SyntenyFinder: A Synteny Blocks Generation and Genome Comparison Tool

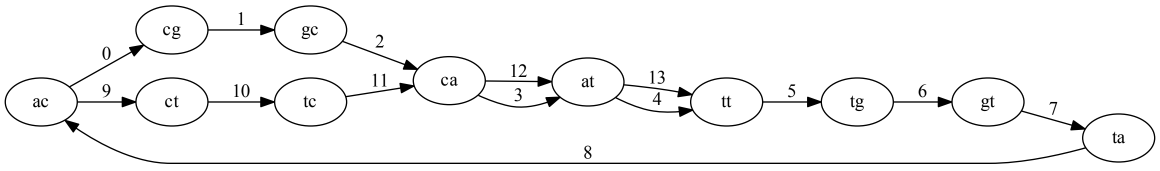
Intern: Ilya Minkin
Advisor: Son Pham

We present an algorithm for finding synteny blocks in genomes. The algorithm is based on colored de Bruijn graphs and graph simplifications. Our method is suitable for finding synteny blocks in genomes that contain regions of highly conserved DNA, i.e. genomes that are evolutionary close to each other. With some modifications the algorithm can be applied in more complicated cases.

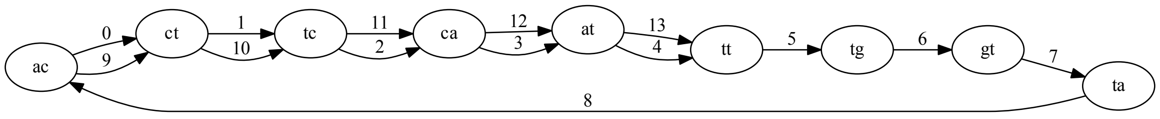
1. Introduction

Recent advances in high throughput sequencing and genome assembling technologies result in appearance of high number of completely sequenced genomes, ranging from bacteria to mammalian [1]. It raises a lot of interesting biological question. For example, it is not obvious how DNA sequence leads to observable phenotype. Comparative genomics addresses this question using the fact that common features of two organisms are often encoded by DNA regions that are conserved within the species [2]. Another question is evolution. It is known that genome rearrangements play important role in adaptation in adaptation of bacteria for different environments [3]. And the more exciting is study of mammalian evolution [4]. Any such research requires a tool that will reveal a structure of the genomes. For example, in order to perform rearrangements analysis, genomes must be decomposed into conservative segments, called synteny blocks. Currently existing tools for solving this problem, like DRIMM-Synteny [6], require genomes to be presented as sequences of enumerated local alignments, or *anchors*.

In this paper we address problem of finding synteny blocks in unannotated sequences. Our approach is based on de Bruijn graphs. De Bruijn graphs are extensively used in bioinformatics for genome assembly [7, 8].



(a) De Bruijn graph built from string "*acgcattgtactcatt*" and $k = 2$. Non-branching paths correspond to multiple copies of the same substrings.



(b) Same de Bruijn graph after simplification. Replacing "*acGca*" by "*acTca*" we obtain long non-branching path that corresponds to the synteny block.

Figure 1: Illustration of de Bruijn graphs and graph simplification

2. Methods

Given a set $S = \{S_1, S_2, \ldots, S_n\}$ of chromosomes, where each chromosome is represented as a string over alphabet $\{A, C, G, T\}$. The task of finding synteny blocks is to find a set of so called conserved regions $C = \{C_1, C_2, \ldots, C_n\}$, where each conserved region C_i is a set of substrings of chromosomes from S. Such regions are supposed to cover most of the genome for closely related species. All substrings forming a conserved region C_i must be similar to each other according to some criterion of similarity. Note that problem of finding synteny blocks in a set of chromosomes is equivalent to a problem of finding synteny blocks in one superchromosome obtained from concatenating all chromosomes from the set – we can just separate chromosomes by special characters.

At this moment there is no generally accepted formal criterion of similarity exist, so the problem of finding synteny blocks is ill-defined. In our work we introduce new criterion of similarity based on de Bruijn graphs and graph simplifications.

As previously mentioned, our method is based on de Bruijn graph. Given a fixed value k and a string S we can build de Bruijn graph G from the string as follows. Let's denote by k-prefix of a string S first k characters of S, and by k-suffix last k characters of S. For each unique substring of length k (called k-mer) found in S, we add a vertex to G and mark it with corresponding k-mer. For each $(k + 1)$-mer w found in S we add edge that connects vertex corresponding to k prefix of w with vertex corresponding to k suffix of w and label the edge with position of first character of w (multiedges with different labels are allowed).

In this graph we consider only paths that have consecutive labels on edges. It is easy to see that with such restriction every path in G corresponds to a substring in S. Example of such de Bruijn graph built from the string $S = $ "*acgcattgtactcatt*" and $k = 2$ is depicted on Figure 1a.

