ISBN: 978-989-8704-10-8 © 2014

OVERLAP GRAPH SIMPLIFICATION USING EDGE RELIABILITY CALCULATION

Sergey Kazakov and Anatoly Shalyto
ITMO University, Computer Technologies Laboratory - Russia, Saint Petersburg

ABSTRACT

One of the problems in genome assembly is overlap graph complexity. Despite the existence of many overlap graph simplification methods, it is still tangled after applying many of them. A novel approach to solve this problem is presented. The proposed algorithm uses several types of information about edges in the overlap graph to simplify it. We define a reliability measure of an edge based on this information. The method has been tested on three bacteria with different characteristics; the results show the applicability and efficiency of the suggested approach.

KEYWORDS

Overlap Graph, de novo Genome Assembly, Bioinformatics

1. INTRODUCTION

Biological research is very important for a better understanding of life and diseases (Soh, 2007). Analysis of the information encoded in the genome is one way to do it. A great effort has been applied to describe organism genomes, for example Human Genome Project (Ridley, 2000; Ng, 2010). Research in this field continues: genome sequencing projects start all over the world, generating a large amount of sequencing data. The first and one of the most fundamental tasks in these projects is to determine the complete genome sequence of an organism.

A number of sequencing technologies are available nowadays, e.g. Ion Torrent, Illumina and others. Most of them are high-throughput sequencing technologies that generate thousands or millions of sequences. However, they have several disadvantages: low quality of reads, short read length, etc. This, in particular, makes the task of reconstructing the complete genome difficult.

Nevertheless, a lot of genome assemblers, like ABySS, MIRA, CLC bio, SOAPdenovo (Luo, 2012), SPAdes (Bankevich, 2012) and others, cope with this task quite successfully. However, assemblers that work well on particular data, do not work as well on other data (Bradnam, 2013). That makes the genome assembly problem still open.

One of the approaches to this problem is to use an *overlap graph* (Simpson, 2011). It is a graph in which vertices correspond to reads and edges correspond to overlaps between them. In comparison to commonly used *k*-mer-based de Bruijn graph (Luo, 2012), it uses a full read length, thus it makes it possible to use the information more thoroughly. An example of the overlap graph is shown in Figure 1.

A real-world overlap graph is usually very tangled after its construction and in this state it is not suitable to work with. A lot of simplification methods to untangle the graph exist. There are well-known methods based on simple steps like merging similar paths and removing tips, more complex methods based on flow algorithms (Pevzner, 2001) and other methods (Lai, 2012). Usually, several of these methods are applied consequently. Ideally, after applying them the overlap graph will be reduced to a single path. This path will be transformed to one contig in the next steps.

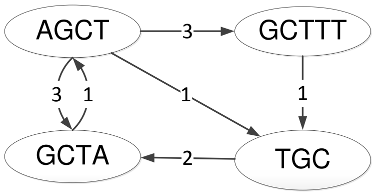

Figure 1. Overlap graph (edges are marked with overlap length)

However, even after applying simplification methods the graph is still tangled. That is why we developed a new method that can help to solve this problem.

The proposed method should be applied before all other steps. It tries to distinguish false edges from true ones. We consider an edge to be false if it connects two reads from different genome parts. For this aim a reliability measure of each edge is calculated based on several types of information about edges in the overlap graph. After that it is possible to remove edges with low reliability, and so, decrease the graph complexity. That allows the subsequent methods to work more efficiently. Also it can be helpful to use edge’s reliabilities in the next steps.

2. OVERLAP GRAPH SIMPLIFICATION METHOD

The idea of the approach is to add additional steps before all steps in overlap graph simplification stage. These additional steps are: calculating reliability of each edge and removing edges with low reliability.

We will consider edge reliability *P*(*e*) as the reliability of the statement “edge *e* is true”. For an edge *e* it is calculated using two following estimations.

