A GENERIC ALGORITHM FOR SCAFFOLDING NEXT-GENERATION SEQUENCING ASSEMBLIES

User manual

February 11, 2016

Contents

Chapter 1: Introduction

The millions of short reads usually involved in *high-throughput sequencing* (HTS) are first assembled into longer fragments called contigs, which are then scaffolded, i.e. ordered and oriented using additional information, to produce even longer sequences called scaffolds. Most existing scaffolders are not suited for using information other than paired reads to perform scaffolding. They use this limited information to construct scaffolds, often preferring scaffold length to accuracy, when faced with the tradeoff.

1.1 GRASS algorithm

GRASS (stands for GeneRic ASsembly Scaffolder) is an assembly scaffolder capable of combining multiple scaffolding information sources, such as paired reads or related genomes. It offers a *Mixed-Integer Programming* (MIP) formulation of the contig scaffolding problem, which combines contig order, distance and orientation in a single optimization objective. The resulting optimization problem is solved using an Expectation-Maximization procedure and an unconstrained binary quadratic programming approximation of the original problem. Please refer to the original manuscript for details.

1.2 Components

GRASS consists of two components: `dataLinker` and `scaffoldOptimizer`. `dataLinker` is responsible for converting available scaffolding information into abstract contig links and exporting the latter for further processing by the `scaffoldOptimizer` module. Contig links provide relative orientation, order and approximate distance for contigs that they connect. This information is used by the `scaffoldOptimizer` to formulate the contig scaffolding problem in terms of mixed integer programming and solve the latter. The MIP formulation allows GRASS to consider contig order, orientation and distance simultaneously while minimizing the extent to which constraints given by the contig links are violated.

`dataLinker` can use any information source to create contig links. Current version of GRASS supports the following information sources:

- Illumina paired reads (aligned to contigs using BWA)
- 454 mate pair reads (aligned to contig using NovoAlign)
- Related genomes (contigs are aligned to genome sequences using MUMmer)

Chapter 2: Installation

2.1 Prerequisites

GRASS is written in C++ and requires GCC 4.4.6 or later to be compiled. It should also work with other compilers, however, this has not been tested. It was written for Linux operating systems (tested on Red Hat Linux).
GRASS relies on several C++ libraries and third-party software packages. These have to be installed before installing GRASS:

1. IBM ILOG CPLEX Optimizer (version $\geq$ 12.2.0.2)
2. NCBI C++ Toolkit (version $\geq$ 7.0.0), `http://www.ncbi.nlm.nih.gov/books/NBK7170/`
3. Boost C++ Libraries (version $\geq$ 1.48.0), `http://www.boost.org/`
4. BamTools C++ API (version $\geq$ 0.9.0), `http://sourceforge.net/projects/bamtools/`
5. BWA (version $\geq$ 0.5.9rc1), `http://bio-bwa.sourceforge.net/`
6. NovoAlign (version $\geq$ 2.07), `http://www.novocraft.com/`
7. SAMTools (version $\geq$ 0.1.12a), `http://samtools.sourceforge.net/`
8. MUMmer (version $\geq$ 3.07), `http://mummer.sourceforge.net/`

The first four prerequisites are required to compile GRASS, whereas the last four are invoked during runtime. Please follow software-specific instructions when installing the prerequisites. The third-party application that GRASS relies on must be reachable via the execution PATH.

2.1.1 IBM ILOG CPLEX Optimizer

To scaffold contigs GRASS splits the problem into sub-problems, which are then solved separately. Each such sub-problem requires solving a *Linear Programming* (LP) problems. The CPLEX C++ API is used to solve these linear programs. CPLEX is available at no charge for academic use (see `http://www-01.ibm.com/support/docview.wss?uid=swg21419058`).

In future versions of GRASS it is planned to switch to a more light-weight LP solver.

2.1.2 NCBI C++ Toolkit

GRASS depends on the NCBI C++ Toolkit for performing global alignment of consecutive contigs during scaffold post-processing. This is done using Hirschberg's linear space algorithm for performing sequence alignment. Source code for the NCBI C++ Toolkit can be obtained from `http://www.ncbi.nlm.nih.gov/books/NBK7170/`.

In future versions of GRASS it is planned to use a stand-alone implementation of the Hirschberg's algorithm instead of the complete NCBI Toolkit.

2.1.3 Boost C++ Libraries

Boost C++ Libraries can be obtained from `http://www.boost.org/`.

