User Guide for GASV and GASVPro (Version 2.0)

Contents

1 Introduction **2**
- 1.1 Summary . . . 2
- 1.2 Requirements . . . 2
- 1.3 GASV Workflow . . . 3
- 1.4 GASV Algorithm . . . 3
- 1.5 Installation . . . 4

2 Quickstart Guide to GASV **4**
- 2.1 Using BAMToGASV to Preprocess BAM files . . . 4
- 2.2 Run GASV to Compute Structural Variants . . . 5
- 2.3 Useful GASV Examples . . . 6

3 BAMToGASV **7**
- 3.1 Requirements . . . 7
- 3.2 Command Line Options . . . 7
- 3.3 Output of BAMToGASV . . . 9

4 GASV **10**
- 4.1 GASV Command Line Options . . . 10
- 4.2 Input File Formats . . . 11
  - 4.2.1 Paired-Read Mapping (PR) File Format . . . 11
  - 4.2.2 Batch File Format (*.in files) . . . 12
- 4.3 Structural Variation Predictions . . . 12
  - 4.3.1 Structural Variant Types . . . 12
  - 4.3.2 Standard Clusters File Format . . . 14
  - 4.3.3 Additional Output Options . . . 14

5 GASVPro **16**
- 5.1 GASVPro Parameters . . . 16
  - 5.1.1 Input Files . . . 16
  - 5.1.2 Probability Model Parameters . . . 16
  - 5.1.3 Formatting Output . . . 17
- 5.2 Running GASVPro . . . 17
- 5.3 GASVPro Output . . . 18

1 Introduction

GASV is software to identify structural variants from paired-end mapping data.

If you use GASV in your research, please cite:

S. Sindi, E. Helman, A. Bashir, B.J. Raphael. A Geometric Approach for Classification and Comparison of Structural Variants. Joint: 17th Annual International Conference on Intelligent Systems in Molecular Biology and 8th Annual International European Conference on Computational Biology (ISMB/ECCB 09). Bioinformatics 2009 25(12):i222-i230. doi:10.1093/bioinformatics/btp208

This product includes software developed by the Solution Engineering, Inc. (`http://www.seisw.com/`).

Contact: Benjamin Raphael at `gasv@cs.brown.edu`

Version: 2.0
Version Date: May 30 2012

GASVPro is a probabilistic version of the GASV algorithm, combining read depth information with discordant paired-read mappings into a single model.

If you use GASVPro in your research, please cite:

S. Sindi, S. Onal, L. Peng, H. Wu, B.J. Raphael. An Integrative Probabilistic Model for Identification of Structural Variation in Sequencing Data. Genome Biology 2012 13(3):R22. doi:10.1186/gb-2012-13-3-r22.

This product contains software developed by Erik Garrison (`http://github.com/ekg`), see license.

Version: 1.0
Version Date: June 21 2012

1.1 Summary

The GASV software utilizes a geometric algorithm to predict structural variation (SV) from a set of discordant mappings of paired-reads (PRs).

1.2 Requirements

Requires: Perl, Java 5 (or higher), *nix OS,
Tested with: Perl v5.10.0, Java 6, Linux 2.6.26

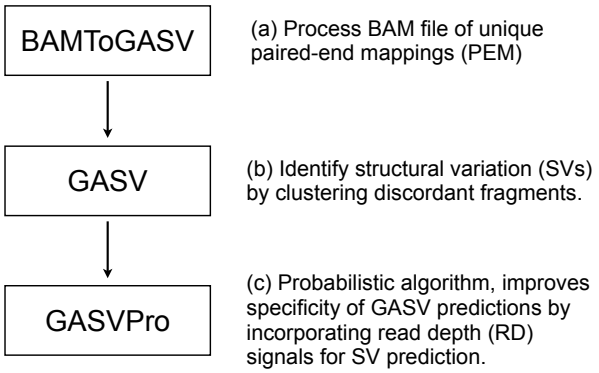


Figure 1: **GASV Workflow**

Building GASV from the source requires ant, available at: http://ant.apache.org/

1.3 GASV Workflow

The primary GASV workflow is an analysis pipeline for SV detection from a set of paired-end mappings (PEM) given in the form of a BAM file. In most workflows the BAM file is pre-processed for analysis with GASV using BAMToGASV (Figure 1(a)) followed by analysis with GASV (Figure 1(b)) to determine a set of potential SVs genome wide. Further downstream analysis of GASV predictions may be conducted with GASVPro, a probabilistic algorithm which integrates read-depth (RD) to improve specificity by eliminating false-positive predictions.

