Bioinformatics
doi.10.1093/bioinformatics/xxxxxx
Advance Access Publication Date: Day Month Year
Manuscript Category

OXFORD

Course Project of Big Data for Biological Applications

Reusing Colors in Rainbowfish

Xun Zhao [1,*]

[1] College of engineering, University of Florida, Gainesville, 32601,USA.

Abstract

Motivation: Within the last two decades, assembling a genome from enormous amount of reads from various DNA sequences has been one of the most challenging and important computational problems in molecular biology, Due to the vast amount of genome data sequencing by NGS machine, the comparing step for all the pairs of reads becomes extremely difficult. Also, Most NGSs cannot read long DNA fragments, and their read lengths are not long enough to detect overlaps between reads, so De Bruijn Graph and colored De Bruijn Graph were introduced. But the space requirement is gigantic, so many succinct structure to store them have been developed, one of them is called rainbowfish, though it can saving a lot of space, but still, there's a way to improve it using a color reusing algorithm.
Results: Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text
Availability: Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text
Contact: xun.zhao@ufl.edu

1 Introduction

Time goes by, 20 years ago, as a creative and intelligence approach, Idury and Waterman [1] firstly introduced and proposed the de Bruijn graph as the data structure to the biological field which finally becomes an important dominant way of modern genomics according to the genome assembly technology in next-generation sequencing applications[2,3,4]. In the biological applications, the de Bruijn graph data structure will produce a huge number of short sequence fragments or pieces such as much succinct representation which could show the complete graph by using small required space for the data storage quickly navigation. How should we do to obtain a de Bruijn graph? Generally, such as genome sequence reads could be represented as the set of sequence strings as mathematical way S in a korder de Bruijn graph in S. The designed de Bruijn graph has distinct vertex v for the different k-1 mersbe represented as k-1 length sequence substring to every node in this whole string. It can also provide a directed edge u v represented by the prefix u of k - 1mer and suffix v for k-1mer. For a more appropriate understanding, we can assume to set the Nk to denote corresponding the sequence set of k-mers. So that the edge contact both the node u in Nk to node v in Nk as if there is sequence has the k+1 length for an S that can construct the S collection to represent the whole overlapping pieces in the DNA small fragments. In the genome sequence detection, our ultimate goal is to find out these overlapping DNA pieces in the very long DNA to be shown as well in our applications. We can see more details and demonstration on the graph for a deeper understanding of the process and definition of the de Bruijn graph construction and principle. Although the de Bruijn graph data structure played the vital role in the next generation sequence scientific method and improved detecting and genotyping accuracy with straightforward and complex genetic variants in an individual or population, the requirement for storage of the biological data still become a big problem due to the large scale of computational complexity. It will be harmful to both biological and computational perspective of this applicable operations. For the biological lab operation, the human genome detecting when encoded with de Bruijn graph also required more than hundreds of gigabytes (GB) storage for the normal experiment. To solve this biological applications problem actually, we aim to increase the memory space for usage of sequence DNA drives us to obtain a more efficient and powerful data structure colored de Bruijn graph. As the variant of classic structure, we are dedicated to adding more additional distinguish method information into de Bruijn graph that we can efficiently encode the data for smaller required space for storage. The new colored de Bruijn graph represents a succinct version of de Bruijn graph. This colored de Bruijn graph can not only reduce the amount of data storage memory efficiently though there is some penalty as the more time for running the operations, which could allow the graph to applied in the apparently larger and more ambitious sequence projects than the past biological application program. For the details, every distinct edge will be distinguished in sets of color. Every color is created to encode the information of every k-mers in the graph. Because of such powerful more space storage, this colored method becomes much popular in the biological application in the world which can eventually not only detect the simple or

more complex genome information but also can represent the distinct gene and assemble the genome of the individual or the population in a highly accurate performance. (**?**) Specifically, rather than using the hash table for the de Bruijn graph topology, Muggli et al. [6] introduced the efficient VARI representation to the colored de Bruijn graph, which collaborates with BOSS representation by saving the memory space though sacrifice the speed of operations. In this proposal, we want to introduce a creative way of reusing the colored de Bruijn graph to decrease the requirement for genome information storage and improvement the accuracy of the colored de Bruijn graph.

