Earmark graph approach to *de novo* genome assembly

M. Dvorkin, A. Kulikov

July 5, 2011

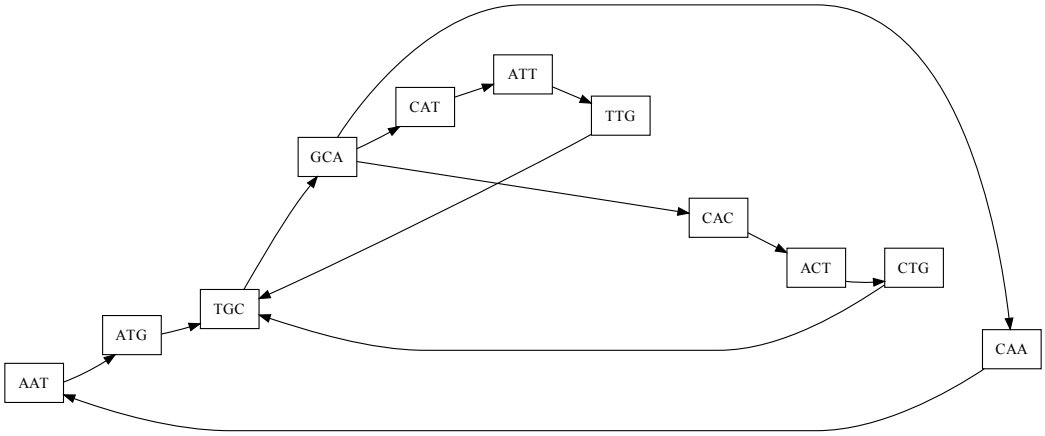
Abstract

A common approach to assembling a genome from short reads is constructing the de Bruijn graph on all k-mers from the given set of reads and finding a traversal of edges in this graph. We propose a new approach that allows to decrease the graph size without losing the essential information from the input data. Instead of using all the k-mers from a read we take only a few of them (and call them earmarked). Besides an obvious advantage of requiring less memory and time for constructing, the resulting earmark graph has several other advantages over the de Bruijn graph. We discuss them in the paper and also present some experimental results.

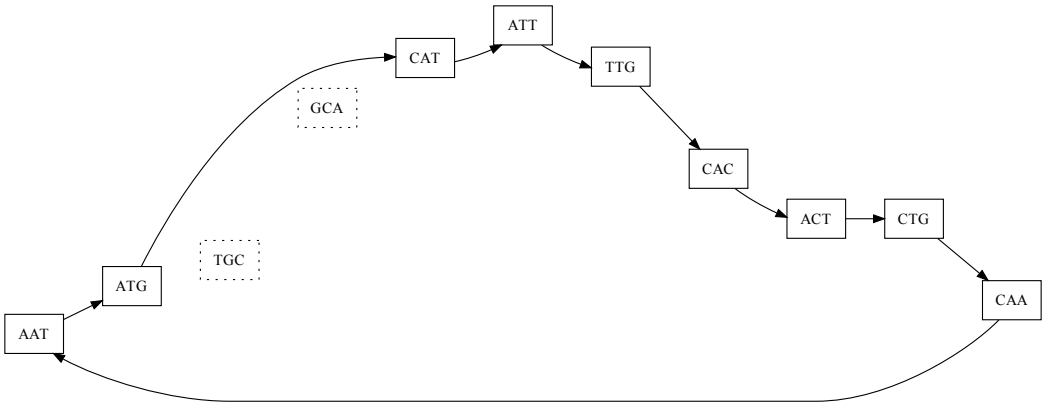
Contents

1 Introduction

A common approach to assembling a genome from short reads is constructing the de Bruijn graph [**?**] on all k-mers from the given set of reads and finding a traversal of edges in this graph. An example of the de Bruijn graph for $k = 3$ and a set of all reads of length 6 of



(a)

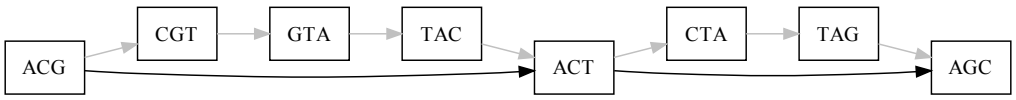


(b)

Figure 1: (a) The de Bruijn graph built on 3-mers of all 6-reads of a toy genome `ATGCATTGCACTGCA`. (b) The earmark graph build on the same set of reads with all 3-mers earmarked except for `TGC` and `GCA`.

a (circular) genome $g = \texttt{ATGCATTGCACTGCA}$ is given in Fig. 1a (edge multiplicities are not shown). The genome spells a Chinese Postman cycle in the de Bruijn graph.