1.4 GASV Algorithm

The GASV algorithm reports structural variant events defined by clusters of discordantly mapped fragments and explicitly represents uncertainty variant endpoints. We briefly overview the approach used in GASV; for a more detailed discussion of the GASV algorithm, refer to Sindi et al. (2009).

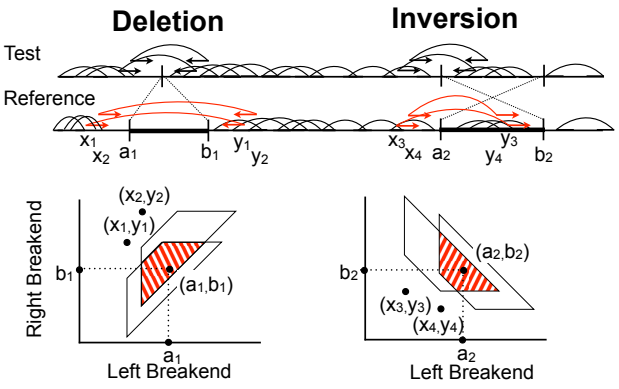


Figure 2: **GASV Algorithm of Structural Variation Detection from Paired-End Sequencing.** (Top) Fragments (black arches) from a test genome are sequenced from both ends and the resulting paired reads are mapped to a reference genome. Fragments containing the breakpoint of a structural variant (black arches with arrows) have a discordant mapping (red). (Bottom) GASV represents the uncertainty in breakpoint location as a breakend polygon (black trapezoids). GASV clusters fragments by computing polygon intersections (shaded red trapezoid) which correspond to fragments supporting the same variant.

A discordant mapping indicates a structural variant in the test genome defined by a novel adjacency (a, b), where positions a and b are adjacent in the test genome, but not in the reference genome (see Figure 2). A single fragment alone does not uniquely specify the pair of breakends (a, b) defining the rearrangement, but rather defines uncertainty in the location of the breakends. Formally, if we assume that a discordant fragment corresponds to exactly one structural variant, then the mapped locations, x and y, of the fragment endpoints (without loss of generality we restrict $x < y$), and the breakends a and b satisfy

$$L_{\min} \leq \text{sign}(x)(a - x) + \text{sign}(y)(b - y) \leq L_{\max},$$

where $\text{sign}(x)$ and $\text{sign}(y)$ are 1 if the reads align to the positive strand and have convergent orientation and -1 otherwise. Here we assume convergent orientation is when reads have opposite orientation with the left read forward and the right read reversed as in the case for Illumina sequencing technology. (Other sequencing protocols will may have different notions of convergent orientation.) The inequality (1) defines a trapezoid in the plane; discordant fragments corresponding to the same structural variant will have overlapping trapezoids and their intersection can be used to further refine the uncertainty in breakend location as in Figure 2. The final set of SV predictions from GASV corresponds to these polygonal intersections.

1.5 Installation

You may either download executable jar files from

`http://code.google.com/p/gasv/downloads/`

or build our code directly from the source.

Building GASV from the source requires ant, available at: `http://ant.apache.org/`. To install from the source check out a read-only working copy from

```
$ svn checkout http://gasv.googlecode.com/svn/trunk/ gasv-read-only
```

and install from the command line with the following command:

```
$ ./install
```

During installation the java *`.class` files for BAMToGASV and GASV are built into the `/build` subdirectory. After installation the following executable files are created:

```
bin/GASV.jar
bin/BAMToGASV.jar
bin/GASVPro
```

2 Quickstart Guide to GASV

This Quickstart Guide will demonstrate many features of GASV including determining the structural variants from paired-read sequences (PRs) whose mappings are encoded in a BAM file.

To demonstrate using BAMToGASV preprocessor in conjunction with GASV, we will use the file "Example.bam" available as a separate download from the GASV website:

```
http://code.google.com/p/gasv
```

More documentation is available for both GASV and BAMToGASV Preprocessor in sections below.

2.1 Using BAMToGASV to Preprocess BAM files

The following command creates GASV input files by considering paired-reads (PRs) from the BAM file "Example.bam" (this particular BAM file contains no library information and contains only reads from chr17, so no translocations):

```
$ java -Xms512m -Xmx2048m -jar BAMToGASV.jar Example.bam
```

As we detail further below, there are many options available for BAMToGASV. For example, use a minimum quality of 30 and determine Lmin/Lmax by 3 standard deviations away from the mean, the command is

```
$ java -Xms512m -Xmx2048m -jar BAMToGASV.jar Example.bam -MAPPING_QUALITY 30 -CUTOFF_LMINLMAX SD=3
```

If the BAM file contains multiple libraries, to process them as one set (rather than indvididually) the command is

```
$ java -Xms512m -Xmx2048m -jar BAMToGASV.jar Example.bam -LIBRARY_SEPARATED all
```

Running the above command creates the following files:

Example.bam.info
Example.bam.gasv.in
Example.bam_all.deletion
Example.bam_all.divergent
Example.bam_all.insertion
Example.bam_all.inversion
Example.bam_all.translocation

Example.info contains information about the concordant fragment length distribution. Example.gasv.in is a batch file of all PR files (except insertions) for input to the GASV program. The rest of the files are PR files in the GASV input format, separated by variant type.