2 Related work

Rainbowfish is a succinct representation of the color information, and uses rank and select operations to lookup the color class corresponding to k-mers in the de Bruijn graph. Ranka(i) returns the number of occurrences of symbol a on the range [0, i], whereas selecta(i) returns the position of the ith occurrence of symbol a. And it is also a representation of colored de Bruijn graph. It is introuduced by Iqbal et al. (Nature Genetics, 2012 as a variant of the de Brujin graph, where each edge is related to some set of colors. And it is aimed at "detecting and genotyping simple and complex genetic variants in an individual or population". The memory usage of the colored de Bruijn graph representation adopted in Cortex precludes this approach from being adopted when the underlying genomes and color sets become too large. While, Because of the bottleneck of space and time, it is a big problem for maintenance and navigation of the De Brujin Graph in genome assembly. In recent years, a lot of research have been done to solve this problem [2]. For example, Holley et. al. proposed the Bloom Filter Trie, which also is a succinct data structure for the colored de Bruiin graph. SplitMEM is a related algorithm to create a colored de Bruijn graph from a set of suffix trees representing the other genomes.

3 Approach

In this section, we have a short introducion about de Brujin grpah and BOSS presentation first, next, we describe the colored de Bruijn graph and VARI representation, then introduce a succinct representation of the color information structure called rainbowfish. Finally proposing a possible approach for Reusing Colors in Rainbowfish so that storage of the colors of the graph can be further reduced by reducing the number of colors used.

3.1 BOSS representation

Within the last two decades, assembling a genome from enormous amount of reads from various DNA sequences has been one of the most challenging and important computational problems in molecular biology. Though the problem is proved to be NP-hard [3], many algorithms have been proposed for the problem. Most old-time algorithms (especially for the long Sanger reads) construct "overlap graph" after finding the overlapping pairs of reads. But this strategy is difficult to apply when using a huge amount of data from more recent epoch-making next-generation sequencer(NGSs). Due to the vast amount of genome data sequencing by NGS machine, the comparing step for all the pairs of reads becomes extremely difficult. Also, Most NGSs cannot read long DNA fragments, and their read lengths are not long enough to detect overlaps between reads. That's why "De Bruijn Graph" being introduced. A de Bruijn graph is a graph where each edge represents a k-mer (a substring of length k) that exists in the reads, and one vertex that connect this edge labelled as prefix (k-1)-mer (a substring of length k-1), the other vertex that connect this edge labelled as suffix (k-1)-mer (a substring of length k-1). Glue all vertices that have the same label. Intuitively, de Bruijn graph can be constructed more efficient that overlap graph, but the overlap phase still a bottleneck for de Bruijn graph due the large amount requirement of memory usage. Then, the introduce of BOSS representation of de Bruijn graph only require $m(2 + \log \sigma)$ bit to store where m is the number of edges in the de Bruijn graph and is the alphabet size (i.e. $\sigma = 4$ in the case of DNA). The size of this presentation is not affected by the value of k and is very small. [4]

3.2 VARI representation

Around 2012, with the introduction of colored de Bruijn graph [5], a variant structure of de Bruijn graph, which is used for detecting and genotyping simple and complex genetic variants in an individual or population. A succinct representation of colored de Bruijn Graph was introduced. [6] And this data structure for colored de Bruijn graph is based on BOSS representation for de Bruijn graph. It reduces a large number of memory usage to store the coloured de Bruijn graph, just like BOSS save a lot of memory usage to store de Bruijn graph.

First, let's have a quick introduction about what is a colored de Bruijn graph. The basically structure of colored de Bruijn graph is the same as the classical de Bruijn graph, however, each vertex ((k-1) mers) and edge (k-mers) is associated a list of colors corresponding the sample that vertex or label exists. For example, there is a set of n samples, and each sample corresponding to a de Bruijn graph. Given a set C of n colors c_1,c_2,c_3,...c_n where c_i corresponds to sample i where all its relevant k-mers and (k-1) mers are colored with c_i. A colored de Bruijn graph may consist many graphs and its corresponding color, except overlapping nodes.

