A Short Sequence Splicing Method for Genome Assembly Using a Three-Dimensional Mixing-Pool of BAC Clones and High-throughput Technology

Xiaojun Kang[1,2,3], Cheng Yang[2], Xuguang Zhao[2], Weiwei Chen[2], and Sifa Zhang[2,*], Yaping Wang[3,*]

[1]*State Key Laboratory of Biogeology and Environmental Geology, Wuhan Hubei 430074, P.R. China*
[2]*School of Computer Science, China University of Geosciences (Wuhan), Wuhan Hubei 430074, P.R. China*
[3]*State Key Laboratory of Freshwater Ecology and Biotechnology, Institute of Hydrobiology, Chinese Academy of Sciences, Wuhan Hubei 430072, P.R. China*

Abstract: Current genome sequencing techniques are expensive, and it is still a major challenge to obtain an individual whole-genome sequence. To reduce the cost of sequencing, this paper introduced a high-throughput sequencing strategy using a three-dimensional mixing-pools based on the cube. Following the strategy, BAC clones were injected into each vertex of the cube, and sequencing of each plane provided information about multiple clones, thereby significantly reducing the cost of sequencing. In addition, Velvet was used to assemble the sequencing data. The scaffold generated from Velvet contained a number of contigs, which were orderless. Therefore, to address this problem, a scaffold assembly algorithm based on multi-way trees was used. The algorithm used a multi-way tree to build the framework of chromosomes, and subsequently, the frame was filled to complete the scaffold assembly. This algorithm alone outperformed Velvet in the assembling of a scaffold.

Keywords: BAC clone data, scaffold assembly algorithm, three-dimensional mixing-pool.

1. INTRODUCTION

Next generation sequencing (NGS) technology has a had a tremendous impact on the biological sciences as it has allowed scientists to perform whole-genome scatter (WGS) sequencing on almost any organism. Compared to Sanger sequencing technology [1], NGS technology is faster, with millions of reads generated at once [2], and cheaper, yet the cost is still expensive for an individual study. Meanwhile, other factors such as short read lengths (usually only 35~100bp [3]), large amount of generated data, and a high error rate make WGS sequence splicing and assembly difficult [4, 5].On the other hand, the complex structure of a genome sequence itself also makes short sequence splicing difficult [6]. To obtain better assembly, the current software [7] requires data for sequencing, from previously established databases of multiple insertion length, or even from partial Sanger sequencing, to make the assembly cost effective.

In order to reduce the cost of sequencing, a high-throughput sequencing strategy using three-dimensional mixing-pool, named as Cube, was designed and implemented along with the zebrafish chromosome data that was used to verify the effectiveness of the strategy. The results showed that the strategy can effectively obtain sequencing data and reduce the cost of sequencing. However, to precisely assemble contigs obtained by Velvet splicing [8, 9], a scaffold assembly algorithm based on multi-way tree was utilized, and multiple sets of data were used for the actual measurement and analysis of the algorithm. The results demonstrate that the algorithm showed an excellent assembly performance.

*Address correspondence to these authors at the School of Computer Science, China University of Geosciences (Wuhan), Wuhan Hubei 430074, P.R. China; Tel: +86 27 67883716; E-mail: zhangsifa@cug.edu.cn and State Key Laboratory of Freshwater Ecology and Biotechnology, Institute of Hydrobiology, Chinese Academy of Sciences, Wuhan Hubei 430072, P.R. China; Tel: +86 27 68780751; E-mail: wangyp@ihb.ac.cn

2. CONSTRUCTION AND ANALYSIS OF THREE-DIMENSIONAL MIXING-POOL

2.1. Construction and Sequencing of Three-Dimensional Mixing-Pool

To verify the effectiveness of a sequencing program, a three-dimensional mixing-pool was utilized, as shown in Fig. (**1**). One pool was constructed using roughly 850 Mb of sequence from the zebrafish genome. Based on preliminary findings, the study design required a three-dimensional mixing-pool of 18×18×18, with a total of 5832 BAC clones, each occupying a vertex on a three-dimensional cube. The three-dimensional cube was constructed with 54 (18×3) planes, and each plane (mixed 18×18=324 clones) was used as a DNA sample for sequencing. By sequencing each plane, the sequencing data of 5832 BAC clones was obtained, thus greatly reducing the cost of sequencing. The specific cube preparation steps are described as follows:

(1) Firstly, 54 planes were prepared which were divided into X, Y, Z.

