DGraph: Algorithms for Shortgun Reads Assembly Using De Bruijn Graph

Jintao Meng, Jianrui Yuan, Jiefeng Cheng, Yanjie Wei, Shengzhong Feng

DGraph: Algorithms for shortgun reads assembly using De Bruijn Graph

Jintao Meng[1,2,4], Jianrui Yuan[2,3], Jiefeng Cheng[2], Yanjie Wei[2], Shengzhong Feng[2*]

[1] Institute of Computing Technology, CAS, Beijing, 100190, PR.China
[2*] Shenzhen Institutes of Advanced Technology, CAS, Shenzhen, 518055, PR. China
[3] Central South University, Changsha, 410083, PR. China
[4] Graduate University of Chinese Academy of Sciences, Beijing, 100049, China
{jt.meng, jr.yuan, jf.cheng, yj.wei, sz.feng}@siat.ac.cn

Abstract. Massively parallel DNA sequencing platforms have become widely available, reducing the cost of DNA sequencing by over two orders of magnitude, and democratizing the field by putting the sequencing capacity of a major genome center in the hands of individual investigators. New challenges include the development of robust protocols for generating sequencing libraries, building effective new approaches to resequence and data-analysis. In this paper we demonstrate a new sequencing algorithm, named DGraph, which has two modules, one module is responsible to construct De Bruijn graph by cutting reads into k-mers, and the other's duty is to simplify this graph and collect all long contigs. The authors didn't adapt the sequence graph reductions operations proposed by RAMANA M.IDURY or Finding Eulerian Superpaths proved by Pavel A.Pevzner or bubble remove steps suggested by Danial Zerbino, As the first operations was computing expensive, and the second one was impractical, and the last one did not benefit either the quality of contigs or the efficiency of the assembler. Our assembler was focused only on efficient and effective error removal and path reduction operations. Applying DGraph to the simulation data of fruit fly Drosophila melanogaster chromosome X, DGraph (3min) is about six times faster than velvet 0.3 (19 mins), and its coverage (92.5%) is also better than velvet (78.2%) when k = 21. Compare to velvet, the results shows that the algorithm of DGraph is a faster program with high quality results.

Keywords: De Bruijn graph, graph algorithm, short read assembler

1 Introduction

Each cell of a living organism contains chromosomes composed of a sequence of DNA base pairs. This sequence, the genome, represents a set of instructions that controls the replication and function of each organism. The automated DNA sequencer gave birth to genomics, the analytic and comparative study of genomes, by allowing scientists to decode entire genomes.

Although genomes vary in size from millions of nucleotides in bacteria to billions of nucleotides in humans and most animals and plants, the chemical reactions

researchers use to decode the DNA base pairs are accurate for only about 600 to 700 nucleotides at a time.

The process of sequencing begins by physically breaking the DNA into millions of random fragments, which are then "read" by a DNA sequencing machine. Next, a computer program called an assembler pieces together the many overlapping reads and reconstructs the original sequence. This general technique, called shotgun sequencing, was introduced by Fred Sanger in 1982. Recently, new sequencing technologies have emerged [1] (Metzker 2005), for example, pyrosequencing (454 sequencing) [2] (Margulies et al. 2005) and sequencing by synthesis (Solexa) [3] (Bentley 2006), both commercially available. Compared to traditional Sanger methods, these technologies produce shorter reads, currently ~200 bp for pyrosequencing and 35bp for Solexa. Until recently, very short read information was only used in the context of a know reference assembly, either for sequencing individuals of the same species as the reference, or readout assays.

Sequencing remains at the core of genomics. Current sequencing approaches are classed into two strategies. The first one is overlap-layout-consensus, its applications includes TIGR assembler [4], Phrap[5], and CAP3 [6]; the second one is graph-theoretical approach, and its applications includes VCAKE[7], SSAKE[8], Velvet[9].

