# Enabling HPRC reference in GED-MAP through bidirected sequence graph to EDS graph transformation

Thomas Büchler
Institute of Theoretical Computer Science, Ulm University, Germany

August 11, 2023

## Abstract

The reference genome plays a key role in genome analyses. In the process of read mapping DNA reads are aligned against the given reference. While the most basic form of reference comprises a single linear sequence, this linear representation cannot include common genomic variations. In response to this limitation, over the past decades a linear reference was combined with a list of variation into a simple structured graph. Several methods were developed to align reads against such graphs [1, 3, 4, 6]. Recently a new (more complex) reference graph was assembled and proposed to be used as the new reference standard [5]. This report describes how this graph can be transformed so it can be used in the mapping software GED-MAP [1].

**Availability:** github.com/thomas-buechler-ulm/gedmap
**Contact:** thomas.buechler@uni-ulm.de

## 1 Introduction

The mapping software GED-MAP is capable of aligning reads to a reference graph with a relatively low consumption of space and memory [1]. A central innovation of this tool is integrating small genomic variation in the sequences of the node labels rather than adding separate nodes for every variant. In its first published version the demanded input for the graph construction was a reference sequence in FASTA format and a enumeration of variations in Variant Call Format (VCF).

This year, a significant development has occurred with the introduction of new reference graphs by the Human Pangenome Reference Consortium [5]. These graphs were build from haplotype assemblies from 47 genetically diverse individuals and are available in Graphical Fragment Assembly (GFA) format. Using this new graph as reference for read mapping is highly interesting. Nevertheless, an inherent obstacle is encountered; the reference graph, as it stands, lacks compatibility with the GED-MAP software. This incompatibility is based on the divergent nature of the underlying graph model. As a response to this challenge, this technical report elaborates a transformation process that enable the usage of the HPRC graph in GED-MAP.

## 2 Basic Definitions

This section provides the definitions used in this report. The first is the sequence graph, which is a simple way to represent a set of sequences in a graph.

**Definition 1 (Sequence Graph)** *A directed node labeled graph, with a set of nodes $V$, a set of edges $E \subseteq V^2$ and a labeling function $l : V \mapsto \{A, C, G, T, N\}^*$, that maps each node to a sequence called label.*

**Definition 2 (Walk)** *Given a labeled graph $G = (V, E)$. We call a sequence $W = (x_1, \ldots, x_n)$ of nodes a walk in $G$, if: $(x_i, x_{i+1}) \in E$ for $1 \leq i < n$. We call $n$ the* length *of the walk $W$. The label of a walk $l(W) = l(x_1) \ldots l(x_n)$ is the concatenation of the labels of the nodes in the sequence. The* size *of $W$ equals the size of its label, $||W|| = |l(W)|$.*

A sequence graph represents all sequences that are labels of walks in the graph. (Respectively, all sequences that belong to a walk from a source to a sink in the graph.) A more general concept is the bidirected sequence graph, in which the nodes can be traversed from different direction to describe the two strands of the DNA. Note: The HPRC graph is a bidirected sequence graph.

**Definition 3 (Bidirected Sequence Graph)** *A modified sequence graph, in which nodes have two orientation $\{+,-\}$. Every edge also indicates in which orientation the nodes are approached. Therefore, $E \subseteq (V \times \{+,-\})^2$. If a node is approached in '+'-orientation its label will be read in forward direction. If a node is approached in '−'-orientation the reverse complement of the label will be read.*

Bidirected sequence graphs allow an smoother representation of inversion. The graph used in GED-MAP is an EDS graph. To define them we first shortly define the elastic degenerate string (EDS).

**Definition 4 (EDS)** *Originally, a sequence of sets of strings [2]. A string matches an ED string, if it can be partitioned to a sequence (of the same length as the EDS) of strings in a way that each string appears in the corresponding set. In [1] it is defined, analogously to regular expressions that only allows 'grouping' and 'or'. E.g. $AT(G|C)TTA$ is an EDS. (In set notation: $\{AT\}\{G,C\}\{TTA\}$.)*

**Definition 5 (EDS Graph)** *A sequence graph, in which the labels are ED strings.*

The EDS graph can represent small variation (often called bubbles) in the ED strings, which means it may need fewer nodes than a sequence graph while preserving the overall structure. Roughly spoken, a bubble is a enclosed subgraph that represents small variants (i.e. single nucleotide polymorphisms, etc.). We formally define the term 'bubble' in the following definitions.

