15.

Li,H. and Durbin,R. (2010) Fast and accurate long-read alignment with Burrows-Wheeler transform. *Bioinformatics*, **26**, 589–595.

Maher,C.A. *et al.* (2009) Transcriptome sequencing to detect gene fusions in cancer. *Nature*, **458**, 91–101.

Marschall,T. *et al.* (2012) CLEVER: clique-enumerating variant finder. *Bioinformatics*, **28**, 2875–2882.

Marschall,T. *et al.* (2013) MATE-CLEVER: mendelian-inheritance-aware discovery and genotyping of midsize and long indels. *Bioinformatics*, **29**, 3143–3150.

Onishi-Seebacher,M. and Korbel,J.O. (2011) Challenges in studying genomic structural variant formation mechanisms: the short-read dilemma and beyond. *Bioessays*, **33**, 840–850.

Paten,B. *et al.* (2008) Enredo and Pecan: genome-wide mammalian consistency-based multiple alignment with paralogs. *Genome Res.*, **18**, 1814–1828.

Pevzner,P.A. *et al.* (2004) *De novo* repeat classification and fragment assembly. *Genome Res.*, **14**, 1786–1796.

Rasmussen,K.R. *et al.* (2006) Efficient q-gram filters for finding all epsilon-matches over a given length. *J. Comput. Biol.*, **13**, 296–308.

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

Trappe,K. (2012) *Multi-Split-Mapping of NGS reads for variant detection.* Master's thesis, Department of Computer Science, Freie Universität Berlin, Berlin, Germany.

Tuzun,E. *et al.* (2005) Fine-scale structural variation of the human genome. *Nat. Genet.*, **37**, 727–732.

Xi,R. *et al.* (2011) Copy number variation detection in whole-genome sequencing data using the bayesian information criterion. *Proc. Natl Acad. Sci. USA*, **108**, E1128–E1136.

Ye,K. *et al.* (2009) Pindel: 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. *et al.* (2009) Sensitive and accurate detection of copy number variants using read depth of coverage. *Genome Res.*, **19**, 1586–1592.