The size of the de Bruijn graph for most genomes is huge making it difficult to process. We propose a new approach that allows to decrease the graph size without losing the essential information from the input data. Instead of using all the k-mers from a read we take only some fraction of them (and call them *earmarked*). Namely, instead of representing each read as a sequence of edges between its consecutive k-mers (in which case a read of length r defines $r - k - 1$ edges) we represent it as just one (or a few, in general) edge between some of its k-mers. For example, for reads $\texttt{ACGTACT}$ and $\texttt{TACTAGC}$ and $k = 3$ instead of all gray edges in the figure below we will have only two black edges joining earmarked k-mers $\texttt{ACG}$, $\texttt{ACT}$, and $\texttt{AGC}$.



We call the resulting graph built on earmarked k-mers *an earmark graph*. It is a special case of the A-Bruijn graph introduced by Pevzner, Tang, and Tesler [**?**]. An example of the earmark graph for the same genome $g = \texttt{ATGCATTGCACTGCA}$ is shown in Fig. 1b. Here, all 3-mers are earmarked except for $\texttt{TGC}$ and $\texttt{GCA}$. Besides an obvious advantage of requiring less memory and time for constructing, the resulting earmark graph has several other advantages over the de Bruijn graph. We discuss them in Section 4.

Our approach is inspired by the work by Roberts et al. [**?**] on sequence comparison. The problem they studied is a pairwise comparison of a set of strings. One of the approaches to this problem is the seed-and-extend approach. One first stores the set of all possible k-mers from a given set of strings. Then, only pairs of strings that both have the same k-mer as a substring are compared. This allows to avoid comparing all pairs of input strings. However the database of all k-mers may be really huge. To reduce the storage requirements Roberts et al. propose to select not all the k-mers, but only those that are minimal in an input string with respect to a certain order. Such k-mers are called *minimizers*. Our earmarked k-mers are close in spirit to minimizers.

2 Earmark graph

In a typical genome assembly setting, one is given a set $\mathcal{R} \subseteq \{\texttt{A}, \texttt{C}, \texttt{G}, \texttt{T}\}^r$ of substrings of length r of an unknown genome $S \in \{\texttt{A}, \texttt{C}, \texttt{G}, \texttt{T}\}^*$. The elements of $\mathcal{R}$ are usually called reads or r-reads. For a read R and indices $1 \le i \le j \le r$, let $R[i, j]$ be a substring of R starting at position i and ending at position j. By a k-mer we mean any string from $\{\texttt{A}, \texttt{C}, \texttt{G}, \texttt{T}\}^k$. Throughout the rest of the paper r and $k < r$ refer, respectively, to read and mer size.

A *de Bruijn graph* $\mathrm{DG}_k(\mathcal{R})$ is defined as follows:

- the set of all substrings of length k (usually called k-mers) of the given reads is the set of vertices;
- two k-mers A and B are joined by a directed edge if there is a read $R \in \mathcal{R}$ and an index $1 \le i \le r-k-1$ such that $A = R[i, i+k-1]$ and $B = R[i+1, i+k]$ (this, in particular, implies that the suffix of A of length $k-1$ equals the prefix of B of the same length).

Note that each read $R \in \mathcal{R}$ is represented by a path of length $r-k-1$ on its k-mers in $\mathrm{DG}_k(\mathcal{R})$.

An earmark graph can be viewed as the de Bruijn graph with some paths contracted into a single edge. Let $\mathcal{E} \subseteq \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}^k$ be a set of k-mers. Below we refer to it as a *set of earmarks* or as a *set of earmarked k-mers*. An *earmark graph* $\mathrm{EG}_k(\mathcal{R}, \mathcal{E})$ is defined as follows:

- $\mathcal{E}$ is the set of vertices;
- two k-mers $A, B \in \mathcal{E}$ are joined by a directed edge if there is a read $R \in \mathcal{R}$ and indices $1 \le i < j \le r-k-1$ such that $A = R[i, i+k-1]$ and $B = R[j, j+k-1]$ and for any $i < t < j$, $R[t, t+k-1] \notin \mathcal{E}$ (i.e., B is the next earmarked k-mer after A in R); the length of this edge is set to $j-i$.

Hence, in the earmark graph each read is represented as a path (of length at most $r-k-1$) on its earmarked k-mers. When constructing an earmark graph it is reasonable to ensure that each read contains at least two earmarked k-mers (so that the length of the corresponding path is at least one). It is easy to see that in the special case when all the possible k-mers are earmarked the earmark graph coincides with the de Bruijn graph.