The assembler following the first approach must first build the graph by computing all possible alignments between the reads. A second stage cleans up the graph by removing transitive edges and resolving ambiguities. The output of this stage comprised a set of nonintersecting simple paths in this refined graph, each such path corresponding to a contig. A final step generates a consensus sequence for each contig by constructing the multiple alignments of the reads that is consistent with the chosen path. Although this approach are relatively easy to implement, but they are inherently local in nature and ignore long-range relationships between reads, which could be useful in detecting and resolving repeats. In addition, all current implementations of the greedy method require huge memory. This limits their applicability on current available hardware to organisms with gnomes of 32 Mbp or less.

The second approach to shotgun sequence assembly uses a sequencing-by-hybridization technology. The idea is create a virtual SBH problem by breaking the reads into overlapping n-mers, where an k-mer is a substring of length k from the original sequence. Next, the assembler builds a directed De Bruijn graph in which each edge corresponds to an k-mer from one of the original sequence reads. Finally, the problem of reconstructing the original DNA molecule corresponds to finding a path that uses all the edges-that is, an Eulerian path. In theory, the Eulerian path approach is computationally far more efficient than the overlap-layout-consensus approach because the assembler can find Eulerian paths in linear time while the problems associated with the overlap-layout-consensus paradigm are NP-complete. Despite this dramatic theoretical difference, the actual performance of existing algorithms indicates that overlap-layout-consensus approach is just as fast as the SBH-based approach.

The algorithm proposed in this paper fell into the graph-theoretical approach. As computation time and quality of contigs still limit the practical use of these implementations adopting de Bruijn graphs approach to genomes on the order of a billion base in size. Compared with the previous works, DGraph will pay attention to these two points. For this result, the authors didn't adapt the sequence graph

reductions operations proposed by RAMANA M.IDURY or Finding Eulerian Superpaths proved by Pavel A.Pevzner or bubble remove steps suggested by Danial Zerbino, As the first operations was computing expensive, and the second one was impractical, and the last one did not benefit either the quality of contigs or the computation efficiency of the assembler. Our assembler was focused only on efficient and effective error removal and path reduction operations.

We organize the paper as follows, Section II will introduce the operations of two modules in detail. The experiment and performance will be demonstrated in Section III, and the last section will discuss the weak points of the algorithm and our further work.

2 Algorithmic Approach

The processes of the algorithm DGraph was descript in figure 1. DGraph has two modules, the first one is graph construction, which is responsible to construct De Bruijn graph by cutting reads into k-mers, and the second one is graph simplification whose duty is to simplify this graph.

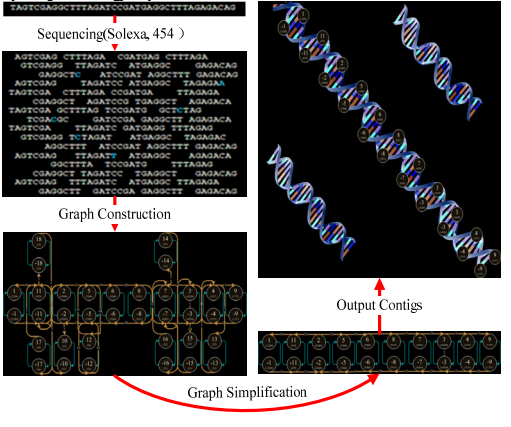

Fig 1. The processes of algorithm DGraph.

2.1 Graph Construction Module

In the De Bruijn graph, each node represents a k-mers, and adjacent k-mers overlap each other with k-1 nucleotides. Each node N is attached to the node ~N, which represents the reverse series of reverse complement k-mers. The adjacent nodes are connected by arc, for each node A to B, a symmetric arc goes from ~B to ~A.

For each read, we cut it into k-mer (k must be an odd number, k<32) and keep it into a hash table. For each k-mers, the hash table will keep this string of k-mers and its node ID in the graph. Each k-mer has a reverse complement.

Then, all adjacent k-mers (or nodes) in the same read are connected. Its reverse complement is also linked with a reverse arc. Each arc has a weight to record how many reads contain this two k-mers. The Construction module of DGraph processes each reads and connects all adjacent k-mers to build a De Bruijn graph.

