BIOINFORMATICS APPLICATIONS NOTE Vol. 28 no. 14 2012, pages 1937–1938
doi:10.1093/bioinformatics/bts297

*Sequence analysis*

Advance Access publication May 18, 2012

# *Gossamer* — a resource-efficient *de novo* assembler

Thomas Conway*, Jeremy Wazny, Andrew Bromage, Justin Zobel and Bryan Beresford-Smith

NICTA Victoria Research Laboratory, Department of Computing and Information Systems, The University of Melbourne, Parkville, Victoria 3010, Australia

Associate Editor: Michael Brudno

**ABSTRACT**

**Motivation:** The *de novo* assembly of short read high-throughput sequencing data poses significant computational challenges. The volume of data is huge; the reads are tiny compared to the underlying sequence, and there are significant numbers of sequencing errors. There are numerous software packages that allow users to assemble short reads, but most are either limited to relatively small genomes (e.g. bacteria) or require large computing infrastructure or employ greedy algorithms and thus often do not yield high-quality results.

**Results:** We have developed *Gossamer*, an implementation of the de Bruijn approach to assembly that requires close to the theoretical minimum of memory, but still allows efficient processing. Our results show that it is space efficient and produces high-quality assemblies.

**Availability:** *Gossamer* is available for non-commercial use from http://www.genomics.csse.unimelb.edu.au/product-gossamer.php.

**Contact:** tom.conway@nicta.com.au

Received on January 9, 2012; revised on April 5, 2012; accepted on May 15, 2012

## 1 INTRODUCTION

High-throughput sequencing technologies have enabled researchers to produce unprecedented volumes of short read data. The *de novo* assembly of such data is a core problem in bioinformatics with numerous applications in the analysis of genomes, metagenomes, and transcriptomes. There are several common approaches to the *de novo* assembly of short read data, including those based on greedy extension (Warren *et al.*, 2006), overlap layout extension (Hernandez *et al.*, 2008), and de Bruijn graphs (Chaisson *et al.*, 2009; Zerbino *et al.*, 2009). Our assembler, *Gossamer*, is an extension of a prototype based on the succinct representation of de Bruijn assembly graphs as a bitmap or set of integers (Conway and Bromage, 2011). It assembles base-space paired reads such as those from an Illumina sequencing platform.

## 2 METHODS

*Gossamer* operates in a series of explicit passes to give the user control of the assembly process. Broadly, assembly proceeds through the following phases: graph construction, graph 'cleaning' to remove spurious edges, alignment of pairs to the Eulerian super-graph, Eulerian super-path lifting, scaffolding and finally contig production. We have constructed a front-end shell script (*gossple.sh*) which invokes these from a single command line, suitable for simple assemblies.

For a given $k$, we construct the de Bruijn graph by extracting from the input all graph edges of length $\rho = k+1$, the $\rho$-mers and their reverse complements; the graph's $k$-mer nodes are implied by their incident edges. *Gossamer* accepts FASTA and FASTQ input and uncompresses files on the fly. The current version of *Gossamer* allows for values of $k \leq 62$.

To attain its memory efficiency, *Gossamer* uses a compressed bitmap representation of the de Bruijn graph (Conway and Bromage, 2011). In brief, for a collection of reads containing $m$ distinct $\rho$-mers, the bitmap has $4^\rho$ entries, each of which is 1 if it corresponds to a $\rho$-mer from the dataset and 0 otherwise. For realistic datasets, these bitmaps are extremely sparse, and the theoretical minimum number of bits required to represent them is $log_2\binom{4^\rho}{m}$. The *Bombus impatiens* dataset used in Section 3 below contains 1 119 868 977 different $\rho$-mers at $k = 45$. The theoretical minimum size for this dataset according to the above definition is about 8 GB. The bitmap we construct actually requires ~9.2 GB of space, which is close to the minimum. In contrast, storing the $\rho$-mers themselves, using a straightforward 2 bits-per-base representation, would require almost 15.7 GB. Note that we have not considered the additional space required to store edge counts, which we also represent compactly.

The figures given above represent the amount of space required to store all of the $\rho$-mers from the example dataset. Realistically, many of those will correspond to errors in the data. *Gossamer* provides multiple operations for removing spurious edges from the graph, both spectral and structural. The spectral error removal operation is the trimming of low-frequency edges. The structural error removal operations are the pruning of *tips*, and the elimination of *bubbles*, both based on the algorithms present in Velvet (Zerbino and Birney, 2008). Unlike the Euler family of assemblers, *Gossamer* does not attempt to correct errors and removes them from the graph.

