Supplemental materials for the introduction of StrainCall

Feng Zeng, Zicheng Wang, Ting Chen

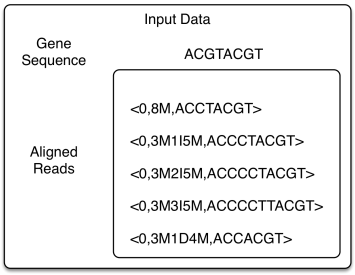
1. Partial order graph: definition and construction

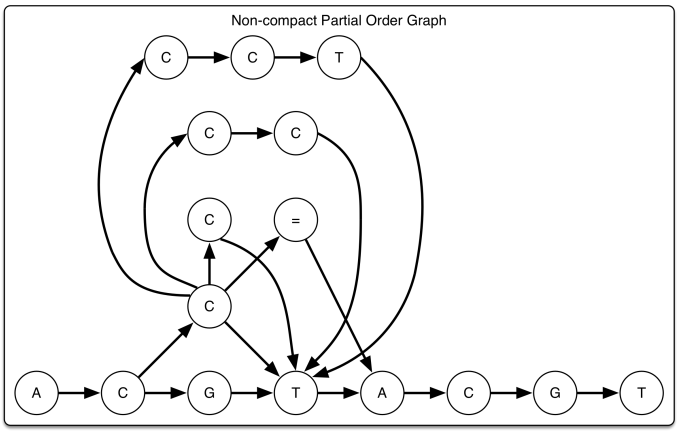
Partial order graph (POG), denoted as $G = \langle V, E \rangle$, is a graph data structure to represent a multiple sequence alignment (MSA). Let $\mathcal{A} = \{A, C, G, T, -, =\}$ be the set of letters that occurring on a MSA, of which the letter '$-$' is for the insertion and the letter '$=$' is for the deletion. Thereafter, a vertex $u \in V$ encodes a non-redundant letter $\alpha_u \in \mathcal{A}$ on an alignment position of a MSA. Two nearby vertexes are linked if they were co-occurring on a sequence in a MSA.

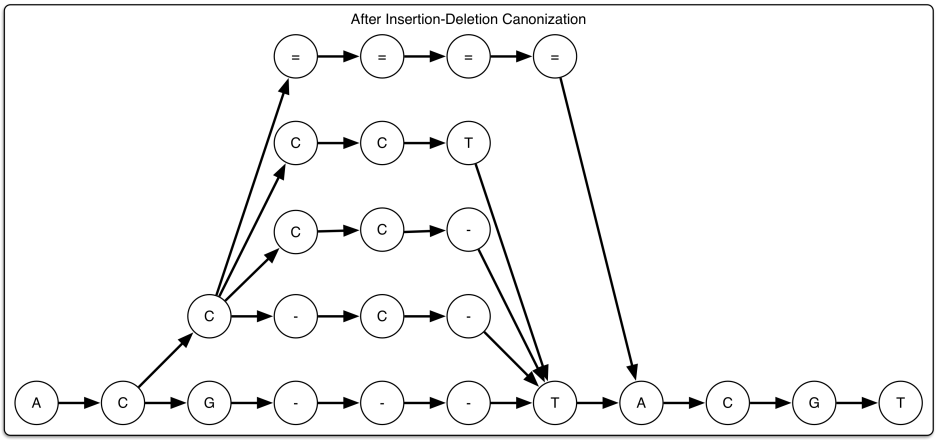
A read $r \in R$ within a read mapping is represented as a 3-tuple, $r = \langle pos_r, cigar_r, seq_r \rangle$. The first element pos_r is the starting position of the read r on a reference sequence. The second element $cigar_r$ is a string to encode the alignment form. The third element seq_r is the sequence of the read r.