**Definition 6 (Walk from $u$ to $v$)** *We call Walk $W = (x_1, \ldots, x_n)$ in the Graph $G = (V, E)$ a walk from $u$ to $v$ if, $(u, x_1) \in E$ and $(x_n, v) \in E$. If $(u, v) \in E$, then the empty sequence $\epsilon$ is a walk from $u$ to $v$.*

**Definition 7 (Bubble between $u$ and $v$)** *We say, there is a bubble of width $t$ between $u$ and $v$ if:*

- *Any walk from $u$ to a sink includes $v$.*
- *Any walk from a source to $v$ includes $u$.*
- *The largest walk from $u$ to $v$ has a maximum size of $t$.*

*We denote a bubble $B = \{W_1, \ldots W_k\}$ by the set of walks between $u$ and $v$. The parameter $t$ ensures the locality of the variations the bubble represents.*

**Definition 8 (Classification of bubbles)**

- Concatenation*: $B = \{\epsilon\}$, in this case $u$ has only one successor $v$ and $v$ has only one predecessor $u$.*
- SNP*: For all $W \in B : ||W|| = 1$.*
- Indel *of node $x$: $B = \{\epsilon, (x)\}$.*

# 3 Transforming the HPRC graph

To use the HPRC graph as reference in GED-MAP, one has to transform the bidirected graph to an EDS graph. This can be done by firstly transforming the graph to a sequence graph and then repeatedly detecting bubbles in the graph and replace them by nodes with EDS labels. The following process describes this procedure in more detail.

## 3.1 Transforming a bidirected graph to a directed graph

To transform the bidirected sequence graph to an 'equivalent' sequence graph we have to get rid of the orientation of the edges while maintaining the represented sequences. For this reason, we will need to add new nodes in some cases. For each node $v$ of a given bidirected sequence graph, one the following four cases applies:

(A) there is no edge connected to $v$,

(B) all edges connected to $v$ approach $v$ in '+'-orientation,

(C) all edges connected to $v$ approach $v$ in '−'-orientation or

(D) some edges approach $v$ in '+'-orientation and some edges approach $v$ in '−'-orientation.

For nodes of case (A) or (B) we do not change anything. If case (C) applies to a node $v$, we replace its label be the reverse complement and change the orientation of all edges connected to $v$ to '+'. If case (D) applies to a node $v$, we add a new node $v^*$ to $V$, with $l(v^*)$ equals the reverse complement of $l(v)$. All edges that connect to the '−'-orientation of $v$ will be replaced by edges that connect to the '+'-orientation of $v^*$. After these changes all orientations on edges will be '+', and therefore can be omitted.

## 3.2 Representing bubbles as ED strings

To make use of the regex-like notation of the EDS graph, we will have to identify bubbles in the sequence graph and replace them with ED strings. This section explains this process in more detail.

### 3.2.1 Finding bubbles in a sequence graph

The method of detecting bubbles is split into several algorithms. The first (Algorithm 1) checks if there is a bubble between two nodes $u$ and $v$. This is done be a depth first search starting at $u$, that keeps track of the walk size $s$. If we see $v$, we do not further expand this path. If an observed walk size is greater than the threshold $t$, or a sink is detected there is no bubble between $u$ and $v$. Hence, the algorithm outputs *false*.

**Algorithm 1** Bubble between $u$ and $v$

**Input:** $u \neq v \in V$, Maximum size $t$.
**Output:** true, if there is a bubble of size $\leq t$ between $u$ and $v$.

```
1:  function BubbleBetween(u, v, t)
2:      S ← empty stack
3:      push (u, 0) to S                    ▷ S contains the end and size of all observed walks
4:      while S not empty do
5:          (e, s) ← Q.pop()
6:          if e ≠ v then
7:              if e is a sink ∨ s > t then
8:                  return false
9:              end if
10:             for all (e, x) ∈ E do
11:                 push (x, s + |l(x)|) to Q        ▷ extend the current path with x
12:             end for
13:         end if
14:     end while
15:     return true
16: end function
```

The second algorithm (Algorithm 2) determines, if there is a bubble starting at $u$. Therefore, the algorithm, starts a random walk at $u$. For each node $v$ of the walk, the first algorithm is called to check, if there is a bubble between $u$ and $v$. If so, the algorithm terminates. If the size of the random walk is $> t$ or the walk reaches a sink, there is no bubble starting at $u$. Applying this algorithm to all nodes of the graph is able to detect all bubbles.

