A fast hybrid short read fragment assembly algorithm

Bertil Schmidt1, Ranjan Sinha2, Bryan Beresford-Smith3 and Simon J. Puglisi4

1School of Computer Engineering, Nanyang Technological University, Singapore 639798, 2Department of Computer Science and Software Engineering, University of Melbourne, 3National ICT Australia, Victoria Research Lab, Victoria 3010 and 4School of Computer Science and IT, RMIT University, Melbourne 3001, Australia

Received on February 8, 2009; revised on June 7, 2009; accepted on June 12, 2009

ABSTRACT

Summary: The shorter and vastly more numerous reads produced by second-generation sequencing technologies require new tools that can assemble massive numbers of reads in reasonable time. Existing short-read assembly tools can be classified into two categories: greedy extension-based and graph-based. While the graph-based approaches are generally superior in terms of assembly quality, the computer resources required for building and storing a huge graph are very high. In this article, we present Taipan, an assembly algorithm which can be viewed as a hybrid of these two approaches. Taipan uses greedy extensions for contig construction but at each step realizes enough of the corresponding read graph to make better decisions as to how assembly should continue. We show that this approach can achieve an assembly quality at least as good as the graph-based approaches used in the popular Edena and Velvet assembly tools using a moderate amount of computing resources.

Availability and Implementation: Source code in C running on Linux is freely available at http://taipan.sourceforge.net

Contact: asbschmidt@ntu.edu.sg

1 INTRODUCTION

Recently, several second-generation DNA sequencing technologies have been introduced. Compared to traditional Sanger shotgun techniques these technologies can generate a larger amount of read data at lower cost (Mardis 2008; Pop and Salzberg, 2008). Examples of already available second-generation technology include products from Illumina, 454 Life Sciences and Applied Biosystems. However, the length of produced reads is significantly shorter than for the classical Sanger technique. For example, the Illumina Genome Analyzer can generate 1.5 Gb of sequence data in a single-end 60h run with a read length of around 36.

Several new assemblers that are tailored towards short reads and high coverage have recently been introduced. SSAKE (Warren et al., 2007), VCAKE (Jeck et al., 2007) and SHARCGS (Dohm et al., 2007) are based on the greedy extension approach, while newer tools like Edena (Hernandez et al., 2008), Velvet (Zerbino and Birney, 2008) and Euler-SR (Chaisson and Pevzner, 2008) use more accurate graph-based approaches. Graph-based approaches can be further distinguished into the overlap graph approach (Edena) and the De Bruijn graph approach (Velvet, Euler-SR). Previous work (Hernandez et al., 2008) has already shown that graph-based approaches achieve superior assembly performance compared to greedy extensions. In this article we present Taipan, a fast hybrid short read fragment assembly algorithm. Taipan uses a combination of the greedy extension and the overlap graph method. Our performance evaluation using real Illumina datasets shows that Taipan can achieve assembly qualities comparable to the graph-based approaches within a reasonable execution time.

2 METHODS

Input to Taipan is a multi-set of reads R of length l each (consisting of the original reads plus their reverse complements), the minimal overlap parameter k < l, and a threshold T. R is stored in a hash table. This hash table allows efficient processing of queries of the form get_overlapping_reads(S,r) for a DNA string S and r ∈ {4, ..., l}, which return all reads in R whose prefix r matches the suffix r of S. Taipan assembles a new contig by choosing a read from R as a seed. This seed is iteratively extended in 3’ direction by one base at a time until there are either insufficient overlapping reads or a repeat is found. Subsequently, the same algorithm is used in 5’ direction. The algorithm to extend contig S in 3’ direction by a single base works as follows. A set O of overlapping reads is retrieved from the hash table by calling get_overlapping_reads(S) for each r ∈ {l, ..., l-1}. Afterwards the directed overlap graph Gw(O,E) is constructed, where (r1, r2) ∈ E if suffix s of r1 matches prefix t of r2 for some t ∈ {k, ..., l}. Gw(O,E) is then built by removing all associative edges from G(O,E) if associative, if there exist two other edges (r1, r2), (r1, r3) ∈ E). The set P consisting of all vertex-disjoint paths of Gw is determined. P is analyzed to determine the single-base extension of S using threshold T as follows: (a) if P contains at least two paths p1, p2 with length (p1) ≥ T, (1, 2), then a repeat is found and extension of S is terminated; (b) if P contains exactly one path p with length (p) ≥ T, then S is extended by the first base of the first path p in the order of the paths (c) if P contains no path p with length (p) ≥ T, then there are insufficient overlapping reads and extension of S is terminated; (d) if none of the above cases apply, Gw is enlarged along each path p and the rules (a)–(d) are recursively applied (up to a maximum of T steps) to the enlarged graph. After the assembly of a new contig all reads that exactly match this contig are removed from R and the hash table. The implementation of Taipan gains efficiency by the fact that O only slightly changes for subsequent extensions and by using a sorted trie data structure to represent G1, ← → and P. Choice of seeds can affect assembly results. Taipan selects an unassembled read with highest occurrence as seed in order to minimize extensions of reads containing sequencing errors.