1874-0707/15 2015 Bentham Open

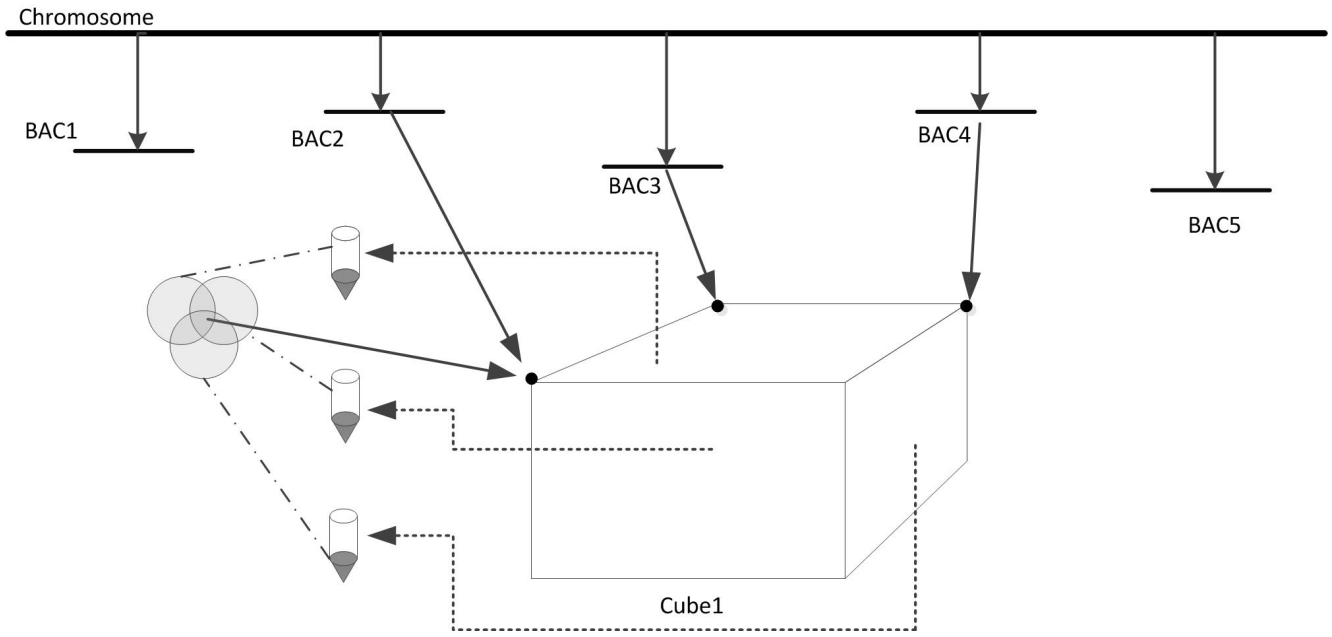

Fig. (1). BAC pooling and DNA extraction. DNA is extracted after pooling each plane(X,Y,Z),then sequence tags are assigned to individual BACs based on presence of three mutually perpendicular planes.

(2) The 5832 BAC clones were placed respectively into the 5832 filling slots of the X-axis, and then respectively into the 5832 filling slots of the Y-axis, and finally, the 5832 filling slots of the Z-axis.

(3) Paired-end sequencing was performedfor each plane of BAC clones.

As each BAC clone was simultaneously present in each plane, the sequence data of each clone could be obtained by calculating the intersection of three mutually perpendicular planes. This paper obtained two sequencing databases when sequencing the cube: a paired-end database with the insertion length of 1 kb and a paired-end database with the insertion length of 8 kb. The sequence length of each database was 100 bp. Finally, the two databases were used for contig splicing. The 8 kb database was used to build a scaffold frame. Then 1 kb database was used to fill the scaffold framework and finalize the splicing of clones.

2.2. Analysis of Single BAC Clone

To reduce the cost of sequencing, multiple BAC clones were placed on one plane at a time for sequencing. Therefore, it was necessary to analyze the cube after the completion of sequencing and each BAC clone was extracted to ensure accuracy . Extraction of each BAC clone from the cube was based on each vertex being an intersection of three mutually perpendicular planes. Therefore, each BAC clone could be obtained by calculating three mutually perpendicular planes in 54 planes. However, BAC clones obtained in this way contained many impurities. For example, a number of paired-end reads that did not belong to a particular clone might also be included. Therefore, after extracting BAC clones from the cube, it was necessary to purify them. During the purification operation, all paired-end reads that existedin multiple clones were deleted, thus all preserved paired-end reads only appeared once in the vertices of one cube. The specific methods of extraction and purification are detailed in the following sections.