Both de Bruijn and earmark graphs are built on a set of reads that can be viewed as a set of reads from an unknown genome. A natural way to build these two graphs on a genome itself is to use the set of all its reads of a particular length. Namely, for a genome $S \in \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}^*$, define $\Gamma_r(S) \subseteq \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}^r$ as a set of all r-mers appearing in S. Then $\mathrm{DG}_k(S)$ and $\mathrm{EG}_k(S, \mathcal{E})$ are just $\mathrm{DG}_k(\Gamma_r(S))$ and $\mathrm{EG}_k(\Gamma_r(S), \mathcal{E})$, respectively. It is easy to see that the genome S spells a Chinese Postman cycle in $\mathrm{DG}_k(S)$.

3 Earmark graph construction

In this section, we give a high-level description of an earmark graph construction procedure. When the graph is constructed we simplify it using methods similar to the ones used in EULER [**?**] and Velvet [**?**].

We first construct an initial set of earmarks and then extend it to reduce the number of tips resulted from bad choice of earmarks.

3.1 Earmarks selection procedure

To construct an initial set of earmarks $\mathcal{E}$, we select from each given read several k-mers that are minimial with respect to some ordering. Namely, let h be a hash-function on the set of all possible k-mers and t be a fixed integer parameter. We then select initial earmarks as follows.

Algorithm 3.1 EarmarksSelection

Input: a set of reads $\mathcal{R} \subseteq \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}^r$ of length r, a hash function $h\colon \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}^k \to \mathbb{N}$ on k-mers, an parameter $t \in \mathbb{N}$
Output: a set of earmarks $\mathcal{E} \subseteq \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}^k$

```
E ← ∅ {set of earmarked k-mers}
H ← ∅ {set of earmarked hash values}
for all R ∈ R do
    find t minimum hash values of all (r − k + 1) k-mers of R and add them to H
end for
if t = 1 then
    for all R ∈ R with only one k-mer with earmarked hash value do
        add to H the second smallest hash value of all the k-mers of R
    end for
end if
for all R ∈ R do
    for all k-mers K of R do
        if h(K) ∈ H then
            E = E ∪ {K}
        end if
    end for
end for
return E
```

We mark hash values but not k-mers themselves to reduce the space required to store what is marked. This way some k-mers may be unwillingly marked due to collisions, but in case of a reasonable hash function the number of such collisions is negligible.

3.2 Tip extension procedure

Assume that we are given a set of reads $\mathcal{R}$ of an unknown genome S. It is easy to see that parts of the genome S that are not covered by any read from $\mathcal{R}$ create vertices of in- or our-degree one in $\mathrm{DG}_k(\mathcal{R})$. Such vertices are called *tips*. At the same time, the earmark graph may contain tips not only because of coverage gaps, but also tips resulting from a bad choice of the set of earmarks.

To give an example, consider two reads `TATGCA` and `GCATCC`. The corresponding path in the de Bruijn graph is shown in Fig. 2a. Assume now that only 3-mers `ATG`, `TGC`, `CAT`, and

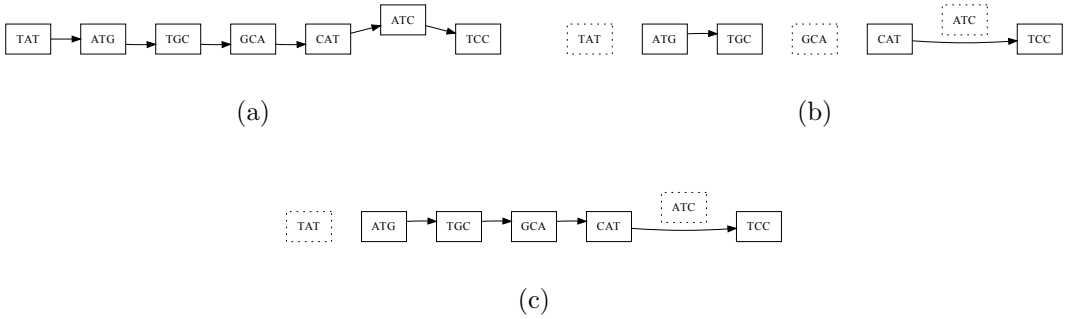


Figure 2: (a) Part of the de Bruijn graph corresponding to reads `TATGCA` and `GCATCC`. (b) The corresponding part of the earmarked graph has a gap if the 3-mer `GCA` is not earmarked. (c) The gap can be avoided by earmarking the 3-mer `GCA`.

`TCC` are earmarked. Then the corresponding part of the earmark graph contains two tips (Fig. 2b). To avoid it we earmark also the 3-mer `GCA` (Fig. 2c).