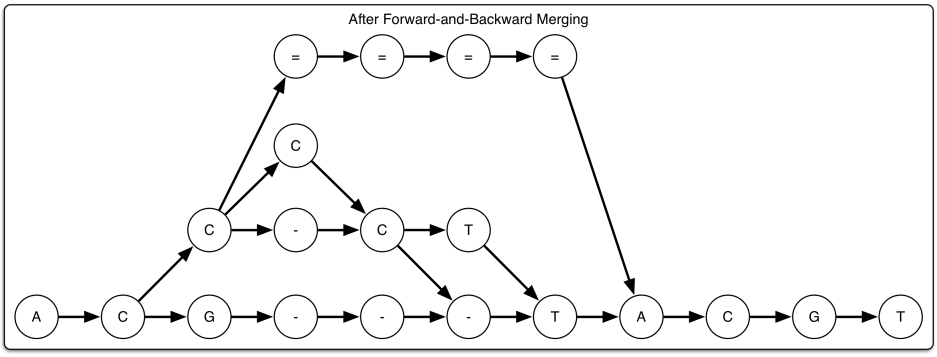
Given a reference sequence T and a read mapping R, a POG is constructed as follows. First, an initial POG G is established, which only includes a backbone for the reference sequence T. In addition, two auxiliary vertexes are introduced to the backbone. One is labeled as '^' and represents the beginning. The other is labeled as '$' and represents the ending. Next, the aligned reads are added to G one by one. Scanning through the current read r from left to right, a vertex is added to G if a mismatch occurs at the current alignment position. For a l-bp insertion, a path of length l is added after the current position, and vertexes on the path are labeled with '$-$'. It is the same for a deletion, but the vertexes are labeled with '$=$'. This operation on both insertions and deletions is simple, and will result in a non-compact graph structure. A forward-backward merging algorithm will be presented later to merge the redundant insertion paths and deletion paths, so as to obtain a compact graph. Third, a multiple sequence alignment method is used to adjust the random insertions. Finally, a compact graph is obtained after using a forward-backward merging algorithm. An example of the POG construction is given at Figure S1.

Figure S1. An example of the POG construction









Pseudocode 1: PartialOrderGraphConstruct(T,R)
$G \leftarrow BuildBackbone(T)$
u is the previous visiting vertex
v is the current visiting vertex
for r **in** R
scan r from left to right
if current alignment position is mismatch, **then** add a sibling w to v, and link u and w
if an insertion occurs before the current position, **then** add a insert path after u
if a deletion occurs before the current position, **then** add a delete path after u
update u and v
reset u and v
canonize the alignment of the insertions
canonize the alignment of the deletions
forward merge the duplicate outgoing vertexs, and backward merge the duplicate incoming vertexes

The level of a vertex on a POG is defined as the distance away from the beginning vertex. A graph walker visits the vertexes following the level order, and to merge the duplicate vertexes. In the forward direction, the graph walker visits the vertexes from the beginning vertex to the ending vertex. At a vertex $u \in V$, the walker scans through the outgoing vertexes $v \in out(u)$. Two vertexes $v_i \in out(u)$ and $v_j \in out(u)$ are merged if they have the same label. After merging, both v_i and v_j are removed from the POG, and a new vertex v is added to take the place. It is the same for the backward merging, but the walker visits the vertexes from the ending vertex to the beginning vertex. A compact graph will be obtained after the forward-backward merging (Figure S1d). Over a compact POG, none of the outgoing vertexes of $v \in V$ are duplicated, so as the incoming vertexes of v.

2. Streaming Dirichlet process mixture clustering

A strain s is a path $(v_1, \cdots, v_m)$, $v_i \in V$ from the beginning vertex to the ending vertex on a POG. Thus, a traversal of all vertexes over a POG will generate a set of all candidate strains. The volume of the set of all candidate strains exponentially grows up along with the number of the branching vertexes on a POG. As sequencing errors usually generate numerous branches over a POG, it is an extremely computational challenge to find a small set of strains in the exponentially explosive set. Another difficulty is the unknown number of the strains in a sequenced population. To address the aforementioned two issues, we design a streaming Dirichlet process mixture clustering method to infer the most possible strains from a POG. The intuitive interpretation is that a graph walker visits the vertexes one level by one level. At each level, the candidate strains are constructed from the previous inferred strains and graph structure. And then, a Dirichlet process mixture clustering method is applied to infer the most possible strains and the abundance levels. This procedure continues until the ending vertex, where a set of the most possible strains is reported as the reconstructed sequences.