2.2.1. Preliminary Extraction of BAC Clones

There is a basic principle of analysis for single BAC clones in a three-dimensional mixing-pool: three mutually perpendicular planes intersect at a point, and three planes of data (a mixing-pool) intersect at single data set (single BAC clone). Therefore, a clone that simultaneously exists on three associated planes belongs to a data set of these corresponding clones. Based on this principle, all clone data can be obtained from X0, Y0, Z0 to X17, Y17, Z17 using the intersection calculation for each of the mutually perpendicular planes.

2.2.2. Purification of BAC Clones

The previous section described the preliminary extraction of BAC clones, namely the tags included in each BAC clone which were parsed out of the cube. For each BAC clone, the operation is guaranteed to extract all the tags contained in it, but the extraction operation also brings a serious problem: for a particular BAC clone, some tags that do not belong may also be extracted from the data set, and this contamination will negatively affect the subsequent cloning assembly operations. To avoid this contamination phenomenon, the initially extracted clones were purified.

Table 1. Analysis on extraction results of BAC clone.

Cube Number	Number of the Initial Tags	Number of the Parsed Tags	Proportion
1	17,496,000	1,289,659	31.77%
2		1,951,817	54.33%
3		1,483,333	38.75%
4		1,966,490	48.67%
5		1,945,673	44.26%
6		3,463,959	36.69%
7		2,843,583	47.48%
8		3,520,113	43.77%

Table 2. Proportion of the effective clones in cube.

Cube Number	Theoretical Number of the Clones	Number of the Parsed Clones	Number of Effective Clones	Proportion
1	5832	5018	2861	49.06%
2		5326	3005	51.53%
3		5213	3046	52.23%
4		5089	2879	49.37%
5		4981	2914	49.97%
6		5032	2977	51.05%
7		5107	3004	51.51%
8		4924	3102	53.19%

This quality checkpoint deletes all tags appearing in multiple clones. The steps implemented for all tags in the 5832 BAC clones are as follows:

(1) For each tag, the sum N of ASCII code of each character in the tag was calculated, followed by the calculation of number of file mapping by the tag (k) .

(2) All tags were divided into 200 small files followed by step (1). The record form of tag in each file contained following data: tag sequence, and the clone number where the tag existed.

(3) The data in each small file was sorted out according to the tag sequence placing common tag sequences adjacent to each other.

(4) A pairwise alignment operation between the sorted tags and the adjacent tags was then performed, and repeated tag numbers were removed, thereby purifying BAC clone data from being contaminated.

(5) The purified data were then restored to 5832 BAC clones based on tag location.

2.3. Test Results and Analysis

According to the extraction and purification methods described above, eight cubes were parsed successively. To assess the effect of data parsing, the proportions of parsed data acquired from the original data are as shown in Table **1**. For each clone, the number of the parsed tags must be more than 500, for the assembly .

As shown in Table **1**, the proportion of parsed tags for each cube was only about 40% of the tags after processing normalization, which were caused by the characteristics of the cube and repetitive sequences. Although parsing decreased the data set, the analytical results of eight cubes provided sufficient coverage of the test data: each cube initially covered 80% of the test data and the parsed data of each cube covered more than 30% of the test data. As a result, eight cubes provided 2.5X coverage of the test data.