Note that two copies of substring "*catt*" form a non-branching path consisting of edges with multiplicity 2 in this graph. Single mismatch in substrings "*acGca*" and "*acTca*" form so-called "bulge", unoriented cycle generated by two valid paths with coinciding ends. If we replace one branch of the bulge by another (replace "*acGca*" by "*acTca*" for example), we will obtain a long non-branching path (Figure 1b).

This heuristic forms basis of our method – conserved regions in different parts of the genome contain conserved basepairs, but such regions are disrupted by indels and mismatches. These differences form bulges in the graph that make it different to infer structure of the synteny blocks. We remove bulges having size less than some predefined constant and thus obtain non-branching paths corresponding to the conserved regions. The process of removing bulges from the graph is called *simplification.* We sustain one-to-one correspondence between the graph and the string – when we change something in the graph, then we change appropriate characters in the string.

Conserved regions can be located on opposite strands of DNA. To handle this, we use *colored* de Bruijn graphs [8]. Given a string S, for each $(k + 1)$-mer found in S we add corresponding edge to the graph and color it *blue*, for each $(k + 1)$-mer found in reverse-complementary counterpart of S we add corresponding edge to the graph and color it *red.* In this graph, non-branching paths with different colors represent synteny blocks located on opposite strands of DNA.

Complete pipeline is following:
1) Concatenate input chromosomes into one superchromosome
2) Build de Bruijn graph from the superchromosome
3) Simplify the graph
4) Output synteny blocks as non-branching path in the graph

Our algorithm depends on two parameters: k and δ (minimum allowed size of a bulge). It is reasonable to use as high k as possible ($k > 50$) to keep graph structure simple and avoid connecting regions that are actually not homologous. So, our method requires that conserved regions in input genomes contain exact shared k-mers. This is not a problem in genomes that are very close to each other (like different strains of a bacteria), but it can create difficulties in genomes that had separated a long time ago.

This issue can be solved, for example, by finding a set of all local alignments in the genomes and substituting one subsequence in each found alignment by another. In results section we will demonstrate on a practical half-synthetic example that with such modifications our method is able to handle complicated cases. So at this point our method is directly applicable to only evolutionary close genomes and in near future we plan to extend it to wider range of use.

Let's formally describe our algorithm for finding synteny blocks. We are given two numbers k and δ and a set $S = \{S_1, S_2, \dots, S_n\}$ of chromosomes, where each chromosome $S_i = (s_{i,1}, s_{i,1}, \dots, s_{i,m_i})$ is a string over alphabet $\Sigma = \{A, C, G, T\}$. Let's denote by $X_1 \uplus X_2$ concatenation of strings X_1 and X_2. We denote by $X[i, j]$ a substring of a string X that starts at i and ends at j, $X[i,j] = (x_i, x_{i+1}, \dots, x_j)$. $Rev(X)$ means reverse-complementary counterpart of a string X.

First step of the algorithm is to obtain superchromosome $\hat{S} = S_1 \uplus S_2 \uplus \dots \uplus S_n$. Concatenated strings are interleaved by special characters that indicate ends of the chromosomes, but we omit these technical details. Then we build a linked list of pairs $L = ((\hat{s}_1, 1), (\hat{s}_2, 2), \dots, (\hat{s}_m, m))$ and l_i is a pointer to i-th item in the list, $Next(l_i) = l_{i+1}$ is next element after l_i. Let's denote by $l_i[1]$ first element of pair of i-th item of the list ($l_i[2]$ is second element). First element of a pair represents character of the sequence, and second element represents it's original position in string $\hat{S}$.

Colored de Bruijn graph is graph $G = (V, E)$ where $V = \Sigma^k$. Set of outgoing edges from vertex v is denoted $Out(v)$. We define three functions:
1) $Pos : E \to \{l_1, l_2, \dots, l_m\}$
2) $Color : E \to \{Blue, Red\}$
3) $Spell : E \to \Sigma^{k+1}$

For each $i \in \{1, 2, \dots, n-k\}$ we add two oriented edges to the graph:
1) $e^+ = (\hat{S}[i, i+k-1], \hat{S}[i+1, i+k])$, where:
$Color(e^+) = Blue$, $Pos(e^+) = l_i$, $Spell(e^+) = \hat{S}[i, i+k]$
2) $e^- = (Rev(\hat{S}[i, i+k-1]), Rev(\hat{S}[i+1, i+k]))$, where:
$Color(e^-) = Red$, $Pos(e^-) = l_{i+k-1}$, $Spell(e^-) = Rev(\hat{S}[i, i+k])$

A *valid* path in G is a sequence of edges $P = (e_1, e_2, \dots, e_n)$ iff $Pos(e_{i+1}) = Next(Pos(e_i))$ and $Color(e_{i+1}) = Color(e_i)$. Let's denote by $Start(P)$ first vertex of the path P and by $End(P)$ the last vertex of P.