2.2 Run GASV to Compute Structural Variants

The following command will report inversion, deletion and divergent structural variants using the batch input file produced by BAMToGASV. Note the default behavior of GASV is to only report variants with at least 4 supporting fragments. In addition, if GASV.jar has not been added to the PATH, then a fully qualified pathname must be used.

```
$ java -jar GASV.jar --batch Example.bam.gasv.in
```

Below is the contents of the *`.clusters` file created, `Example.bam.gasv.in.clusters`:

```
#Cluster_ID: LeftChr: LeftBreakPoint:    RightChr: RightBreakPoint:   Num PRS: Localization: Type:
c99          17       660077,660155      17        660177,660255      4        39.8          D
c106         17       685625,685815      17        685996,686186      4        100.6         D
c4552        17       34908617,34908798  17        34908870,34909051  5        68.4          D
c5262        17       40566182,40566317  17        40567335,40567462  4        68.3          IR
c6367        17       50617507,50617679  17        50617548,50617720  4        87.8          D
```

The tab separated fields indicate that there are 5 predictions, 4 deletions and 1 inversion structural variant on chromosome 17. Examining the fourth line more carefully, cluster c5262 supports an inversion with left breakpoint occurs between positions 40566182, 40566316 and the right breakpoint between positions 40567335, 40567462.

In addition, the final columns tell us:

- 4 fragments support this SV (Num PRs = 4)

- The breakpoint localization is 68. The smaller the localization the more precisely the structural variant breakpoint is determined. Refer below and to Sindi, et al. (2009) for more discussion of localization.

- This variant is a reciprocal inversion (Type = IR) Among the 4 PRs at least one is in the ++ orientation (contains the left breakpoint) and one in the -- orientation (contains the right breakpoint).

2.3 Useful GASV Examples

1. --minClusterSize <val> Report only predictions with at least a minimum number of supporting fragments. (By default, this parameter is set to 4.) For increased sensitivity, users can lower the threshold. For example to obtain all predictions supported by at least 2 mappings:

```
$ java -jar GASV.jar --minClusterSize 2 --batch Example.bam.gasv.in
```

2. --maximal In some cases a set of overlapping fragments may correspond to multiple breakpoints. The "--maximal" option will output all maximal breakpoint events.

3. --maxPairedEndsPerWin <val> The speed at which GASV runs corresponds to the number of fragments in a particular genomic region. Often some genomic regions (such as centromeres) will have a high density of discordant fragments. GASV's running time can be significantly increased by skipping such high density regions with the "--maxPairedEndsPerWin" option.

4. --output <val> As we detail below, there are three different output formats that can be specified for `*.clusters` file. The reads format, "--output reads" will list the names of the reads in a cluster.

```
$ java -jar GASV.jar --output reads --batch Example.bam.gasv.in
```

3 BAMToGASV

BAMToGASV generates GASV formatted input files from the SAM/BAM alignment format. SAM files are also accepted as input, but for clarity this manual will refer to BAM files only. Picard, a Java tool for processing BAM files, is packaged in the JAR file. Running BAMToGASV with no arguments produces basic usage information.

3.1 Requirements

BAMToGASV is currently designed to process BAM files where each read has at most one reported mapping.

Java 6 is required to run BAMtoGASV. We suggest initializing the Java heap size using the `-Xms512m -Xmx2048m` options, which requires running on a machine with more than 2G of memory. The BAM file must containing pairing information in the flags - if it does not, then Picard's `fixmate` function is called to make a new BAM file with pairing information and then the program exits.

3.2 Command Line Options

```
java -Xms512m -Xmx2048m -jar BAMToGASV.jar <BAMFile> [Options]
```

-LIBRARY_SEPARATED <String> (Default: sep)
Determines whether or not to consider multiple libraries separately

sep Produce a set of output files for each library.

all Produce a single set of output files for all libraries.

If no libary information is provided in the BAM file header, it will default to the 'all' setting. In the case where no library info is found or 'all' is passed as -LIBRARY_SEPARATED, the word "all" is used as the LIBRARY_ID in the output file names.