**Algorithm 2** Bubble start $u$

**Input:** $u \in V$, Maximum size $t$.
**Output:** true, if there is a bubble of size $\leq t$ starting at $u$.

```
1:  function BubbleStart(u, t)
2:      e ← u                               ▷ Current end of the walk
3:      s ← 0                               ▷ Current size of the walk
4:      while true do
5:          if e is sink ∨s > t then
6:              return false
7:          end if
8:          e ← arbitrary successor of e
9:          s ← s + |l(e)|
10:         if BubbleBetween(u, e, t) then
11:             return true
12:         end if
13:     end while
14: end function
```

### 3.2.2 Transforming a sequence qraph to an EDS graph

To transform the sequence graph to an EDS graph, we check for each node, if there is a bubble starting at this node. If this is the case, we replace the nodes of the bubble by a new node, representing the bubble in the EDS.

Algorithm 3 outlines this procedure. The EDS string that describes a bubble is defined in following definition. An example how a bubble is transformed is shown in Figure 1.

**Definition 9 (EDS label of a bubble)** *Let $B = \{W_1, \ldots, W_k\}$ be a bubble between $u$ and $v$. The EDS label of this bubble is $l(B) = u\big(\, l(W_1) \,|\, \ldots \,|\, l(W_k) \,\big)v$. (Special case: if the bubble classifies as concatenation, then $l(B) = l(u)l(v)$.)*

**Algorithm 3** Transforms the sequence graph to an EDS graph

**Input:** Sequence graph $G = (V, E)$, threshold $t$ for bubble sizes.

```
for all u ∈ V do
    if BubbleStart(u, t) then
        Determine the bubble B between u and v.
        Add a new node b with EDS label l(B) to V.
        The predecessors of b are the predecessors of u and the successors of b are the successors of v.
        remove v, u, and all nodes in the bubble B from V.
        (Repeat this iteration of the for loop for u.)
    end if
end for
```


Figure 1: Replace a bubble by a single node with an EDS label

# 4 Experimental evaluation

The original CHM13 reference graph from the HPRC [5] has over 85 million nodes. Of these, 69 million nodes are only visited in '+'-orientation, 9 million nodes are only visited in '−'-orientation and less than 7 million nodes are visited in both orientations. A few thousand are not connected to other nodes. Furthermore, the graph contains fewer than 180 thousand sources and 280 thousand sinks.

In this graph the described methods first add about 7 million nodes for reverse complements and then detect 38 million bubbles (with width $t \leq 50$). The bubbles can be divided in 19.4 million SNP, 14.7 million concatenations, 2.6 million Indel and 1.7 million other bubbles. The transformed graph has about 6.4 million nodes. On a single thread on an AMD EPYC 7742 processor with 256 GB RAM this transformation took less than 20 minutes. One might wonder, why so many 'simple' concatenations occur. This is because the addition of a new node for a reverse complement may cause several concatenations. An example is depicted in Figure 2.

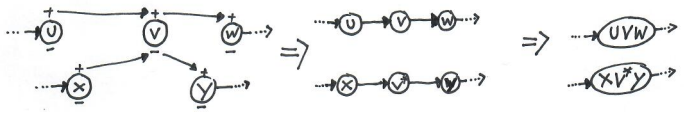


Figure 2: First: add a node for the reverse complement of $v$ and adjust edges. Then detect bubbles and concatenate the paths.

Performing the mapping experiments of [1] with the HPRC CHM13 graph as reference requires about 25% more resources (time and memory) than using the graph generated from the linear reference with the list of variations.

# 5 Known Issues

The proposed method possesses certain limitations that do not hinder its utilization, but provide opportunities for further optimization.

**Nesting** According to the definition, there can be bubbles inside other bubbles and this could lead to nested ED strings. Since GED-MAP does not support nested ED strings, we omit such bubbles. (To be more precise, if the program ignores all detected bubbles, that contain a node with ED string as label.)

**Ambiguousness of the output** The order in which the nodes of $V$ are processed in Algorithm 3 affects the output of the transformation. At nested bubbles for example, the representation will be different if the inner bubble will be detected a) before or b) after the outer bubble, see Figure 3. Either way the output graph will represent the same sequences and has fewer nodes than the input graph.