2.2 Graph Simplification Module

2.2.1 Error link removal

Generally speaking, 1%~3% errors will be imported into De Bruijn graph from input reads because of the precision of modern sequencing machines. All low coverage arcs will be thought as error links and must be deleted.

There will be a threshold θ for each arc in the De Bruijn graph, if the weight of arc is less than θ, this arc will be deleted. Then if some node is disconnected with other nodes we will also delete that node.

2.2.2 Path Simplification

This step is to simplify the De Bruijn graph. When a node A has only one outgoing arc that points to node B, then the two nodes can be merged, and their reverse complements should be merged too.

Our module will degrade each chain of nodes into one node. After simplification, the number of nodes will be greatly declined. There possibly are some isolated nodes, we simply delete all of these isolated nodes, as we think it was introduced by error reads.

2.2.3 Tips Removal

Tip is a chain of node which disconnected on one end. All tips will be removed if it is shorter than 2k, this parameter is a cutoff length which was chosen to delete all erroneous constructs.

After that our algorithm produced a much more simply graph without loss of information. We did not do any further operations as the other algorithm does, because we did not agree that bubbles removing step was appropriate operation in velvet[9], it simply combined two path from node A to node B into one, but these two path may actually exist in the finally contigs. We also did not apply graph reduction operation mentioned in [10], in principle, this algorithm was great, it did give a way to simplify the De Bruijn graph; However it is not practical to realize the graph reduction, and the

complexity on maintain the data structure defined in [10] is enormous high, not to mention their complex mathematic operation. Even the prototype developed by the authors of [8] will use 10s on assembling 20k data.

3 Experimental Results

The sequence data is produced from Drosophila melanogaster (fruit fly) chromosome X (its sequence size is 21.7M, the NCBI reference sequence ID is NC_004354.3). Our reads producer program will cut NC_004354.3 into short reads with length range from 200bp to 400bp, and the coverage of our read set is 30. Errors rate of 1% was introduced into the read set. The final size of the read set is about 571M.

The DGraph algorithm was compared with velvet 0.3 in our Evaluation in both contig quality and algorithm performance.

The read set will be assembled by DGraph and velvet. We ran the entire test with different parameter k, which was an odd number ranged from 19 to 31. We verified the assemblies by aligning the resulting contigs to the reference sequence by NCBI blast [11] to calculate the quality of the results.

First, the quality analysis of the resulting contigs was illustrated from figure 2 to figure 7. Figure 2 shows that the number of resulting contigs produced by DGraph are about two-third as many as the number of contigs output by velvet. In max length of contigs, DGraph is two times longer than velvet according to figure 3. The same trend also happens in figure 4 on their N50 length. Figure 5 demonstrated that DGraph has less number of contigs larger than N50 compared to velvet, which means that contigs produced by DGraph are mainly consisted with a few long contigs and a large amount of short contigs. All in all, the four figure shows that DGraph has much less but much longer contigs longer than N50 compared with velvet. What's more, the coverage of DGraph's resulting contigs (92.5%) is slightly better than the coverage of velvet's according (78.2%) to figure 6.

Second, In figure 7, DGraph consumed about one-sixth as much cpu time as velvet did, which is 3minutes and 19minutes respectively. But in terms of memory consumption DGraph is two times larger than velvet. Generally speaking, DGraph can produce higher quality of resulting contigs with one-sixth cpu time used by velvet, only at the cost of more memory.