-OUTPUT_PREFIX <String> (Default: BAM filename)
The prefix for your output files.

-CUTOFF_LMINLMAX <String> (Default: PCT=99%)
Specifies lower and upper bounds on the fragment distribution.

PCT=X% Take the quantile on the top/bottom X percent.

SD=X Take the standard deviation above/below the mean.

EXACT=X,Y Set Lmin to X and Lmax to Y.

FILE=fname File of the form '<libname> <CUTOFF_LMINLMAX>' for using different cutoffs for individual libraries.
EXAMPLE:
library_id1 EXACT=123,456
library_id2 SD=3
library_id3 PCT=90%

In the case that the computed Lmax is smaller than twice the read length (which is the minimum fragment length), Lmax is reset to be twice the read length.

-USE_NUMBER_READS <Integer> (Default: 500000)

The number of fragments in the BAM file to use in computing Lmax and Lmin. Note that a relatively small number of fragments will be sufficient to get a good estimate of Lmin and Lmax. Depending on the size of the BAM file and running time constraints between 500,000 or 1,000,000 would be sufficient for most applications. If EXACT=X,Y is specified for the CUTOFF_LMINLMAX argument, no fragments will be used, and USE_NUMBER_READS argument is ignored.

-CHROMOSOME_NAMING_FILE <String> (Default: none)

File of the form '<ChrName>\t<Integer>' for specifying non-default chromosome namings. This is an optional parameter, you do not need this option if you are using the default chromosome naming references (a number, X, Y, chr+a number, chrX, and chrY). The second column contains unique integer IDs for each non-default chromosome name.
EXAMPLE:
Ca21chr1 1
Ca21chr2 2
Ca21-mtDNA 9

-PROPER_LENGTH <Integer> (Default: 10000)

Ignore PRs with separation larger than PROPER_LENGTH when calculating Lmin and Lmax. Extreme outliers (PRs with mapped length >10Mb) can cause huge standard deviation values which will produce large values for Lmax in with the SD option. If you use EXACT=X,Y or PCT=X% as CUTOFF_LMINLMAX, this option will not be considered.

-PLATFORM <String> (Default: Illumina)

Paired Illumina reads are sequenced from different strands of the fragment, while paired SOLiD reads are sequenced from the same strand of the fragment.

Illumina Reads are sequenced with an Illumina-like platform.

SOLiD Reads are sequenced with a SOLiD-like platform.

If the BAM header information reports a platform different from the -PLATFORM option, a warning is emitted but the program continues with the -PLATFORM option.

-WRITE_CONCORDANT <Boolean> (Default: False)

Writes a file of concordant PRs of the form '<chr> <start> <end>' to an OUTPUT_PREFIX.LIBRARY_ID.concordant file. WARNING - this may be a very large file. In a future release this will be a compressed binary file.

-WRITE_LOWQ <Boolean> (Default: False)

Writes a file of low-quality pairs (pairs where at least on read has quality below -MAPPING_QUALITY). WARNING - this may be a very large file.

-VALIDATION_STRINGENCY <String> (Default: silent)

Picard performs internal testing of BAM records as they are read into the BAMtoGASV program. The VALIDATION_STRINGENCY option determines how stringent this internal testing is.

silent Read SAM records without any validation.

lenient Read SAM records and emit a warning when a record is not formatted properly.

strict Read SAM records and die when a record is not formatted properly.

Picard's default setting is Strict; however, this assumes a very well-formatted BAM file from Illumina sequencing. If the output files are empty or not what you expect, we suggest running with VALIDATION_STRINGENCY set to lenient or strict to determine improperly-formatted BAM records.

3.3 Output of BAMToGASV

The output of BAMToGASV is a series of discordant PR files (see description below for GASV), an info file with summary statistics on the various libraries, and a batch input file for GASV.

Discordant fragments are grouped into output files according to the type of structural variant they indicate: deletion, insertion, inversion and translocation. Divergent fragments indicate a inter-chromosomal rearrangement that is not a deletion, inversion, or insertion.

OUTPUT_PREFIX(_LIBRARY_ID).deletion

OUTPUT_PREFIX(_LIBRARY_ID).divergent

OUTPUT_PREFIX(_LIBRARY_ID).insertion

OUTPUT_PREFIX(_LIBRARY_ID).inversion

OUTPUT_PREFIX(_LIBRARY_ID).translocation

OUTPUT_PREFIX.info

OUTPUT_PREFIX.gasv.in

In addition to a GASV formatted input file, GASV requires the minimum/maximum allowable fragment length (Lmin/Lmax) for concordant pairs. The values of Lmin and Lmax are automatically computed by the BAMToGASV. The calculated values of Lmin and Lmax are output to stdout and written to OUTPUT_PREFIX.info. In the Quickstart guide, the contents of Example.bam.info are:

```
LibraryName     Lmin    Lmax
all             153     246
```

The batch input file is written to OUTPUT_PREFIX.gasv.in. It is a GASV input format that is described below. All PR files except insertions are written to this file; GASV does not currently support insertions.