- **Theoretical estimation $P_{\text{th}}(e)$** is calculated using the overlap length and the distribution of the number of repeats over repeat length.
- **Experimental estimation $P_{\exp}(e)$** is calculated by analyzing the *k*-mer coverage dependence for overlapping reads. The *k*-mer coverage dependence is a dependence of the number of *k*-mers found in all reads depending on the position of this *k*-mer in overlapping reads.

Reliability $P(e)$ is calculated by next equation:

$$P(e) = \alpha P_{\text{th}}(e) + \beta P_{\exp}(e), \qquad (2)$$

where α and β are coefficients ($\alpha + \beta = 1$). These coefficients can be set depending on our confidence in the aforementioned estimations. For example, in metagenome projects with highly non-uniform read coverage the coefficient β of experimental estimation can be decreased.

Consider two overlapping reads caused by a repeat in the genome (shown in Figure 2). As can be seen from the bottom part of the figure, constraints for boundaries of overlapping reads and the repeat should be the following:

1. $(A.\text{begin} < R.\text{begin})$ and $(R.\text{begin} \leq B.\text{begin})$,
2. $(A.\text{end} \leq R.\text{end})$ and $(R.\text{end} < B.\text{end})$.

Constraints (*R*.begin ≤ *B*.begin) and (*R*.end ≥ *A*.end) arose because a repeat should cover the overlapping part. Constraints (*R*.begin > *A*.begin) and (*R*.end < *B*.end) – because otherwise read *A* or read *B* will be fully covered by a repeat, and, thus, the edge between reads *A* and *B* would not be false because we do not know exactly where the read covered by the repeat was.

So, $P_{\text{th}}(e)$ and $P_{\exp}(e)$ estimate the reliability of a situation that there was no repeat *R* satisfying the described conditions that lead to an overlap between reads.

The theoretical estimation $P_{\text{th}}(e)$ is calculated using the equation:

$$P_{\text{th}}(e) = 1 - \textstyle\sum_R P_{\text{repeat}}(R), \qquad (3)$$

where summation is done over repeats *R* satisfying conditions above, and $P_{\text{repeat}}(R)$ is a probability that a repeat *R* with certain positions *R*.begin and *R*.end exists (and this repeat cannot be enlarged). $P_{\text{repeat}}(R)$ is estimated using a statistical model of the distribution of the number of repeats over repeat length.

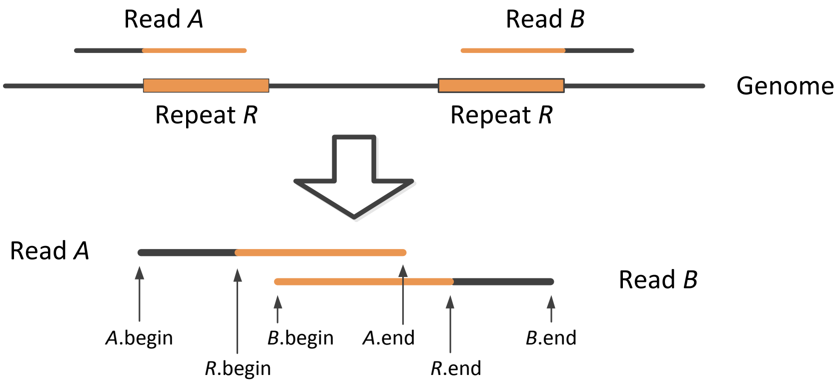

Figure 2. Overlapping reads caused by the repeat

The experimental estimation $P_{\text{exp}}(e)$ is calculated using the following steps. First of all, for each k-mer the number of times it is present in all reads is calculated. After that, for k-mers in overlapping reads a coverage dependence plot shown in Figure 3 can be constructed. Now we can notice an interesting feature: if a repeat that covers an overlapping part really exists, k-mers in the repeat will be covered on average two times or more. Therefore, in the next step, we will try to select such a central part of the plot (i.e. such R.begin and R.end satisfying conditions above), that k-mer coverage on this part will be at least two times greater than in other parts. Thus, the k-mer coverage dependence plot can be approximated by the scheme pictured in Figure 3. After that, the experimental estimation $P_{\text{exp}}(e)$ is calculated as a dissimilarity coefficient of these two plots.