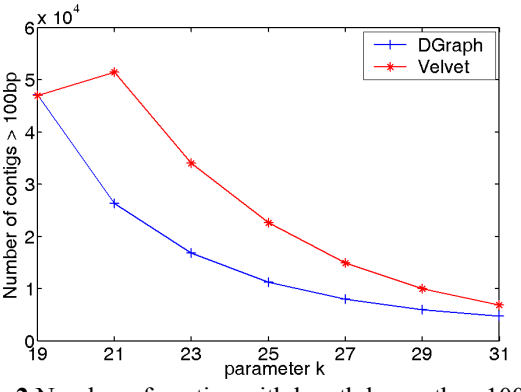

Fig.2 Number of contigs with length larger than 100bp

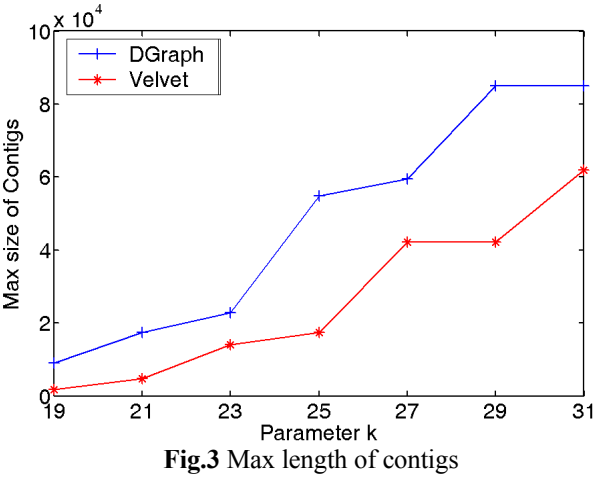

Fig.3 Max length of contigs

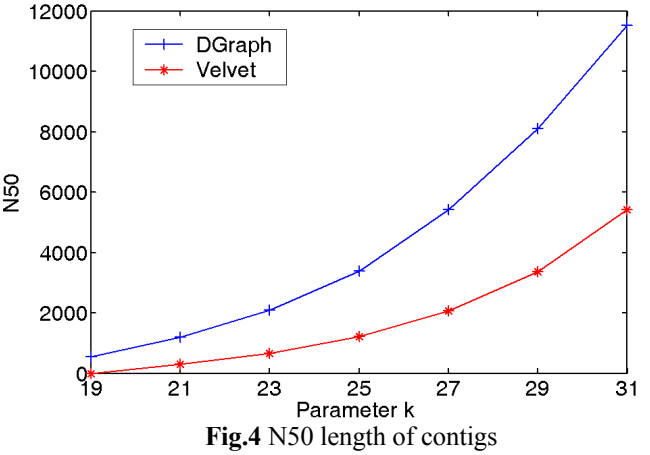

Fig.4 N50 length of contigs

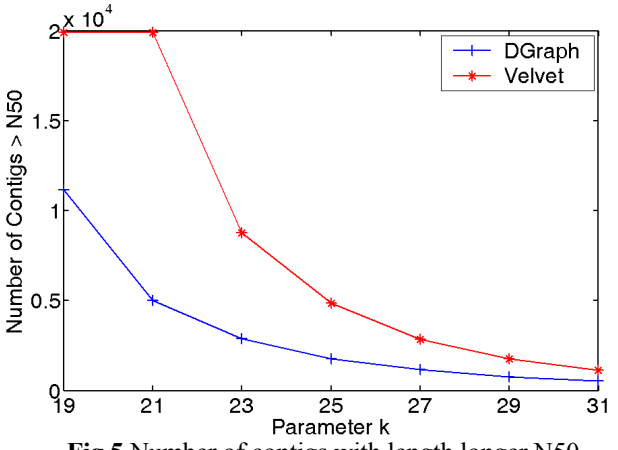

Fig.5 Number of contigs with length longer N50

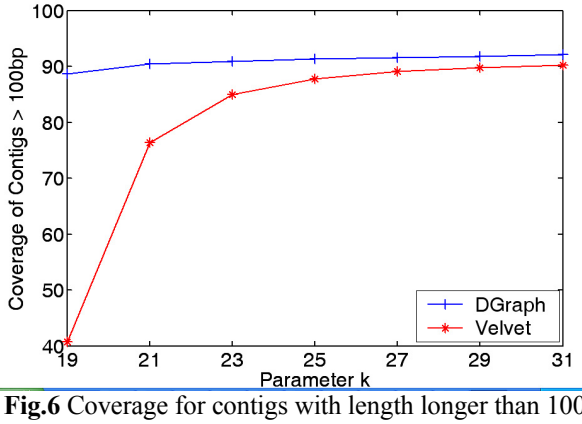

Fig.6 Coverage for contigs with length longer than 100bp

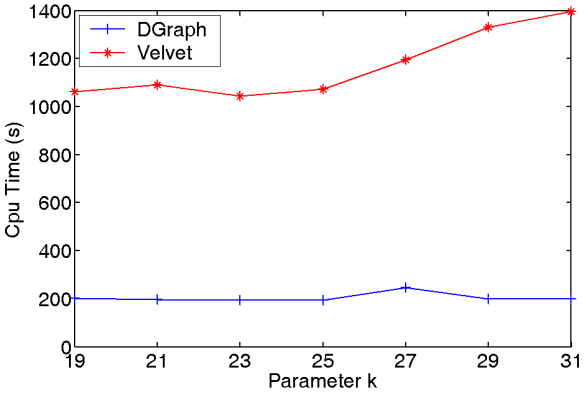

Fig.7 CPU time consumed by DGraph and Velvet

4 Discussion

The DNA sequencing technology field has become a quickly moving target, Fast and high quality Algorithms that can assemble millions of small DNA fragments into gene sequences are located in our research scope. DGraph, a new de novo sequencing algorithm was proposed in this paper, Core operations, such as graph construction and graph simplification was implemented effectively to support high quality configs. The experiment demonstrated that DGraph outperformance velvet both on assembler speed and contigs' quality. DGraph was just a basical assembler framework on resequencing, father optimization on memory consumption and code parallelization was urgent work in the development of next version. Last, but not least, is there exist a practical method to reduce the simplified graph, Can this method work efficiently and indeed improve the quality of resulted contigs.

Acknowledgements

This work is supported by NSFC (Grant No. 61103049) and Shenzhen Research Fund (Grant No.JC201005270342A). The author also thanks Lin Fang, Bingqiang Wang and their teammates from BGI, and Sitong Sheng from HYK Gene for their supports and suggestions on this work.