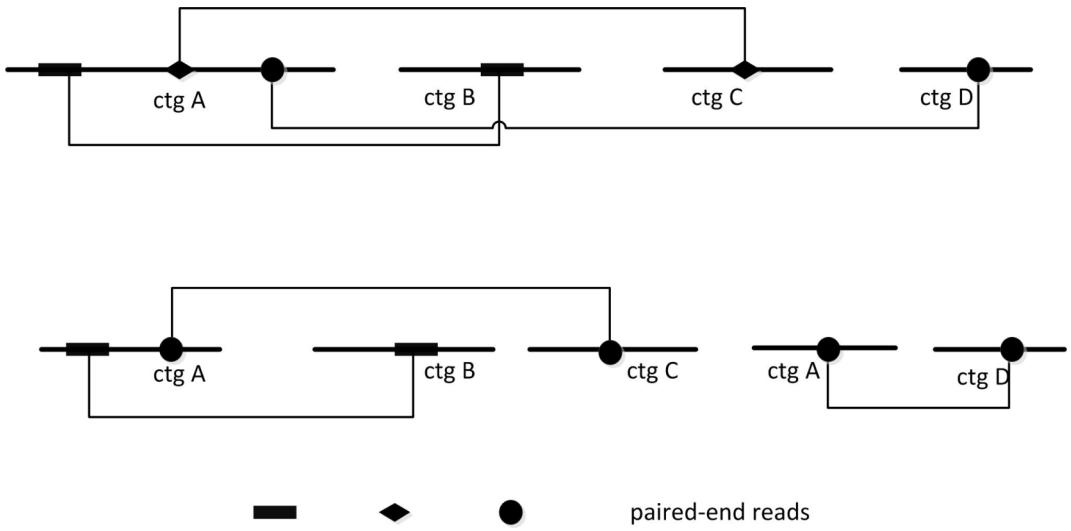

Fig. (2). Scaffold assembly when affected by repetitive sequence. The correct order of 4 contigs is A-B-C-D. They are connected by some paired-end reads. When there are repetitive sequences, the order may chaotic, and the result of assembly is A-B-C-A-D.

On the other hand, approximately 50% of the effective clones were parsed out of each cube, implying that half of the cube data covered 80% of the test data with each cube covering roughly 40% of the test data (Table **2**). Thus, eight cubes provided 3.2X coverage of the original test data, which was sufficient to assemble the test data. Therefore, high-throughput sequencing using a three-dimensional mixing-pool could effectively restore the test data at a significantly reduced cost.

3. SCAFFOLD ASSEMBLY ALGORITHM BASED ON MULTI-WAY TREE

3.1. Problem Analysis

Paired-ends are used to assemble contigs into a scaffold. This process identifies shared sequences to splice overlapping contigs together. One contig (A) may have a number of paired-end corresponding to multiple contigs (B, C, D) as shown in Fig. (**2**). The order of these contigs corresponding with A must be determined. Under normal conditions, the order is determined by other sequence in B, C, and D compared to A. Sometimes, however, repetitive sequences can lead to errors in alignment. The order of four contigs A, B, C and D in the chromosome is A-B-C-D, butbecause ctg A had repetitive sequences, incorrect results could be obtained (A-B-C-A-D) if these were spliced according to normal conditions. To avoid this error, we utilized a scaffold assembly algorithm based on a multi-way tree.

3.2. Establishment of Multi-Way Tree

To avoid splicing errors caused by repetitive sequences, scaffold assembly was completed by creating a multi-way tree to establish the overall framework of chromosomes and then fill the frame. Two paired-end databases were used during the process of contig stitching and scaffold assembly. The insertion length of the first database was 1KB, whereas the insertion length of the second database was 8KB. However, the sequence length in each database was 100bp. The 8KB insertion length database was used to create the multi-way tree, and the 1KB insertion length database was used to fill the frame. A flow chart describing the establishment of the multi-way tree is shown in Algorithm **1**.

Algorithm 1:

Input:

paired-end reads which insertion length is 1KB and 8KB

Output:

Scaffold established with multi-way tree

1 Select the longest contig as a initial node

2 Search the child node that has a distance of 8kb from the initial node

3 **while** there is a child node **do**

4 Search the child node of the next level for each child node

5 **end**

Each contig functioned as a node. To begin, the longest contig was selected as an initial node. Following this, a child node roughly 8KB from the initial one was chosen through paired-end search. An ordinal relation between these child nodes and the initial nodes was observed and, some may be located towards the left or right of an initial node. For this study, only the nodes located towards the right of the initial one were used for extension. This process continued for each of the child nodes, continuing selection for extension towards the right. Eventually, a subsequent child node could not be identified, and the path was terminated. Circulation, search, and extension were conducted in this way until all the paths were terminated. Following this, the same method was used to search and extend the paths to the left of the initial nodes. The path with the maximum number of nodes (*i.e.* the longest path) was chosen as an overall framework of the chromosome.

Table 3. Contrast experiment between scaffold assembly algorithm based on multi-way tree and velvet algorithm.