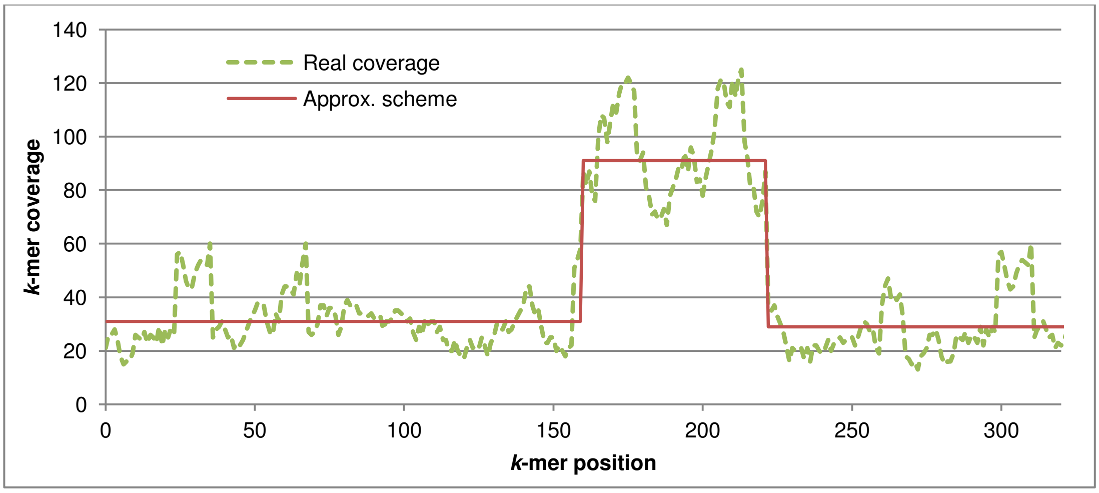

Figure 3. k-mer coverage plots

3. EXPERIMENTS

Experiments were performed over three different bacteria: *Haemophilus influenza, Escherichia coli* and *Buchnera aphidicola*. Each input dataset was obtained one of these ways:

- **Errorless reads** were generated from a reference genome.
- **Reads with errors** were generated from the same genome (errors were made by the generating program).
- **Real reads** were sequenced from a bacterium.

All input datasets consisted of single reads with average length more than 200 bp. The coverage of the genomes by reads ranged from 20x to 40x.

Experiments were performed using the *ITMO Genome Assembler* (Alexandrov, 2012) without applying the proposed method and with it. After running two versions of the assembler on the same input dataset, the resulting contigs were evaluated in terms of assembly quality. It was measured using various numerical characteristics. The most important are average contig length and number of contigs with misassembles (contigs that are not present in the reference genome). Traditional priorities in contigs assembly are the following:

1. **Reduce the number of contigs with misassembles**.
2. **Increase average contig length** (without changing total length of all assembled contigs).

The values of coefficients α and β in equation 2 were set to 0.6 and 0.4, correspondingly. An edge was considered to have low reliability (and later removed), if $P(e)$ was less than 0.30.

Here we present the results of experiments with two bacteria – *H. influenzae* (Table 1) and *E. coli* (Table2).

Table 1. The results of experiments with bacterium *H.influenzae*. Genome length is 1,813,033 nucleotides.

Dataset	Errorless reads		Reads with errors	
Algorithm Characteristic	Assembler without method	Assembler with method	Assembler without method	Assembler with method
Number of contigs	71	**62**	99	**74**
Total length	**1,789,421**	1,788,234	**1,793,292**	1,789,284
Maximal length	404,738	**405,614**	338,054	**338,438**
Average length	25,203	**28,842**	18,114	**24,180**
Minimal length	264	**279**	217	217
N50	**143,348**	143,093	92,731	**99,343**
N90	27,927	**33,879**	21,834	**27,874**
Contigs with misassembles	1	**0**	3	**1**

Table 2. The results of experiments with bacterium *E. coli*. Genome length is 4,639,675 nucleotides.