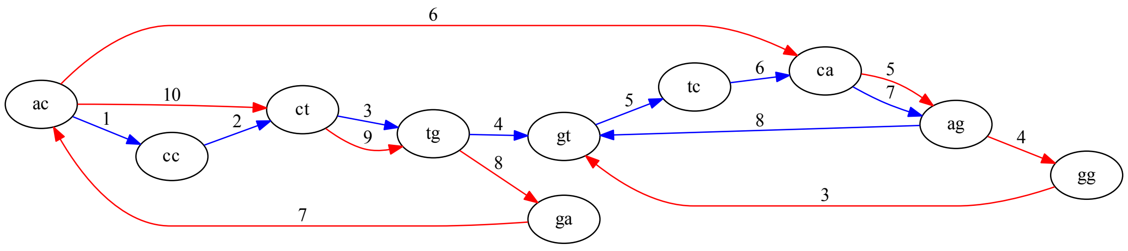


Figure 2: Colored de Bruijn graph for string $S = "acctgtcagt"$

A pair of valid paths $B = \{b_1, b_2\}$ is called a *bulge*, iff following holds:
1) $Start(b_1) = Start(b_2) \wedge End(b_1) = End(b_2)$
2) b_1 and b_2 have no common vertices except $Start(b_1)$ and $End(b_1)$
3) There are no edges $e_1 \in b_1, e_2 \in b_2$ such that $Spell(e_1) = Spell(e_2)$
4) Their sets of positions do not intersect

A bulge $B = \{b_1, b_2\}$ is called *bad* iff $|b_1| < \delta \wedge |b_2| < \delta$ where δ is a parameter.

A vertex v is called *bifurcation* iff there are at least two outgoing (ingoing) edges e_1, e_2 incident v such that $Spell(e_1) \neq Spell(e_2)$. A set of paths $P_{nb} = \{P_1, P_2, \dots, P_n\}$ is said to form a *non-branching path* iff $|P_1| = |P_2| = \dots |P_n|$ and $Spell(e_{i,k}) = Spell(e_{j,k})$, where $e_{i,j}$ denote j-th edge in the i-th path, i.e. all paths spell the same substring.

Let's illustrate above definitions on a simple example. Colored de Bruijn graph built from string $\hat{S} = "acctgtcagt"$ is depicted on figure 2. Here $Rev(\hat{S}) = "actgacaggt"$. Edges' labels denote indices of elements of list L. Vertices $"ac", "ct", "tg"$ are bifurcations, while $"cc", "tc", "ga"$ are not. Two paths $("ac", "ct")$ and $(("ac", "cc"), ("cc", "ct"))$ form a bulge. Two multiedges $("ct", "tg")$ form a non-branching path.

Pseudocode of our algorithm for synteny blocks finding is presented below. Procedure of building a de Bruijn graph is described above. Procedure *ReplaceBranch* works as follows. It takes two paths (edges) as input and then finds two bifurcations with highest degree: one that lies on path that starts by edge e_1 and one that lies on path e_2. A path that has lowest maximum degree of a bifurcation is replaced with the other.

Unfortunately, this description lacks many details (due to lack of time).

$SyntenyFinder(S = \{S_1, S_2, \ldots, S_n\}, k, \delta)$

```
1: $\hat{S} = S_1 \uplus S_2 \uplus \ldots \uplus S_n$
2: $L = ((\hat{s}_1, 1), (\hat{s}_2, 2), \ldots, (\hat{s}_m, m))$
3: $G = BuildDeBruijn(L, k)$
4: $run = True$
5: while $run == True$ do
6:     $run = False$
7:     for $v \in V(G)$ do
8:         for $e_1, e_2 \in Out(v)$ do
9:             if $FormBadBulge(e_1, e_2, \delta)$ then
10:                $run = True$
11:                $ReplaceBranch(e_1, e_2)$
12: $OutputNonBranchingPaths(G)$
```