Note: The output produced from BAMToGASV.jar may slightly differ from output produced by earlier versions. Refer to RELEASE_NOTES.txt for more information.

4 GASV

GASV.jar runs the GASV algorithm on a set of discordant fragment mappings and reports a set of SV predictions. GASV.jar can be run directly on the output of BAMToGASV, using a "gasv.in" file:

```
java -jar GASV.jar <InputFile>.gasv.in
```

4.1 GASV Command Line Options

There are many options to GASV which will allow users to customize their own SV analysis pipeline. Running GASV.jar with no arguments produces basic usage information.

```
java -jar GASV.jar

Program: GASV (Geometric Analysis for Structural Variation)
Version: 2.0

Usage:   java -jar GASV.jar [options] <InputFile>

         InputFile: PR file
           java -jar GASV.jar [options] <InputFile>

         InputFile: Batch File
           java -jar GASV.jar [options] --batch <InputFile>

Output:    <InputFile>.clusters
Default:   Reports all SVs supported by at least 4 mappings.
```

--help Outputs detailed information on options to GASV.

--nohead Output file will not have a header.

--outputdir <dir> Directory into which output file will be placed.

--verbose Prints extra info to standard output.

--debug Run program in debug mode

--fast Makes GASV run faster, but risks OutOfMemory errors

--batch Input file(s) are GASV Batch Files (*.in files listing PR files), rather than the PR file(s) themselves. The --lmin and --lmax options are ignored if --batch is specified.

--lmin <int> Specify integer value to use for Lmin variable. Default value used otherwise. Ignored if the –batch option is specified.

--lmax <int> Specify integer value to use for Lmax variable. Default value used otherwise. Ignored if the –batch option is specified.

--minClusterSize <int> Only those clusters with at least <int> supporting PRs are reported.

--maxClusterSize <int> Only those clusters with at most <int> supporting PRs are reported.

--maxCliqueSize <int> For clusters with greater than<int> paired end sequences, calculation of maximal sub clusters is performed. Also only at most <int> PR names will be output per cluster (though numPR column will still reflect real number of PRs). Localization for such clusters is always -1.

--maxPairedEndsPerWin <int> Does not apply if --fast otion used. If more than <int> paired end sequences are found in the current window, any extra paired end sequences are discarded.

--maximal For each connected component without a common intersection (localization reported as -1), the maximal sub-clusters (each with a common intersection) will be reported.

--numChrom <int> Specify number of chromosomes in the genome. Default is 24.

--output <mode> Specify the GASV cluster output mode (see Output File Formats).

1. standard: Default output mode
2. reads: Default mode and names of supporting fragments
3. regions :Detailed mode with region coordinates and fragment names

--nonreciprocal Only PRs with exact matching orientations will be clustered together. E.g. +/+ and -/- inversion PRs will be segregated into separate clusters. (See Figures 3 and 4.)

4.2 Input File Formats

GASV accepts two types of input files; either a set of discordant mappings or a batch file listing multiple files of discordant mappings. If you are using a BAM file, these files will be automatically created with BAMToGASV.jar. Although, users may supply their own files.

4.2.1 Paired-Read Mapping (PR) File Format

Paired-read (PR) files are used for input of mapped paired-end sequences. Each line in the file specifies a paired-mapping. The 9 columns are tab-separated in the following format:

Name: A string with a unique PR identifier.

Left Chromosome: The chromosome where the left read mapped. (Acceptable formats: CHR1, chr1, 1.)

Left Mapped Start: The start coordinate in the reference of the left read mapping.

Left Mapped End: The end coordinate in the reference of the left read mapping.

Left End Orientation: The orientation of the left read. (Acceptable formats: include +/-, Plus/Minus)

Right Chromosome: Chromosome where the right read mapped.

Right Mapped Start: The start coordinate in the reference of the right read mapping.

Right Mapped End: The end coordinate in the reference of the right read mapping.

Right End Orientation: The orientation of the right read.

IMPORTANT PRE-PROCESSING STEP

PR files must be in sorted order prior to running GASV. If your PR files were created with BAMToGASV.jar they will already be sorted. Otherwise before running GASV you should run the sorting script:

Example: ./scripts/sortPR.bash PRFileToSort

Note: Since mapped start and end are in terms of the reference sequence, Start < End. (See Table 1 for example.)