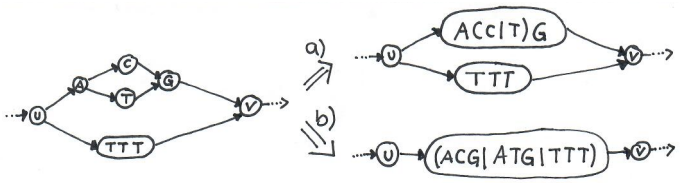


Figure 3: Nested bubbles. Depending if $u$ or $A$ is processed first, the resulting graph is more like a) or b).

**Unobserved paths** Using the graph without the path information can generate walks that were not observed before. To handle this, one would have to check the observed paths instead of performing a breadth first search in Algorithm 1 and take a existing path instead of a random walk in Algorithm 2. We did not use path information, to have a more generic approach and because GED-MAP does not use path information as well. Figure 4 depicts an example.

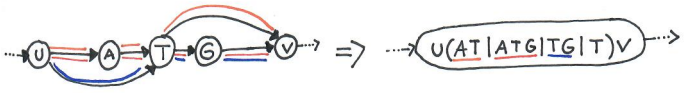


Figure 4: Unobserved paths added. Colors represent observed paths. The path '$u\,T\,v$' did not exist before.

**Small nodes remaining** The basic idea of this transformation is to omit nodes with small labels and represent them the ED strings of longer labels. However, not all small nodes do lie in enclosed bubbles. It could sometime be worth to separate paths as shown in Figure 5. But this report does not give an answer on how this can be done.

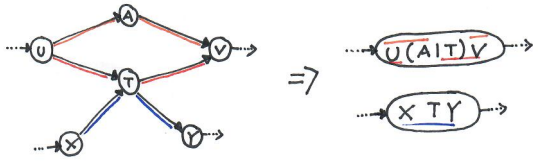


Figure 5: Small nodes remaining: Both graphs represent the same paths. The graph on the right would be more appropriate to use in GED-MAP. Unfortunately, the presented method cannot detect this transformation.

**Program output** The alignment will be given Sequence Alignment Map (SAM) format. This format provides the sequence and starting position of the alignment. Traditionally this was a position on the linear reference. The HPRC graph defines the position on the linear reference for many nodes. For these nodes GED-MAP will output these 'traditional' positions. If the alignment starts at a node, that has no defined reference position, GED-MAP outputs the node number and the offset from the on this node, where the alignment starts. A perhaps better solution would be using the Graph Alignment Format (GAF) in the output.

# 6 Conclusion

In summary, the transformation of the bidirected sequence graph into an EDS graph, as described in this technical report, has been successfully integrated into GED-MAP (available on github.com/thomas-buechler-ulm/gedmap). This transformation led to a remarkable reduction of over 90% in the number of nodes of the HPRC reference graph. However, certain subgraphs still contain multiple small nodes closely connected, as e.g. depicted in Figure 5. These subgraphs, especially when aligned with GED-MAP's focus on graphs with longer node labels, may slightly impact the program's performance. The mapping process with the transformed

HPRC graph currently requires approximately 25% more time and space than the one utilizing the graph generated from the linear reference and variation list. Further optimization might be possible by incorporating path information in definition of bubble, which could potentially further decrease the node count and optimize mapping speed. We acknowledge the need for such refinements and are working towards even more efficient versions of GED-MAP.

# References


[1] Thomas Büchler, Jannik Olbrich, and Enno Ohlebusch. Efficient short read mapping to a pangenome that is represented by a graph of ed strings. *Bioinformatics*, 39(5), 2023.

[2] Costas Iliopoulos et al. Efficient pattern matching in elastic-degenerate texts. In *Language and Automata Theory and Applications*, volume 10168 of *LNCS*. Springer, 2017.

[3] Daehwan Kim et al. Graph-based genome alignment and genotyping with HISAT2 and HISAT-genotype. *Nature Biotechnology*, 37(8), 2019.

[4] Brice Letcher et al. Gramtools enables multiscale variation analysis with genome graphs. *Genome biology*, 22, 2021.

[5] Wen-Wei Liao et al. A draft human pangenome reference. *Nature*, 617(7960), 2023.

[6] Jouni Sirén et al. Pangenomics enables genotyping of known structural variants in 5202 diverse genomes. *Science*, 374, 2021.