References

1. Michael L. Metzker, Jing Lu, Richard A. Gibbs: Electrophoretically Uniform Fluorescent Dyes for Automated DNA Sequencing. Science, no. 271, vol. 5254, pp. 1420 – 1422 (2009)
2. Margulies M, etc: Genome sequencing in microfabricated high-density picolitre reactors. Nature, no.4, vol. 441 (2006)
3. Bentley DR,: Whole-genome re-sequencing. Current opinion in genetics & development, no.16 vol.6, pp.545-552 (2006)
4. Granger G. Sutton, Owen White, Mark D. Adams, Anthony R. Kerlavage: TIGR Assembler: A New Tool for Assembling Large Shotgun Sequencing Projects, Genome Science and Technology, no.1 vol.1 pp.9-19 (1995)
5. P. Green. http://bozeman.mbt.washington.edu/phrap.docs/phrap.html.
6. X. Huang and A. Madan: CAP3: A DNA Sequence Assembly Program. Genome Research, no.9, pp.868-877, 1990
7. Jan F. Kreuze, Ana Perez, Milton Untiveros, Dora Quispe, Segundo Fuentes, Ian Barker and Reinhard Simon: Complete viral genome sequence and discovery of novel viruses by deep sequencing of small RNAs: A generic method for diagnosis, discovery and sequencing of viruses. Virology. no.388 vol.1 pp.1-7 (2009)
8. René L. Warren, Granger G. Sutton, Steven J. M. Jones, et al: Assembling millions of short DNA sequences using SSAKE. Bioinformatics no.23 vol.4 pp.500-501 (2007)
9. Zerbino DR, Birney E: Velvet: algorithms for de novo short read assembly using De Bruijn graphs. Genome Res, no.18, vol.5, pp.821-829 (2008)
10. RM Idury, MS Waterman: A New Algorithm for DNA Sequence Assembly. Journal of Computational Biology (1995)
11. blast, http://www.ncbi.nlm.nih.gov/blast/producttable.shtml#mega