2.1.4 BamTools C++ API

BamTools API is used for processing BAM files containing read alignment. BamTools can be downloaded from `http://sourceforge.net/projects/bamtools/`.

2.1.5 BWA

Burrows-Wheeler Aligner (BWA) is used (by the `dataLinker` module of GRASS) to align Illumina sequencing reads to the contigs. The alignments are output to a temporary file in SAM format. BWA can be obtained from `http://bio-bwa.sourceforge.net/`.

2.1.6 NovoAlign

Similarly to BWA and Illumina reads, NovoAlign is used to align 454 reads. The aligner can be obtained from `http://www.novocraft.com/`.

2.1.7 SAMTools

SAMTools are used to convert alignment output from SAM to BAM file format. The latter is processed by GRASS using functions from the BamTools API. SAMTools can be obtained from `http://samtools.sourceforge.net/`.

2.1.8 MUMmer

MUMmer is used for alining contigs to reference genomes, both for creation of scaffolding links (by the `dataLinker` module) and evaluation of scaffolds (performed by breakpointCounter). MUMmer can be obtained from `http://mummer.sourceforge.net/`.

2.2 Obtaining GRASS

GRASS is freely available under the GNU GPL v3 license and can be obtained. The latest version of GRASS can be checked out from its SVN repository:

```
svn checkout http://tud-scaffolding.googlecode.com/svn/trunk/ grass
```

Alternatively GRASS source code can be downloaded from

`http://code.google.com/p/tud-scaffolding/downloads/list`

GRASS prerequisites are not distributed with it and have to obtained from their respective websites.

2.3 Compiling GRASS

GRASS comes with a simple `Makefile`, but not with a configuration script. All compiler options, include and library paths must be set manually. This can be done by changing `Makefile.config` accordingly. The things that must be changed are the include and library paths (`-I` and `-L` compiler switches). These paths must point to installation directories of the prerequisite libraries.

After the changes are complete, the source code can be compiled by running make from the source code root directory:

```
make
```

This compiles the core set GRASS tools: `breakpointCounter`, `coverageUtil`, `dataLinker` and `scaffoldOptimizer`. The corresponding executables can be found in a newly created `bin` folder.

Chapter 3: Running GRASS

The following section consider an example of scaffolding *E.coli* using Illumina paired end reads from SRR001665 (insert size 216±10 bp, 160 ×) and SRR001666 (insert size 488±18 bp, 107 ×). For the purpose of this example assembled contigs are stored in file `contigs.fa`, left and right reads are stored in files `SRR001665_1.fastq` and `SRR001665_2.fastq`, and `SRR001666_1.fastq` and `SRR001666_2.fastq` for the two libraries respectively.

3.1 Creating contig links

Contig links can be created from the available data using the `dataLinker` module. The links are created by aligning paired reads to contigs or by aligning contigs to related genome sequences, and considering the resulting alignments. Several important parameters determine minimal quality of the considered alignments:

- `-minlength <n>` sets the minimum length an alignment to consider creating a link between two contigs. This setting is used for paired read alignments and for alignments to related genome sequences. If an alignment/read is shorter that n bp, no contig links are created for it.
- `-mapq <n>` sets the minimum mapping quality of read pair alignments to consider creating a link between two contigs they align to. Mapping quality reported by the read aligner (BWA or NovoAlign is used). If a read's mapping quality is lower than n, no contig links are created for it.
- `-maxedit <n>` sets the maximum allowed edit distance in a read alignment. If a read's alignment contains more than n mismatches/indels, no contig links are created for it.
- `-maxhits <n>` sets the maximum allowed number of read alignments per read for it to be considered in contig link creating. Contig links created from reads with more than one alignment are marked as ambiguous.
- `-nooverlapdeviation <n>` sets the maximum allowed deviation from the mean insert size of a read pair under the assumption that contigs do not overlap. If the minimum required insert size between two reads aligned to different contigs required for the two contig not to overlap is more than n standard deviation away for the mean insert size, no links are created for this read pair.

The default GRASS behavior is to use strict settings for these parameters: `-minlength 30`, `-mapq 36`, `-maxedit 0`, `-maxhits 5` and disabled `-nooverlapdeviation`.

3.1.1 From paired reads

To create links from paired, each paired read library has to be specified by listing FASTQ files with it's left and right reads, mean insert size and standard deviation. For Illumina *paired end* reads it can be done by using `-illumina <left> <right> <mean> <std>` switch:

```
-illumina SRR001665_1.fastq SRR001665_2.fastq 216 10
-illumina SRR001666_1.fastq SRR001666_2.fastq 488 18
```

Similarly, switch `-454` can be used to input 454 *mate pair* reads.