First, a generative model is addressed as follows. In a population, a read r is generated by first randomly picking a strain s out of the population and then sequencing. Let $a = \left(a_{s_1}, \cdots, a_{s_k} \cdots, a_{s_K}\right)$ count the number of the reads assigned to the strain s_k. It gives an estimation of the strain abundance levels. A strain s is sampled from the population according to the abundance level a using a Dirichlet process. Specifically, the Dirichlet process uses a Polya urn scheme to sample a parameter setting π_S of a multinomial distribution over the strains according to a. Next, a strain is sampled from the multinomial distribution with the parameter setting π_S. Subsequently, a read r is sequenced from the sampled strain s according to the generation probability $p(r|s)$. The above procedure is summarized as the following equations,

$$\pi_S \sim Dirichlet(a)$$
$$s \sim Multinomial(\pi_S)$$
$$r \sim p(r|s)$$

The generation probability $p(r|s)$ is the production of the probability of observing a letter $\beta \in \mathcal{A}$ emitted from the strain s $p(\beta|\alpha, s)$, $\alpha \in \mathcal{A}$. Thus, $p(r|s) = \prod_i p(\beta_i|\alpha_i, s)$, where i indexes the alignment position.

During the inference, the assignment of a read r is conducted according to the following posterior probability,

$$p(s|r) \propto p(r|s)p(s) = \frac{a_s}{\sum_{s'} a_{s'}} \prod_i p(\beta_i|\alpha_i, s)$$

For the paired-end reads, both segments of a read pair shall occur in the same strain. To this end, the assignment of the second read r_2 is refined to follow the probability,

$$p(s|r_2, r_1) \propto p(r_2|s)p(r_1|s)p(s)$$

where $p(r_1|s)$ is the generation probability of the first read r_1 of a read pair. It would guarantee that the assignments of both segments of a read pair are consistent with each other at probability.

It is noted that we do not use an additional probability to open a new cluster in the above equation. The additional opening probability is a small probability, and used in the existed methods to allow for the creation of a new strain in order to avoid missing the possible strains. Unlike the existed methods, we generate the set of candidate strains using the POG structure, which guarantees the set of candidate strains cover all possible strains. Therefore, it is not necessary to use an additional opening probability in the formula.

Upon the assignment of sequencing reads R_s to a strain s, the parameters $p(\beta|\alpha, s)$, $\alpha, \beta \in \mathcal{A}$ are updated by

$$p(\beta_i|\alpha_i, s) = \frac{the\ count\ of\ (\beta, \alpha)\ in\ R_s}{the\ count\ of\ \alpha\ in\ R_s}$$

And also, the counting vector a is updated by the assignments.

To sum up, a graph walker visits the vertexes one level by one level. Suppose that a set of the most possible strains S^*_{l-1} is inferred at the $(l-1)$-th level. Thus, a set of

candidate strains S_l at the l-th level is constructed from S^*_{l-1} by extending a strain $s \in S^*_{l-1}$ with an outgoing vertex v' of v, where v is the ending vertex of s. For a new strain $s' \in S_l$ and its preceding strain s, the new counting is $a_{s'} = \frac{\chi(v,v')}{\chi(v)} a_s$, $\chi(v, v')$ is the number of reads covering both v and v', and $\chi(v)$ is the number of reads covering v. Next, a Dirichlet process mixture clustering is used to infer the distribution over the strains S_l by updating the counting vector a according to the read assignments. A strain $s \in S_l$ will be terminated if $\rho_s = \frac{a_s}{\sum_{s' \in S_l} a_{s'}}$ is less than a threshold τ. The threshold τ represents the bound of a low-abundance strain, and is allowed to specify by the users. After terminating the low-abundance strains, a subset of S_l is retained as the most possible strains S^*_l at the l-th level, and input to the next level. It is called streaming as it applies the Dirichlete process mixture clustering one level by one level. The following pseudocode summarizes the above procedure.

Pseudocode 2: StreamingDirichletClustering(G, R)
traverse the graph G one level by one level
at l-th level, construct the candidate strains S_l from the previous inferred strains S^*_{l-1}
infer a and $p(\beta \mid \alpha, s)$ using DirichletClustering(S_l, R)
construct S^*_l by terminating $\forall s \in S_l$ if $\rho_s < \tau$