The tip extension procedure is suggested to handle this issue, see Algorithm 3.2. complete

Algorithm 3.2 TipExtension

Input: a set of reads $\mathcal{R} \subseteq \{\mathtt{A},\mathtt{C},\mathtt{G},\mathtt{T}\}^r$ of length r, a set of earmarks $\mathcal{E} \subseteq \{\mathtt{A},\mathtt{C},\mathtt{G},\mathtt{T}\}^k$
Output: an extended set of earmarks $\mathcal{E} \subseteq \mathcal{E}' \subseteq \{\mathtt{A},\mathtt{C},\mathtt{G},\mathtt{T}\}^k$
  $\mathcal{L} \leftarrow \mathcal{E}$, $\mathcal{R} \leftarrow \mathcal{E}$ {sets of left and right tips, respectively}
  for all $R \in \mathcal{R}$ **do**
    remove all but the first earmarked k-mer of R from $\mathcal{L}$
    remove all but the last earmarked k-mer of R from $\mathcal{R}$
  end for
  return $\mathcal{E}$

1. By processing all the reads, find out for each earmarked k-mer, whether we have seen any earmarked k-mer to the left of it and to the right from it. Those earmarked k-mers that do not have either left or right "neighbours" are called left and right tips, respectively.

2. By processing all the reads, find the collection of all the non-earmarked k-mers that are present (at least once) in a read to the left of a left tip, or to the right of the right tip. Call them possible tip extensions.

3. By processing all the reads, for each possible tip extension, record whether there was any earmarked k-mer to the left of it and to the right of it.

Now that this information is collected, each tip is being extended with a possible tip extension using the following rules (in the order preference).

1. Check if any of its possible tip extensions was already selected as an earmark (as a tip extension for some previously processed tip, using rules 2 and 3 below). If so, continue to the next tip.
2. Check if any of its possible tip extensions has both left and right neighbours. We select this extension as an earmark and thus eliminate at least one undesirable tip.
3. Select as an earmark the possible tip extension that is most distant (in base pairs) from the tip being proccesed. Doing this, we do not eliminate a tip but we extend it as far as possible. This helps us not to lose information near the actual gaps (or the ends of actual scaffolds).

If no new earmarked k-mers were introduced after this procedure, then stop, otherwise repeat the entire procedure.

4 Advantages of the earmark graph over the de Bruijn graph

Smaller size. The earmark graph requires less time and memory for construction. Note also that one can control the size of the earmark graph by varying the size of the set of earmarks (e.g., by varying the value of the parameter t of Algorithm 3.1).

Some of short repeats are already resolved. To give an example, consider two reads `TTGCAC` and `ATGCAT` sharing a 4-mer `TGCA`. They are represented as two paths, shown in Fig. 3a, in the de Bruijn graph built on 3-mers (this is a part of the de Bruijn graph from Fig. 1). This is a typical repeat. The edge `TGC` $\rightarrow$ `GCA` has two incoming edges and two outgoing edges. While spelling a genome through this part of the graph it is not clear which incoming edge corresponds to which outgoing edge. However in the earmarked graph this repeat may be already resolved if the 3-mers `TGC` and `GCA` are not earmarked, see Fig. 3b.

Less errors. Erroneous k-mers can be excluded from the graph already on the construction stage.

TBW

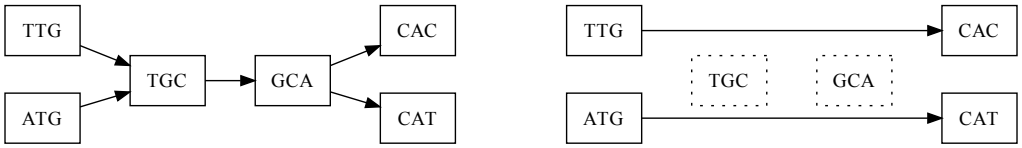
what else?

5 Practical results

fill in Table 1

6 Discussion and further ideas

to be written



(a) (b)

Figure 3: A small repeat in the de Bruijn graph (a) may be resolved in the earmarked graph (b) in case k-mers from the repeated region are not earmarked.

		de Bruijn	earmarked
first 10% of E.coli	# vertices	809730	62420
	# edges	810354	62900
+ compression	# vertices	5020	1376
	# edges	5644	1856
+ tips clipping	# vertices	1310	818
	# edges	1934	1298
+ bulge removal	# vertices	964	472
	# edges	1410	750
+ erroneous connection removal	# vertices	398	212
	# edges	570	314
+ tips clipping	# vertices	360	172
	# edges	532	274
+ bulge removal	# vertices	242	106
	# edges	354	166

Table 1: Practical results