PRA	1	146590306	146591106	+	1	146630713	146631513	+
PRB	2	46590283	46591083	+	1	146630714	146631514	+
PRC	2	46590283	46591083	+	1	156630714	156631514	+
PRD	6	146586311	146587111	+	10	146633908	146634708	+

Table 1: Example PR file format

4.2.2 Batch File Format (*.in files)

The GASV Batch input files provide a list of the PR file(s) to be processed. As mentioned above, each of the PR files must be sorted by chromosome and genomic coordinates. The BAMToGASV program does this automatically. If your data comes from elsewhere, the script ./scripts/sortPR.bash should be used to sort files prior to running GASV.

For PR files, each line gives the name of a PR file, the word "PR" to indicate the file type, and the minimum (Lmin) and maximum (Lmax) fragment length for paired-read mappings in the file. Columns are tab-separated in the following format:

FILE PR Lmin Lmax

4.3 Structural Variation Predictions

GASV outputs a clusters file named <InputFile>.clusters that lists all SV predictions with supporting information. There are three possible output formats for clusters files (detailed below). We first describe the different structural variant types reported by GASV, and provide illustrative figures for translocations (Figure 3) and inversions (Figure 4).

4.3.1 Structural Variant Types

GASV Structural Variant Types

D = Deletion

I = Inversion (see Figure 4)

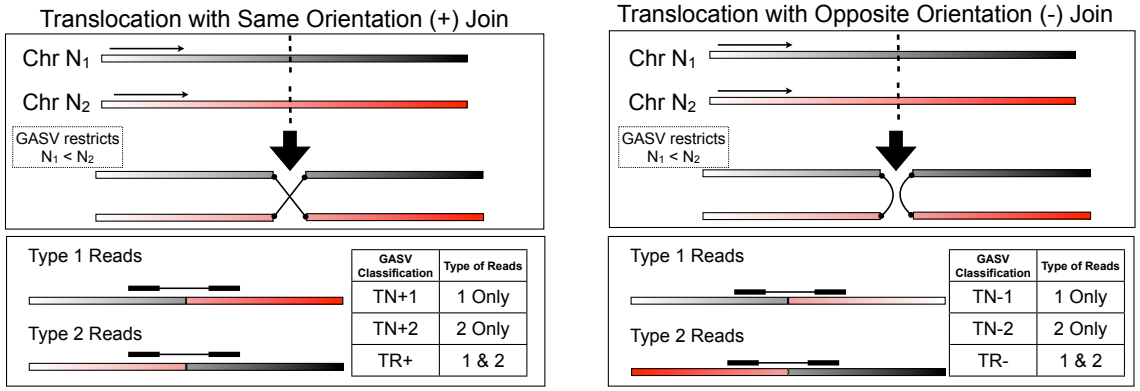
- **IR** = Reciprocal Inversion(both ++ and --)
- **I+** = Inversion (++ side only)
- **I-** = Inversion (-- side only)

V = Divergent

DV = Combination of divergent and deletion discordant PRs.

T = Translocation; (see below and Figure 3)

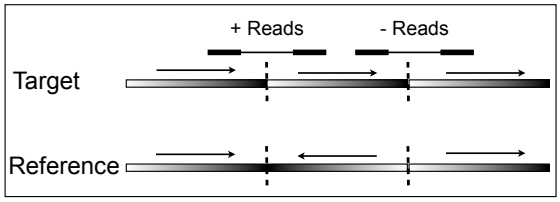
- **TR** = Reciprocal Translocation
- **TN** = Nonreciprocal



(a) Translocations: T+

(b) Translocations: T-

Figure 3: **GASV Algorithm Translocation Definition.**



GASV Classification	Type of Reads
I+	+ Only
I-	- Only
IR	Both + & -

Figure 4: **GASV Algorithm Inversion Definition.**

The GASV algorithm distinguishes between several types of translocations based on the orientations of the respective chromosomes at the join (Figure 3); opposite (-) or same (+); and which resulting chromosome in the test genome was observed; type 1 or type 2. In the top panel of Figure 3, we illustrate the break and join event creating the translocation. While in the bottom panel, we show the resulting chromosomes present in the test genome, from which segments were sampled. As default mode, GASV will attempt to merge predictions from both type 1

and type 2 fragments to identify reciprocal translocation events (TR+ or TR-). If the --noreciprocal flag has been set, clusters from type 1 and type 2 fragments will always be reported separately (TN-1, TN-2, TN+1, TN+2). A similar criteria applies to inversion structural variants (see Figure 4).