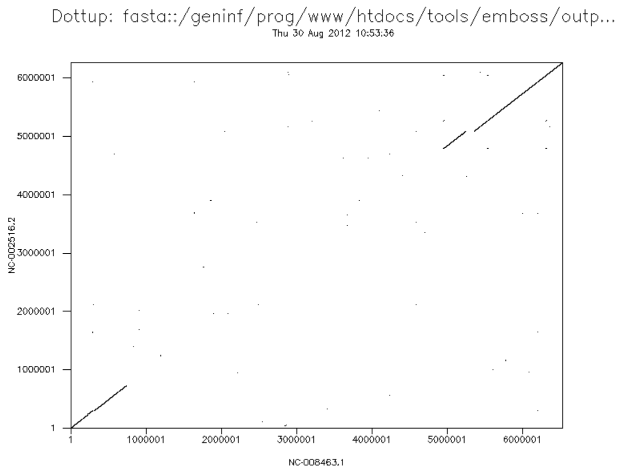
3. Experimental results

We have implemented our algorithm in C++. To evaluate performance of our program, we have performed several tests. First test set consists of two bacteria from *Pseudomonas aeruginosa* group: *Pseudomonas aeruginosa PAO1* and *Pseudomonas aeruginosa UCBPP-PA14*. We built dotplots (see figure 3) to explore conserved regions in these strains. It shows that there are two absolutely conserved regions at the start and the end of the genomes and a big inversion in the middle. We ran our program on the genomes with $k = 5000$ and $\delta = 25000$. As a result, each genome is represented as a permutations of signed integers. Positive number denote occurrence of a synteny block on the positive strand, negative number denote occurrence on the negative strand. Here are resulting permutations (one line corresponds to a genome):
+9 -0 -1 -2 -3 -4 -5 -6 +7 +10 +8
+9 -7 +6 +5 +4 +3 +2 +1 +0 +10 +8

One can see that found synteny blocks are consistent with the dotplots.

Another dataset include three strains of *Mycobacterium tuberculosis H37Rv*:
Laboratory reference strain *H37Rv*
CCDC5180
CCDC5079

One of this strains has multiple drugs resistance. We used $K = 1000$ and $\delta = 5000$ to find synteny blocks. We have found 19 synteny blocks with multiplicity 3. These blocks cover 96% of the genome.



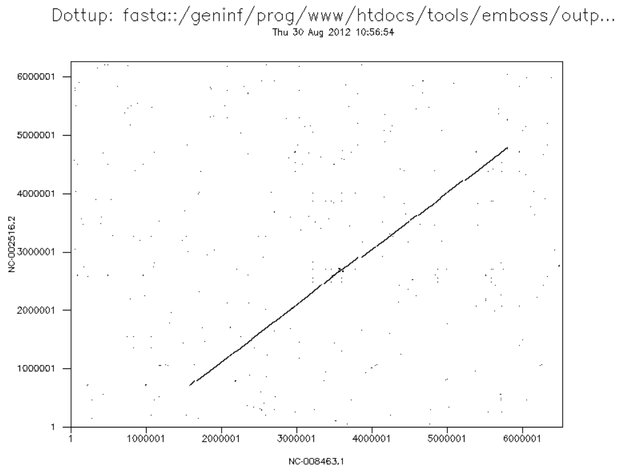


Figure 3: Dot plots of bacteria from *Pseudomonas aeruginosa* group. Left dot plot corresponds to PA01 against PA14, right dot plot corresponds to reverse-complementary of PA01 against PA14

4. Conclusion

In this work we present an algorithm, that is able to find synteny blocks from unannotated sequences. It can be applied to genomes of species that are evolutionary close to each other. With some modifications our method can be extended for finding synteny blocks in more distant genomes.

References

[1] Genome 10K Community of scientists. Genome 10K: A proposal to obtain whole-genome sequence for 10000 vertebrate species. Journal of Heredity, 100(6): 659-674.

[2] Ross C. Hardison. Comparative genomics. PLoS Biology 1(2): e58.

[3] Fabien Aujoulat, Frederic Roger, Alice Bourdier, Anne Lotthe Brigitte Lamy, Helene Marchandin, Estelle Jumas-Bilak. From environment to man: genome evolution and adaptation of human opportunistic bacterial pathogens. Genes 2012, 3(2), 191-232.

[4] Pavel Pevzner, Glenn Tesler. Genome rearrangements in mammalian evolution: lessons from human and mouse genomes. Genome Research. 2003 Jan;13(1):37-45.

[5] Manolis Kellis, Bruce W. Birren, Eric S. Lander. Proof and evolutionary analysis of ancient genome duplication in the yeast Saccharomyces cerevisiae. Nature 2004 Apr 8;428 (6983): 617-24.

[6] Son K. Pham, Pavel A. Pevzner. DRIMM-Synteny: decomposing genomes into evolutionary conserved segments. Bioinformatics (2010) 26 (20): 2509-2516.

[7] Pavel A. Pevzner, Haixu Tang, Michael S. Waterman. An Eulerian path approach to DNA fragment assembly. Proc. Natl. Acad. Sci. USA. 2001 Aug 14; 98(17): 9748-53.

[8] Zamin Iqbal, Mario Caccamo, Isaac Turner, Paul Flicek, McVean. De novo assembly and genotyping of variants using colored de Bruijn graphs. Nat Genet. 2012 Jan 8;44(2):226-32. doi: 10.1038/ng.1028.