Data Sources	Ctg Quantity	Assembly Length of Algorithm in this Article	Assembly Length of Velvet
Chromosome 22	108	117,573	30,795
Chromosome 22	86	108,798	29,865
Chromosome 22	87	122,379	59,884
Chromosome 24	95	120,893	40,702
Chromosome 24	113	121,759	32,716
Chromosome 24	107	122,678	59,937
Chromosome 24	99	119,374	21,329
Chromosome 25	64	988,595	60,016
Chromosome 25	97	100,987	49,057
Chromosome 25	106	122,336	38,467

3.3. Filling of Multi-Way Tree

After completing the overall framework, it was filled in with further extensions where, each node in the framework was referred to as a stub. For each stub, paired-end was used to find contigs roughly 1KB from the stub. If multiple contigs were found, the ordinal relationship between these contigs and stub was determined. By filling out each stub in this process, the assembly of whole chromosome was completed.

3.4. Test Results and Analysis

To validate the correctness and validity of the scaffold assembly algorithm using a multi-way tree, four sets of data in chromosome 22, 24, and 25 of zebrafish were selected for analysis. The test data and results generated by Velvet are shown in Table **3**. In addition, a total of ten sets of data from multiple fish chromosomes were used to compare assembly by velvet with multi-way tree scaffold assembly. The "Ctg quantity" column indicates the number of contigs used to assemble the scaffold, which were obtained by velvet splicing paired-end. Most contig lengthswere very short because a high number were obtained by velvet splicing; therefore, only those contigs longer than 100 bp were selected for scaffold assembly.

Of the 10 sets of test data, the assembled length almost reached 100KB which is very close to the original. Therefore, the algorithm used in this paper sufficiently assembled a correctly sequenced scaffold. However, the algorithm does have some shortcomings, for example, chromosome ends may not be assembled. This problem will require subsequent work and may be solved using multiple paths from multi-way tree analysis rather than simply selecting the longest path.

The assembled length using velvet was substantially less than 60KB while the algorithm developed in this study generally assembled a construct greater than 100KB. Therefore, this algorithm not only corrected splicing error created by Velvet software but also greatly improved the assembled length and demonstrated good splicing performance.

CONCLUSION

This study described the design and implementation of a high-throughput sequencing strategy using a three-dimensional mixing-pool and scaffold assembly with a multi-way tree-based algorithm. The experimental results show that the strategy was feasible, and the method reduced the cost of sequencing while improving assembly. This provides a new direction for the improvement of genome sequencing and might greatly reduce costs, thus contributing to the development of genome sequencing projects.

CONFLICT OF INTEREST

The authors confirm that this article content has no conflict of interest.

ACKNOWLEDGEMENTS

The project was supported by the National Natural Science Foundation of China under Grant 31372573, State Key Laboratory of Freshwater Ecology and Biotechnology under Grant 2013FB04, State Key Laboratory of Biogeology and Environmental Geology China University of Geosciences (No.GBL31506), and the National Science Foundation for Post-doctoral Scientists of China under Grant 2014M552119, Supported by Wuhan Branch, Super computing Center, CAS, China.

REFERENCES

[1] Sanger F, Nicklen S. Coulson AR. DNA sequencing with chainterminating inhibitors. Proc Natl Acad Sci USA 1977; 74(12): 5463-7.

[2] Metzker ML. Sequencing technologies-the next generation. Nat Rev Genet 2010; 11(1): 31-46.

[3] Steven L, Phillippy AM, Zimin A, *et al*. GAGE: A critical evaluation of genome assemblies and assembly algorithms. Genome Res 2012; 22: 557-67.

[4] Li R, Fan W, Tian G. The sequence and de novo assembly of the giant panda genome. Nature 2010; 463: 311-7.
[5] Star B, Nederbragt AJ, Jentoft S. The genome sequence of Atlantic cod reveals a unique immune system. Nature 2011; 477: 207-210.
[6] Schatz MC, Delcher AL, Salzberg SL. Assembly of large genomes using secondgeneration sequencing. Genome Res 2010; 20(9): 1165-73.
[7] Fan W, Li R. Test driving genome assemblers. Nature Biotechnol 2012; 30(4): 330-331.
[8] Zerbino DR, Birney E. Velvet: algorithms for de novo short read assembly using de Bruijn graphs. Genome Res 2008; 18: 821-9.
[9] Pevzner PA, Tang H, Waterman MS. An Eulerian path approach to DNA fragment assembly. Proc Natl Acad Sci 2001; 98: 9748-53.

Received: May 16, 2015 Revised: August 23, 2015 Accepted: September 31, 2015