GRASS can use aligned reads to estimate contig coverage. This information can optionally be used to detect repeat contigs. Data required by `scaffoldOptimizer` to estimate contig coverage can be recorded by specified the `-readcoverage filename.coverage` switch.

For the considered example the following command creates contig links of weight 1 from Illumina paired reads:

```
dataLinker -weight 1 -illumina SRR001665_1.fastq SRR001665_2.fastq 216 10 \
                     -illumina SRR001666_1.fastq SRR001666_2.fastq 488 18 \
                     -readcoverage ecoli.coverage \
                     -output ecoli.opt \
                     -bwathreads 12 contigs.fa
```

The reusable `-weight <n>` switch specifies a weight of contig links created from the information sources that are specified after it. A link of weight n is created for every suitable read pair. The creating contig links are saved into a file specified by the `-output <filename>` switch. Switches prefixed with `-bwa` control behavior of the BWA read aligner executed in the background to align reads to contigs. `-bwathreads` specifies the number of threads used by BWA.

3.1.2 From related genome sequences

GRASS can use genome sequences of (closely) related organisms to guide its scaffolding process. It does so by aligning contigs to the genome sequences and creating links for contigs based on the alignment. A related genome sequence can be added as an information source be specifying a FASTA file with the sequence(s) and a standard deviation associated with this sequence using the `-seq <reference.fa> <sigma>` switch. The standard deviation σ should be chosen such that it reflects the degree of uncertainty in the distances derived by the contig alignment. A value of $\sigma = 3000$ could be a good first guess. If, for example, *E.coli* strain DH10B genome is stored in file `dh10b.fa`, then the following command can be used to create contig links using it:

```
dataLinker -weight 70 -seq dh10b.fa 3000 \
                      -output ecoli.opt contigs.fa
```

3.2 Scaffolding linked contigs

Contig links created at the previous step can be used by the `scaffoldOptimizer` module to construct scaffolds. This module provides various options for controlling the amount of time and the number of attempts spent on optimization; the parameters for computing global sequence alignments of overlapping contigs; and options controlling contig link bundling.

By default contig link bundling is turned on (option `-bundle yes`), links are sorted prior to bundling (option `-sort yes`) and ambiguous contig links are removed (option `-ambiguous no`). These settings show good performance in practice.

When provided with a coverage depth estimate and a contig read coverage file produced by the `dataLinker` module, GRASS automatically detect repeat contigs and removed contig links incident to them. This functionality is enabled by the

```
-repeat-coverage <exp. cov.> <cov. filename>
```

switch. For the considered *E.coli* example, the following switch can be used:

```
-repeat-coverage 264 ecoli.coverage
```

It's important to filter contig links that are likely to be incorrect prior to scaffolding. For this purpose contig links with weights smaller than an erosion threshold e are removed at the start of the algorithm. The erosion threshold is set using the `-erosion <e>` switch. A default value of 5 is suitable for most scaffolding applications, but can be increased to produce more confident scaffolds or chosen to implement combinatorial contig link selection (e.g. keep a link only if it's supported by multiple information sources) in presence of multiple information sources.

The following command is sufficient for scaffolding the considered *E.coli* example:

```
scaffoldOptimizer -repeat-coverage 264 ecoli.coverage \
                -output scaffolds.fa ecoli.opt
```

It outputs constructed scaffolds into a file specified by the `-output <filename>` switch.

Chapter 4: Known issues & contact

4.1 Known issues

GRASS is continuously improved and the discovered bugs or issues are fixed as soon as possible. However, several less critical issues remain:

1. GRASS depends on a large number of libraries and application, which are time-consuming to obtain, compile and setup. Most of the large dependencies (such as CPLEX, the NCBI C++ Toolkit and Boost C++ Libraries) and waived by switching to more light-weight libraries.

2. GRASS' `Makefile` does not correctly handle source code dependencies. An occasional `make clean` may be required to ensure that all output is up to date.

We hope to resolve these issues in the next release of GRASS.

4.2 Contact information

You can report bugs or contact us with questions or suggestions via

- Via GRASS Google code project: `http://code.google.com/p/tud-scaffolding/`
- Via email: a.gritsenko[at-symbol]tudelft.nl