4.3.2 Standard Clusters File Format

Columns are tab-separated in the following format:

#Cluster_ID: Unique text identifier for the cluster c<cluster_number>. In --maximal mode, the sub-clusters are named as in: c[MajorClusterNumber].[MinorSubclusterNumber]

LeftChr: Chromosome containing the left breakpoint.

LeftBreakPoint: Interval containing the left breakpoint.

RightChr: Chromosome containing the right breakpoint. (Note that except for translocations LeftChr = RightChr).

RightBreakPoint: Interval containing the right breakpoint.

Num PRS: Number of PRs that support this variant.

Localization: Square root of the breakpoint region (polygon of intersection). The smaller the localization, the more precisely the breakpoint can be determined. See below for more information on localization.

Type: The type of structural variant indicated (See Section 4.3.1).

Example Clusters File:

#Cluster_ID:	LeftChr:	LeftBreakPoint:	RightChr:	RightBreakPoint:	Num PRS:	Localization:	Type:
c521	17	40566182,40566330	17	40567335,40567463	4	86.4	IR

Table 2: GASV Default Clusters Format

Note: If the localization is equal to -1, this means that while a set of PRs overlap, there is not a single structural variant that can explain the data. Running GASV with the "--maximal" flag will output all maximal structural variant predictions from such a cluster.

For examples and more information on maximality, and localization refer to Sindi, et al. (2009).

4.3.3 Additional Output Options

GASV provides two additional formatting options for the clusters file. These options are accessible by using the --output <val> option.

reads: The same as "standard" but has an additional column with a comma separated list of the names of names of PRs in each cluster.

regions: Highly detailed output that specifying the coordinates of the breakpoint polygon itself. The tab separated columns are:

#Cluster_ID:

Num PRS:

Localization:

Type:

List of PRS:

LeftChr:

RightChr:

Boundary Points:

The boundary points are given in the form: $x1, y1, x2, y2, \ldots$ where the points (x_i, y_i) correspond to the boundary of the breakpoint polygon.

Note: The output produced from GASV.jar may differ from output produced by earlier versions of GASV. Refer to RELEASE_NOTES.txt for more information.

5 GASVPro

5.1 Running GASVPro

GASVPro is a probabilistic version of our original GASV algorithm, designed to be run after GASV. GASVPro combines read depth information along with discordant paired-read mappings into a single probabilistic model.

GASVPro can be run with a PARAMETERS file (put code). There are many different options available, detailed below; however, for ease of use a PARAMETERS file is created with the execution of BAMToGASV when **-WRITE_CONCORDANT** option set to **True** (see Example above). Users are encouraged to begin with this default parameters file and to add/modify options as needed by their specific application.

5.2 GASVPro Parameters

Put the full parameters file; mention that you can comment things in the file and that in the figure shown we separate the required from the optional parameters.

5.2.1 Input Files

Because GASVPro was designed to be run after BAMtoGASV and GASV, the input files to GASVPro will appear familiar to those who have experience with the GASV pipeline. There are also a number of parameters needed by the probabilistic model of GASVPro.

Clusters File: must be in GASV Regions Clusters Format (see section 3.2).

- **NOTE:** The `--output regions` parameter must be passed to GASV to produce a clusters file of the correct format.
- **NOTE:** The current version of GASVPro does not support divergent variants. They can be included in cluster files but will be ignored.

Concordant File: must be in BAMtoGASV default output format (<chr> <start> <end>).

- The BAMtoGASV preprocessor produces these files with the **-WRITE_CONCORDANT** option set to **True**. See Section 3.2 for details.
- **NOTE:** If users provide their own concordant file it must follow the same format and **must** be sorted numerically by start coordinate.

Uniqueness File: Because GASVPro's model is based on read depth, users may want to incorporate biases in the expected coverage (such as GC content or local mapability). A uniqueness file is a **single file** of format <chr> <start> <end> <unique>. (See below for more information.) The uniqueness file **must** be sorted numerically by start coordinate.

5.2.2 Probability Model Parameters

Average Fragment Length (Lavg) & Average Read Length (ReadLen): *follow from sequencing libraries. In the event of multiple libraries, use the average over all datasets.*

Lambda & Perr: *parameters for probabilistic model, see manuscript.*

Minimum Fragment Size (Limit): this value is the smallest variant that GASVPro will use concordant coverage on.

Tolerance: Tolerance is used to remove regions that have abnormally high concordant coverage and, thus, can be eliminated from consideration as a structural variant. A tolerance of t will eliminate from further consideration as a potential structural variant a prediction with N concordant fragments provided $P(\geq N \mid \text{no variant}) > t$.