3.3 Rainbowfish

In another succinct representation of coloured de Bruijn graph, which can be called Rainbowfish, it also adopts the BOSS representation of the de Bruijn graph topology, and compress the stored color information of a graph even further to achieve a more space usage. [7]

We basically use the same structure as Rainbowfish, but we came up with a new strategy to reuse colors in the construction of the graph.

Here, we briefly introduce the definition of rank and select operation that are used to look up the color class corresponding to k-mers in the colored de Bruijn graph.

Rank and select are operations that are commonly used for navigating within succinct data structures. For a bit vector $B[0, 1, ..., n-1]$, rank(i) returns the number of 1s in the prefix $B[0, 1, ..., i]$ of B. select(j) returns the position of jth 1, which is, the smallest index i such that rank(i) = j. For example, for the 12-bit vector $B[0, ..., 11]$ =100101001010, RANK(5) = 3, because there are three bits set to one in the 6-bit prefix $B[0, ..., 5]$ of B, and SELECT(4) = 8, because B[8] is the fourth 1 in the bit vector.

Now, let's have short introduction for rainbowfish:

There are two fundamental observations for Rainbowfish's compact representation of color information:

First, it is often the case that many of the k-mers in a colored de Bruijn graph share the same set of colors. So, let's define an equivalence relation over the set of k-mers in de Bruijn graph, and denote Col(.) as the function that maps each k-mer to its corresponding set of colors. If and only if Col(k_1) = Col(k_2), we say k_1 k_2($k_1$and k_2 "are color-equivalent "), and we refer the set of colors shared by k_1 and k_2 color class.

Second, the color classes distribution frequency is not uniform. It would be useful if we use a small number of bits represent color classes occurring frequently, and use larger number of bits represent color classes occurring less frequent. (The approach is similar to Huffman code)

In Rainbowfish, the color class representation has three components which are derived from color matrix. Color matrix is used to store <k-mers, color-set> pairs from colored de Bruijn graph. It stores the mappings between labels and color classes in an equivalence class table(ECT). As labels are assigned sequentially, this is simply an array of bit vectors

encoding the corresponding color sets. Except equivalence class table, Rainbowfish maintains other two bitvectors, a boundary bit vector (BBV) and a label bit vector (LBV).

All color classes that stored in the equivalence class table (along with their corresponding variable-length labels). But how do we store a mapping from k-mers to the variable-length labels? That's where label bit vector comes in, the labels are stored in the order in which k-mers are stored in the rank order induced by BOSS. However, it's hard to read the label corresponding to the k-mer of a specific rank because its variable length. Since we don't know where it begins and where it ends. Hence, another bitvector—boundary bit vector is used to mark the boundary of each variable-length label in LBV. It is the same size as LBV and has a bit set to 1 at each index where a new label starts in LBV.

3.4 Reusing color in Rainbowfish

In our color-reusing algorithm, we are trying to reduce the number of used color set at color matrix.

Every time a new read came in, first we insert all the k-mers from R into colored de Brujin graph G with some special temporary color t. if there are some k-mers from R can't fit into graoh G, then we randomly assign any color set c from G to them, in this case, there are no new color come in, so color reduced. Otherwise we find all colors of their neighboring k-mers, if there's no color c among their neighbor except those k-mers from R, then we use color c on those k-mers from R yet already inserted into G instead of color t.

4 Evaluation

4.1 Evaluation plan

In our project, we mainly measure the space usage. Since color de Bruijn graph is a succinct data structure, we could measure the space usage by measuring the space usage of ECT, LBV and BBV on disk. In other words, the space cost in constructing the underlying de Bruijn graph is not included. For comparision, we prefer compare out data structure with the uncompressed color matrix, VARI and regular rainbowfish data structure. Here the uncompressed color matrix is generated by VARI. The space usage of VARI just includes the space spent on color class information on disk. In both method, the space usage does not include the space used by BOSS which is the representation of the underlying de Bruijn graph.