Dataset	Errorless reads		Real reads	
Algorithm Characteristic	Assembler without method	Assembler with method	Assembler without method	Assembler with method
Number of contigs	160	**140**	**405**	418
Total length	**4,599,061**	4,598,439	4,624,017	**4,629,167**
Maximal length	269,909	**327,486**	**164,338**	110,238
Average length	28,744	**32,846**	**11,417**	11,075
Minimal length	**254**	251	**214**	211
N50	112,550	**132,952**	**36,589**	36,001
N90	31,325	**31,960**	**9,764**	9,595
Contigs with misassembles	2	**1**	15	**9**

As can be seen from tables, the quality of the resulting contigs in all experiments was improved: in three experiments in both two main characteristics, in the last one – improvement was only in one characteristic. And even in the last case contig quality is increased due to 40% decreasing the number of contigs with misassembles.

In the case of *B. aphidicola* the application of the proposed method does not significantly change contig quality. We assume that it is due to the smaller genome size of this bacterium (641,895 nucleotides).

4. CONCLUSION AND FUTURE WORK

We have developed a new method for overlap graph simplification. This method was tested on different bacteria datasets. Experimental evaluation presented in section 3 demonstrates that the proposed algorithm is quite successful as an additional step to solve the overlap graph simplification problem.

As a future work, it is very important to check compatibility of the proposed algorithm with other assemblers, as well as in cases of lager graphs, and to select cases in which the algorithm is helpful. In particular, it is very important to understand how to select the values of algorithm’s parameters. Another direction is to try to apply this approach in other structures, for example, in de Bruijn graphs.

ACKNOWLEDGEMENT

The authors would like to thank Daniil Chivilikhin, Alexey Sergushichev, Anton Alexandrov and Fedor Tsarev for their helpful comments and reviews.

This work was partially financially supported by the Government of Russian Federation, Grant 074-U01.

REFERENCES

Alexandrov, A. et al., 2012. Combining de Bruijn graph, overlaps graph and microassembly for de novo genome assembly. *Proceedings of «Bioinformatics 2012»*. Stockholm, Sweden, p. 72.

Bankevich, A. et al., 2012. SPAdes: A New Genome Assembly Algorithm and Its Applications to Single-Cell Sequencing. *Journal of Computational Biology,* Vol. 19(5), pp. 455–477.

Bradnam, K.R. et al., 2013. Assemblathon 2: evaluating de novo methods of genome assembly in three vertebrate species. *GigaScience*, Vol. 2 Issue 10.

Lai, B. et al., 2012. A de novo metagenomic assembly program for shotgun DNA reads. *Bioinformatics*, Vol. 28, No. 11, pp. 1455–1462.

Luo, R. et al., 2012. SOAPdenovo2: an empirically improved memory-efficient short-read de novo assembler. *GigaScience*, Vol. 1 Issue 18.

Ng, E.Y.K, Pang, M.P., 2010. Comparison of nucleotide DNA alignment search programmes. *Int. J. Medical Engineering and Informatics*, Vol. 2, No. 2, pp.163–176.

Pevzner, P.A., Tang, H., 2001. A de novo metagenomic assembly program for shotgun DNA reads. *Bioinformatics*, Vol. 17, Suppl. 1, pp. 225–233.

Ridley, M., 2000. *Genome*. HarperCollins Publishers, New York, USA.

Simpson, J.T., Durbin R., 2011. Efficient de novo assembly of large genomes using compressed data structures. *Genome Res*, Vol. 22(3), pp.549–556.

Soh, D. et al., 2007. Enabling more sophisticated gene expression analysis for understanding diseases and optimizing treatments. *ACM SIGKDD Explorations Newsletter*, Vol. 9 Issue 1.