After these graph-cleaning passes, the resulting de Bruijn graph contains many fewer spurious edges, and the unbranched paths can be read off as preliminary contigs. Read pair information is utilized by aligning both ends of the pair to the de Bruijn graph to find pairs of 'anchors' into parts of the graph judged to be most likely unique (i.e. copy-number one) in the underlying genome. For each pair of anchors with sufficient support, a search is performed to find a unique path that is consistent with the bounds defined by the distribution of insert sizes. Where such paths are found, an Eulerian super-path is constructed. In the case where a supporting path is not found, the alignment of the read pairs can be used to perform scaffolding, by inferring the relative orientation and displacement of contigs.

## 3 RESULTS

We have evaluated *Gossamer*'s performance on the datasets used in the GAGE (Genome Assembly Gold-Standard Evaluations) study (Salzberg *et al.*, 2011). GAGE is a recent attempt to assess the capabilities of a collection of modern assemblers on a range of datasets, ranging from small bacterial genomes, to a human chromosome and an entire bumblebee genome. For comparison with *Gossamer*, we have rerun the most recent versions of *SOAPdenovo* (Li *et al.*, 2010) and *SGA* (String Graph Assembler) (Simpson and

*To whom correspondence should be addressed.

**Table 1.** Comparison of assembly results for *Gossamer* (Go), *SOAPdenovo* (SO) and *SGA* (SG), when run on the GAGE datasets: *Staphylococcus aureus* (SA), 2.9 Mb; *Rhodobacter sphaeroides* (RS), 4.6 Mb, human chromosome 14 (HG), 88.3 Mb; and *Bombus impatiens* (BI), estimated 250 Mb.

| Data | Tool | Time (s) | Mem (MB) | Contigs | | | | | Scaffolds | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Num | N50 (kb) | E-size (kb) | Errs | N50C (kb) | Num | N50 (kb) | E-size (kb) | Errs | N50C (kb) |
| SA | Go | 197 | **483** | 135 | 48.1 | 73.0 | 21 | 46.1 | **31** | **828** | **612** | 7 | **828** |
| | SO | **71** | 704 | **114** | **271.5** | **218** | 48 | **56.3** | 100 | 331 | 299 | 4 | 331 |
| | SG | 2688 | 1293 | 1183 | 4.0 | 4.7 | **11** | 4.0 | 536 | 113 | 150 | **0** | 113 |
| RS | Go | 258 | **531** | 744 | 12.2 | 15.3 | 19 | 12.0 | 270 | 132 | 131 | 2 | 132 |
| | SO | **94** | 833 | **210** | **138.9** | **161** | 328 | **17.6** | **174** | **667** | **478** | 5 | **343** |
| | SG | 4100 | 2089 | 2695 | 2.2 | 3.2 | **11** | 2.2 | 1739 | 47.2 | 46 | **0** | 42.5 |
| HG | Go | 7156 | **2721** | **29 622** | **4.6** | **6.9** | **1697** | **4.3** | **6932** | 369 | **681** | **172** | **182** |
| | SO | **1770** | 8812 | 41692 | 2.2 | 3.5 | 4589 | 2.2 | 13 436 | **402** | 487 | 254 | 83 |
| | SG | – | 39372 | – | – | – | – | – | – | – | – | – | – |
| BI | Go | 48 916 | **7926** | **51 518** | **10.9** | **17.2** | NA | NA | 25 996 | 240 | 297 | NA | NA |
| | SO | **21 664** | 23730 | 56557 | 9.0 | 13.0 | NA | NA | **6013** | **1429** | **1728** | NA | NA |

The columns read as follows: *Num*, the number of sequences produced; *N50*, the N50 statistic calculated with respect to the genome size; E-size, the most likely size of the contig or scaffold containing some random base in the genome, [see (Salzberg *et al.*, 2011) for detail]; *Errs*, the number of misjoins and for the contig value, also the number of indels >5 bases; and *N50C*, the N50 calculated after splitting all contigs/scaffolds at error locations. The best result in each column, for each dataset, is indicated in bold.

Durbin, 2012) on the same datasets, employing the same assembly 'recipes' that were used in the published GAGE result.