4.1.1 Pass algorithm

We adopt a 2-Pass algorithm to construct our data structure. In the first pass, we will perform the following four step:

- 1. Read the uncompressed color matrix.
- 2. Compute the distinct color classes and count frequency of each class.
- 3. Order classes in descending order based on their frequency.
- 4. Assign labels to color classes.

After the first pass, we could get the ECT. In the first, to count the frequency of each color class, we prefer to use a sperse hash table since it could help us save a lot of memory. In the harsh table, the color class is the key and its frequency is the value. Then we start the second pass:

- 1. Read the uncompressed color matrix again.
- 2. Add the label of each k-mer to the label bit vector(LBV).
- 3. Build the boundary bit vector(BBV).

After the second pass, we could get the LBV and BBV . Then we save ECT , LBV and BBV on disk to measure the space usage.

4.2 Data

For this project, to measure the space usage of color de bruijn graph, we consider using the following four datasets which is shown in table 1. the first dataset is the E. coli datasets. Originally, this dataset consists of 5598 different strains. We will generate samples from this dataset. The sample sizes are 100, 1000, 2000, 3000, 4000, 5000 and 5598. By this way, we could measure the scalability of our project.We also perform experiments with k-mer size is 63 to test the performance of our data structure when K is higher. Another dataset we want to try is beef safty. This dataset consists of 87 metagenomic samples hence this dataset yield a complex colored de Bruijn graph. So we used this dataset in our project. Finally, we consider building the colored de Bruijn graph on the human transcriptome dataset. This dataset consists of $95,000$ colors, but only 159 million k-mer. Hence this dataset could give us an brief idea about how our data structure will perform when the number of colors becomes very large but the number of k-mer is relatively small.

Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text. Figure 2 shows that the above method Text Text Text Text Text Text Text Text Text Text Text Text. **?** might want to know about text text text text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text. Figure 2 shows that the above method Text Text Text Text Text Text Text Text Text Text Text Text. **?** might want to know about text text text text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text. Figure 2 shows that the above method Text Text Text Text Text Text Text Text Text Text Text Text. **?** might want to know about text text text text

Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text. Figure 2 shows that the above method Text Text Text Text Text Text Text Text Text Text Text Text. **?** might want to know about text text text text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text. Figure 2 shows that the above method Text Text Text Text Text Text Text Text Text Text Text Text. **?** might want to know about text text text text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text. Figure 2 shows that the above method Text Text Text Text Text Text Text Text Text Text Text Text. **?** might want to know about text text text text

Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text. Figure 2 shows that the above method Text Text Text Text Text Text Text Text Text Text Text Text. **?** might want to know about text text text text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text. Figure 2 shows that the above method Text Text Text Text Text Text Text Text Text Text Text Text. **?** might want to know about text text text text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text. Figure 2 shows that the above method Text Text Text Text Text Text Text Text Text Text Text Text. **?** might want to know about text text text text

Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text. Figure 2 shows that the above method Text Text Text Text Text Text Text Text Text Text Text Text. **?** might want to know about text text text text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text. Figure 2 shows that the above method Text Text Text Text Text Text Text Text Text Text Text Text. **?** might want to know about text text text text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text. Figure 2 shows that the above method Text Text Text Text Text Text Text Text Text Text Text Text. **?** might want to know about text text text text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text. Figure 2 shows that the above method Text Text Text Text Text Text Text Text Text Text Text Text.

Table 1. This is table caption

head1	head2	head3	head4
row1	row1	row1	row1
row2	row2	row2	row2
row3	row3	row3	row3
row4	row4	row4	row4

This is a footnote

Fig. 1. Caption, caption.

? might want to know about text text text text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text.

Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text. Figure 2 shows that the above method Text Text Text Text Text Text Text Text Text Text Text Text. **?** might want to know about text text text text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text. Figure 2 shows that the above method Text Text Text Text Text Text Text Text Text Text Text. **?** might want to know about text text text text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text. Figure 2 shows that the above method Text Text Text Text Text Text Text Text Text Text Text. **?** might want to know about text text text text

Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text. Figure 2 shows that the above method Text Text Text Text Text Text Text Text Text Text Text Text. **?** might want to know about text text text text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text. Figure 2 shows that the above method Text Text Text Text Text Text Text Text Text Text Text. **?** might want to know about text text text text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text. Figure 2 shows that the above method Text Text Text Text Text Text Text Text Text Text Text. **?** might want to know about text text text text

4.3 Test1

Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text. Figure 2 shows that the above method Text Text Text Text Text Text Text Text Text Text Text Text. **?** might want to know about text text text text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text. Figure 2 shows that the above method Text Text Text Text Text Text Text Text Text Text Text. **?** might want to know about text text text text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text. Figure 2 shows that the above method Text Text Text Text Text Text Text Text Text Text Text. **?** might want to know about text text text text

5 Discussion

Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text. Figure 2 shows that the above method Text Text Text Text Text Text Text Text Text Text Text Text. **?** might want to know about text text text text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text. Figure 2 shows that the above method Text Text Text Text Text Text Text Text Text Text Text. **?** might want to know about text text text text Text Text Text Text Text Text Text Text Text Text.

Table 1 shows that Text Text Text Text Text Text Text Text Text Text. Figure 2 shows that the above method Text Text. Text Text Text Text Text Text Text Text. Figure 2 shows that the above method Text Text. Text Text Text Text Text Text Text Text Text. Figure 2 shows that the above method Text Text.

6 Conclusion

(Table 1) Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text. Figure 2 shows that the above method Text Text Text Text Text Text Text Text Text Text Text Text. **?** might want to know about text text text text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text. Figure 2 shows that the above method Text Text Text Text Text Text Text Text Text Text Text. **?** might want to know about text text text text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text. Figure 2 shows that the above method Text Text Text Text Text Text Text Text Text Text Text.

Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text. Figure 2 shows that the above method Text Text Text Text Text Text Text Text Text Text Text Text. **?** might want to know about text text text text (**?**)

1. this is item, use enumerate
2. this is item, use enumerate
3. this is item, use enumerate

Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text. Figure 2 shows that the

above method Text Text Text Text Text Text Text Text Text Text Text Text. **?** might want to know about text text text text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text. Figure 2 shows that the above method Text Text Text Text Text Text Text Text Text Text Text Text Text. **?** might want to know about text text text text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text.

Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text. Figure 2 shows that the above method Text Text Text Text

Acknowledgements

Text Text Text Text Text Text Text Text. **?** might want to know about text text text text

References

[1]Idury, R., Waterman, M.: A new algorithm for DNA sequence assembly. Journal of Com- putational Biology 2, 291–306 (1995).

[1][1]Muggli, M., Puglisi, S., Ronen, R., Boucher, C.: Misassembly detection using paired-end sequence reads and optical mapping data. Bioinformatics (special issue of ISMB 2015) 31(12), i80–i88 (2015).

[2]Myers, E.W.: Toward simplifying and accurately formulating fragment assembly. Journal of Comutational Biology 2, 275–290 (1995).

[3]Alexander Bowe, Taku Onodera, Kunihiko Sadakane, and Tetsuo Shibuya. Succinct de Bruijn graphs. In Proceedings of the International Workshop on Algorithms in Bioinformatics, pages 225– 235. Springer, 2012.

[4]Iqbal, Z., Caccamo, M., Turner, I., Flicek, P., McVean, G.: De novo assembly and geno- typing of variants using colored de Bruijn graphs. Nature Genetics 44, 226–232 (2012)

[5]Martin D. Muggli, Alexander Bowe, Noelle R. Noyes, Paul Morley, Keith Belk, Robert Raymond, Travis Gagie, Simon J. Puglisi, and Christina Boucher. Succinct Colored de Bruijn Graphs. Bioin- formatics, 2017.

[6]Almodaresi, Fatemeh, Prashant Pandey, and Rob Patro. "Rainbowfish: A Succinct Colored de Bruijn Graph Representation." bioRxiv (2017): 138016.