Max Uniqueness Value: this parameter is the value assigned to the most mappable regions in the uniqueness files.

Minimum Scaled Uniqueness: this parameter is the minimum value to which GASVPro will scale the uniqueness values for each variant. **NOTE:** When this parameter is set to 1, all regions will have the same uniqueness value.

5.2.3 Formatting Output

Max Chromosome Number: this is the highest *numerical value label* of all the chromosomes in the Clusters File.

Translocations Mode: running GASVPro in translocations mode will enable the processing of translocation clusters along with deletions and inversions (see page 13 for accepted translocation types). **WARNING: Processing translocations takes exponentially more time than processing deletions and inversions alone.** If possible, we recommend running Translocations Only mode on datasets with large numbers of translocations and possibly running two copies of GASVPro in parallel (one for translocations, another for inversions and deletions).

Translocations Only (TransOnly): setting this value to `true` will process translocations ONLY, ignoring all other types. **WARNING: This operation will take hours.**

Likelihood Ratio Threshold (LRThreshold): Users can input a numerical value or the string `all`. GASVPro will only return clusters with a LR greater than or equal to the input value (See section 5.3 for explanation of LR). If `all` is given, all clusters will be printed.

Verbose: specifying `Y` will print information for each cluster processed.

5.3 Running GASVPro

GASVPro runs by providing a parameters file to the program. We have provided a parameters file for running GASVPro, and highly recommend its use for convenience and to ensure input accuracy. The parameters file also allows for clean record-keeping and can be annotated.

```
# A Sample GASVPro parameters file
# Annotation can appear on any line beginning with a hash mark (#).
clusterFile: <ClusterFile>
ConcordantFile: <ConcordanceFile>
UNIQUEFile: <UniquenessFile>
Lavg: <value>
ReadLen: <value>
Lambda: <value>
Perr: <value>
Limit: <value>
Tolerance: <value>
MaxChrNumber: <value>
MaxUniqueValue: <value>
MinScaledUniqueness: <value>
LRThreshold: <value/all>
Translocations: <true/false>
TransOnly: <true/false>
Verbose: Y
```

Figure 5: **A Sample GASVPro Parameters File**

We have preprogrammed default values for some of these parameters. These hard-coded values are as follows:

Perr = `0.01`
Limit = `1000`
Tolerance = `.0001`
MaxChrNumber = `100` NOTE: It is highly advisable to lower this value to reduce runtime, if possible. $More \Downarrow$
TranslocationsMode = `false`
TransOnly = `false`
LRThreshold: `all`

Finally, using a parameters file allows the user to run GASVPro with the following command:

```
$ ./GASVPro {ParametersFile}
```

5.4 GASVPro Model Overview

We provide a brief overview of the GASVPro model, a full description of our model and algorithm is available in (cite manuscript).

5.4.1 Poisson Coverage Model

What is Lambda, what is perr. How is read depth integrated

5.4.2 Uniqueness Scaling

How uniqueness scaling is done.

5.5 GASVPro Output

After successful completion GASVPro will output two files, a `.clusters` file and a `.coverage` file. Input parameters relevant to subsequent steps in the GASV pipeline will also be included in the filenames. The `.clusters` file contains the variance probabilities that GASVPro calculated. Here is an example GASVPro output, contained in the `.clusters` file:

#Cluster_ID:	NumPRS:	Localization:	Type:	···	···	···	···	Likelihood Ratio:
c19	1	212	TNR-	···	···	···	···	620.355

Table 3: An Example GASVPro Output Cluster (see Section 4.3.3 for all fields.)

Let the probability that a given cluster C *is* a variant be given by P_v and the probability that C *is not* a variant be given by P_{nv}. The Likelihood Ratio therefore is $P_v - P_{nv}$. The rest of the information in the `.clusters` file is identical to the input file.

Additionally, GASVPro will output another file which offers more information generated directly by the program. This file will have a `.coverage` extension and includes the following information:

#Cluster_ID: the cluster number.

Type: See page 13. NOTE: This GASVPro version currently does not support divergent variants.

NumDiscordants: number of discordant fragments supporting this cluster.

LogVariantProbability: the log probability that this cluster is a variant.

LogNoVariantProbability: the log probability that this cluster is not a variant.

Start: start coordinate.

End: end coordinate.

ConcordantCovLeft: concordant fragment coverage on left.

ConcordantCovRight: concordant fragment coverage on right.

MapabilityLeft: cluster uniqueness on left.

MapabilityRight: cluster uniqueness on right.

Algorithm Code: indicates which process was used to calculate these values.