3 RESULTS

We have evaluated the performance of Taipan in terms of both runtime and assembly quality using two real Solexa/Illumina
datasets and compared the results to Edena and Velvet (Table 1). The datasets have been obtained experimentally by (Hernandez et al., 2008) and (Dohm, et al., 2007) using the Illumina Genome Analyzer for sequencing the *Staphylococcus aureus* strain MW2 (*Aureus*) and *Helicobacter acinonychis* strain Sheeba (*Helicobacter*). Respective read files were downloaded from http://www.genomic.ch/edena.php and http://sharcgs.molgen.mpg.de/download.shtml. A summary of both datasets is provided in the format: genome GenBank ID, plasmid GenBank ID, genome length, coverage, read length, number of reads:

- **Staphylococcus**: NC_003923, NC_005011, 2.82 Mb, 48×, 35, 3.80 M
- **Helicobacter**: NC_008299, NC_008230, 1.55 Mb, 284×, 36, 12.3 M

We have executed Taipan, Edena 2.1.1 (for Linux-64), and Velvet 0.7.31 for each dataset on a workstation with Quadcore Xeon E5440 2.83 Ghz CPU and 16 GB RAM running Linux CentOS 5.2-64 bit. Table 1 reports the assembly performance using the best tested parameter setting for each tool. The best tested parameter settings for the *Aureus* dataset are:

- Taipan: $k = 19, T = 8$
- Edena: minimum overlap = 21 (strict)/20 (non-strict)
- Velvet: hash size = 21, coverage cutoff = 7.

The best tested parameter settings for the *Helicobacter* dataset are:

- Taipan: $k = 27, T = 18$
- Edena: minimum overlap = 27 (strict)/26 (non-strict)
- Velvet: hash size = 27, coverage cutoff = 8.

All tools use a minimal contig length of 100 bp. Produced contigs are aligned to the respective circular reference genome and plasmid sequences using Exonerate (Slater and Birney, 2005). A contig is classified as correct if it fully aligns with a base similarity of at least 98%. Otherwise, it is classified as misassembled. Calculation of N50, N80, average length, maximum length, and coverage only considers contigs classified as correct. Measured runtimes are real execution times and include both preprocessing and assembly time.

For the *Aureus* dataset, strict mode Edena produces no misassembled contigs but has the lowest N50 value and the shortest runtime. Furthermore, the total size of *Taipan*’s misassembled contigs (2.8 kb) is significantly shorter compared to Velvet (14.5 kb) and Edena non-strict (36.6 kb). For the *Helicobacter* dataset, both Taipan and Edena strict produce no misassembled contigs, whereby *Taipan* has a clearly higher N50 value. Edena non-strict and Velvet produce N50 values close to *Taipan* but generate a single long misassembled contig and two short misassembled contigs, respectively. *Taipan* remains faster than Edena, however, due to the usage of a large threshold parameter ($T = 18$), *Taipan* is slower than Velvet.

In conclusion, *Taipan* is an efficient and accurate tool for de-novo genome assembly of short reads generated by high-throughput sequencing machines. The utilized algorithm shows that simple operations that do not require the whole overlap graph can achieve results similar to methods that construct the whole graph. *Taipan*’s results for both the medium-coverage (48×) *Aureus* dataset and the high-coverage *Helicobacter* dataset (284×) are competitive in terms of speed and quality compared to Edena as well as Velvet. We are planning to improve *Taipan* further by running read error correction beforehand and including paired-read data information.

**Conflict of Interest:** none declared.

**REFERENCES**


---

Table 1. Comparison of assembly results for the *Aureus* dataset and the *Helicobacter* dataset as obtained by *Taipan*, Edena and Velvet

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Tool</th>
<th>Number of correct contigs</th>
<th>Number of misassembled contigs (total size)</th>
<th>Correct contigs</th>
<th>Coverage (%)</th>
<th>Runtime (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>N50</td>
<td>N80</td>
<td>Average length</td>
</tr>
<tr>
<td><em>Aureus</em></td>
<td>Taipan</td>
<td>692</td>
<td>16 (2.8 kb)</td>
<td>11 162</td>
<td>4623</td>
<td>3997</td>
</tr>
<tr>
<td></td>
<td>Edena strict</td>
<td>1124</td>
<td>0 (0 bp)</td>
<td>5924</td>
<td>2253</td>
<td>4623</td>
</tr>
<tr>
<td></td>
<td>Edena non-strict</td>
<td>740</td>
<td>17 (36.6 kb)</td>
<td>9053</td>
<td>3899</td>
<td>3713</td>
</tr>
<tr>
<td></td>
<td>velvet</td>
<td>945</td>
<td>5 (14.5 kb)</td>
<td>7289</td>
<td>2692</td>
<td>2898</td>
</tr>
<tr>
<td><em>Helicobacter</em></td>
<td>Taipan</td>
<td>271</td>
<td>0 (0 bp)</td>
<td>13 312</td>
<td>5317</td>
<td>5600</td>
</tr>
<tr>
<td></td>
<td>Edena strict</td>
<td>336</td>
<td>0 (0 bp)</td>
<td>10 167</td>
<td>4674</td>
<td>4539</td>
</tr>
<tr>
<td></td>
<td>Edena non-strict</td>
<td>302</td>
<td>1 (24.1 kb)</td>
<td>13 206</td>
<td>5210</td>
<td>4979</td>
</tr>
<tr>
<td></td>
<td>velvet</td>
<td>278</td>
<td>2 (0.2 kb)</td>
<td>12 590</td>
<td>5350</td>
<td>5465</td>
</tr>
</tbody>
</table>

*For the *Helicobacter* dataset, both *Taipan* and Edena strict produce no misassembled contigs, respectively. *Taipan* remains faster than Edena, however, due to the usage of a large threshold parameter ($T = 18$), *Taipan* is slower than Velvet.*