The *SOAPdenovo* results were generated by *SOAPdenovo* 1.0.5 and *SOAPGapCloser* 1.12. We used version 0.9.19 of *SGA* in combination with *ABySS* 1.2.5 (Simpson *et al.*, 2009), which *SGA* requires to perform scaffolding. Running *SGA* with a more recent version of *ABySS* (1.3.2) yielded scaffolds with almost no improvement over the original contigs.

The GAGE datasets are available in three forms: original reads and two varieties of corrected reads. Each assembler was run on all datasets and the best result selected. We have done the same in our evaluation of *Gossamer*.

The results of the assemblies are shown in Table 1. All figures, other than time and memory usage, were generated by the publicly available GAGE evaluation and validation scripts. Because no reference exists for *Bombus impatiens*, the number of errors and corrected N50 cannot be calculated.

We report the minimum amount of memory 'required' by each assembler to run to completion. During initial graph construction, *Gossamer* can make use of additional memory to hold temporary buffers, potentially saving some writes to disk, and thereby improving runtime. The use of this additional memory is only incidental, however, and does not affect the assembler's output. For all the tests, we used a single server with 8 AMD Barcelona cores and 32GB RAM running Ubuntu Linux, and we have configured the assemblers to use as many cores and as much of that memory as possible. Note that although the published GAGE figures include a result for *SGA* on the human chromosome 14 datasets, we were unable to run the assembler satisfactorily on our machines on account of its memory usage. As mentioned in the published GAGE result, *SGA* is not able to run on the *Bombus impatiens* data.

For the *Staphylococcus aureus* and *Rhodobacter sphaeroides* datasets, *Gossamer*'s scaffolds contain a number of errors that have no significant bearing on the N50 score. *SOAPdenovo*'s *S. aureus* result has the same feature, but for the larger genomes the drop in quality is significant. Where it was able to run, *SGA* has produced the shortest contigs, but with the fewest errors.

Overall, *Gossamer* requires consistently less memory for assembly than *SOAPdenovo* and *SGA*. This is especially significant for the larger genomes. *SOAPdenovo* was, without exception, the quickest of the assemblers tested, while *SGA*'s memory usage and runtimes are significantly higher than the other assemblers'.

The large difference between assemblers in the *Bombus impatiens* results requires further investigation. Without a reference, we cannot be sure of the quality of the generated sequences. We note that, at least for the other data assemblies, *Gossamer*'s N50 results appear more stable in the presence of assembly errors, and so *Gossamer*'s N50 for the *Bombus impatiens* dataset may be a more reliable indicator of the actual assembly quality.

*Funding*: National ICT Australia (NICTA) is funded by the Australian Government's Department of Communications; Information Technology and the Arts; Australian Research Council through Backing Australia's Ability and ICT Centre of Excellence programs.

*Conflict of Interest*: none declared.

## REFERENCES

Chaisson,M.J. *et al.* (2009) *De novo* fragment assembly with short mate-paired reads: does the read length matter? *Genome Res.*, **19**, 336–346.

Conway,T.C. and Bromage,A.J. (2011) Succinct data structures for assembling large genomes. *Bioinformatics*, **27**, 479–486.

Hernandez,D. *et al.* (2008) *De novo* bacterial genome sequencing: Millions of very short reads assembled on a desktop computer. *Genome Res.*, **18**, 802–809.

Li,R. *et al.* (2010) *De novo* assembly of human genomes with massively parallel short read sequencing. *Genome Res.*, **20**, 265–272.

Salzberg,S.L. *et al.* (2011) GAGE: a critical evaluation of genome assemblies and assembly algorithms. *Genome Res.*, **22**, 557–567.

Simpson,J.T. and Durbin,R. (2012) Efficient de novo assembly of large genomes using compressed data structures. *Genome Res.*, **22**, 549–556.

Simpson,J.T. *et al.* (2009) ABySS: a parallel assembler for short read sequence data. *Genome Res.*, **19**, 1117–1123.

Warren,R.L. *et al.* (2006) Assembling millions of short dna sequences using ssake. *Bioinformatics*, **23**, 500–501.

Zerbino,D.R. and Birney,E. (2008) Velvet: algorithms for *de novo* short read assembly using de bruijn graphs. *Genome Res.*, **18**, 821–829.

Zerbino,D.R. *et al.* (2009) Pebble and rock band: heuristic resolution of repeats and scaffolding in the velvet short-read de novo assembler. *PLoS ONE*, **4**, e8407.