Optimal_k - DB-graph inference by accurate sampling
(Fast and accurate) selection of parameters for genome assembly (by sampling)
Sampling the genomic assembly landscape
AssemblyAdvisor - insights on genomic content and assembly quality of sequencing libraries

1 Abstract

Motivation: There is no clear way on how to chose parameters k-mer size and abundance for a De Bruijn based de novo assembler. As *de novo* genome assembly is time consuming for large genomes, it is of importance to chose these parameters well in order to prevent multiple runs. Current software for estimating k only optimize certain features such as maximizing the number of genomic k-mers. There is a need for more clear objectives such as E-size or N50.

Results: We provide a method (optimal_k) to estimate average unitig length, N50 and E-size for all combinations of minimum abundance and k in one run. As unitigs are a foundation of the de Bruijn graph, estimating these quantities provides an understanding of the quality of a DBG based genome assembly as well as a good base for chosing the best combination of k and abundance. The estimations obtained by optimal_k are extremely accurate. [We also note that these estimations also accurately predict the best quality for DBG based assemblers that perform more steps such as tip removals, bubble popping and usage of paried end read information.]

2 Introduction

Mention that there are not many tools for computing optimal parameters at all. And make sure to mention that memry is not the issue. Mention the positives about our methods like speed and clear objective function but make sure to mention that it's memory requiring but thats not a problem if you are going to do the assembly anyway!!

A unitig of a graph is a maximal unary path. In the contig assembly phase, popular genome assemblers report a unitig decomposition of the assembly graph, after some artifacts have been been dealt with, like tip removal and bubble popping.

2.0.1 Choosing parameters in a DBG assembler

There are in particular two important parameters to chose for DBG assemblers, the k-mer size and the minimum k-mer abundance a_{min}. The abundance a of a k-mer is the number of times a k-mer occurs in the reads. a_{min} specifies the minimum abundance that a k-mer can have in order to be included as a node in the DBG. A read error generates k new k-mers in the graph (assuming that k is large enough so that the probability of seeing a random k-mer is negligible, *e.g.* around $k > 25$). As read errors occur in any sequencing technique, chosing $a = 1$ is infeasible as it will explode the size of the graph. Furthermore, as reads are non randomly occurring in *e.g.* Illumina reads (the most commonly used sequencing technique for assembly), $a = 2, 3$ might not eve suffice to remove sequencing errors in the DBG. Even though removing erroneous k-mers from the DBG controls the memory usage of an assembler and can help to remove spurious paths in the graph, the most important parameter is the k-mer size. Larger k will reduce the abundance of k-mers at any position on the genome, thus gaps are introduced where there is less than a_{min} k-mers that shares less than $k-1$ bases overlap. Also, with larger k, it is more probable for a k-mer to contian an error, thus, we decrease the number of k-mers belonging to the reference sequence after some given k (kmergenie). However, a genome contins more repetitive sequences with smaller k, thus there will be more repetitive nodes in the DBG for smaller nodes (see Figure XX). This suggests a trade-off when choosing k. The optimal k-mer size depends on the library coverage, error rate and genome composition. cite optimal-, kmergenie, k and say som smart things about it here

Mention that too high k can intruduce missassemblies even for unitigs as a repetitive region A,B—R—C,D appear unary if a and C are removed from the graph due to the increased k-mer length (if ARC anf BRD are true paths), this should be unlikely though. Illustrate this in the k=4 grapg in illustration with lost coverage over the CAA repeat (need one more nucleotide then in that case)

3 Methods

The general idea is to provide the user with metrics such as unitigs N50 and E-Size and average number of genomic vertices in a DBG for all possible k-mer sizes and abundances. We implement a FM-index data structure described in cite XX. This allows us to query a k-mer, its in and out neighbors in O() time. We furthermore derive formulas for how much we need to sample in order to reach a given accuracy on all our estimates.

Say that one of the main ideas is to do weighted sampling.

Say that we compute for all abundances at the same time.

Say that we can query every sampled k-mer in parallel.

3.1 Basic notions and algorithmic building blocks

We assume that the input consists of a set R of n reads. We denote by K_k the multiset of all k-mers in the reads, and by $|\mathsf{K}_k|$ its length. For example, if all reads have the same length r, then $|\mathsf{K}_k| = n(r - k + 1)$. Moreover, we denote by $\mathsf{DB}_{k,a}$ the de Bruijn graph with vertices of length k and *minimum abundance* a. That is, the set of vertices of $\mathsf{DB}_{k,a}$ is the set of all k-mers in the reads which occur at least a times in R, and two vertices of $\mathsf{DB}_{k,a}$ are connected by an arc if they have a suffix-prefix overlap of length $k - 1$. Let $V(\mathsf{DB}_{k,a})$ denote the set of vertices of $\mathsf{DB}_{k,a}$. For all $x \in \mathsf{K}_k$, let $\alpha(x)$ denote the abundance of k-mer x in R. We also denote by $\mathbb{I}(x, a)$ an indicator variable equal to 1 if the $\alpha(x) \geqslant a$, and to 0 otherwise.

We denote by $\delta^+_{k,a}(v)$ the number of out-neighbors of v in $\mathsf{DB}_{k,a}$, and by $\delta^-_{k,a}(v)$ the number of in-neighbors of v in $\mathsf{DB}_{k,a}$. A node v of $\mathsf{DB}_{k,a}$ is called *unary* if $\delta^-_{k,a}(v) = \delta^+_{k,a}(v) = 1$. We will also use a boolean $\mathsf{isUnary}_{k,a}(v)$ equal to true if and only if x is a unary node in $\mathsf{DB}_{k,a}$. If $\delta^-_{k,a}(v) = \delta^+_{k,a}(v) = 0$ then we say that v is an *isolated* node. A path in $\mathsf{DB}_{k,a}$ is called a *unitig* if all its internal vertices are unary, and its two extremities are not. When clear from the context, we will also use the term unitig to denote the *string* spelled by a unitig path in $\mathsf{DB}_{k,a}$.

Throughout the paper, for clarity we will use to the above conceptually clean definitions of de Bruijn graph. We should point out that in practice also reverse complements need to be taken into account. There are different ways of representing this information, but a widely accepted notion is the one of bi-directed de Bruijn graph [4]. The first application of bidirected graph for modeling DNA molecules was proposed in [3], and appear in popular works such as [2].

As mentioned, we index all reads as separate sequences in the RLCSA data structure. Given a pattern x of length k, this index can return the total number of occurrences of x in all of the indexed sequences, in time $O(k)$. Equivalently, given x we can obtain $\alpha(x)$ in $O(k)$ time. Similarly, we can compute $\delta^+_{k,a}(x)$ by querying the index for the four possible out-neighbors of x, namely $x[2..k]\mathtt{A}$ and $x[2..k]\mathtt{C}$, $x[2..k]\mathtt{G}$, $x[2..k]\mathtt{T}$, and for each of them compute their abundance. If this is greater than a, then it is a node in $\mathsf{DB}_{k,a}$.

A crucial difference with respect to building a de Bruijn graph for every value of k and a is the following one: for a given value of k, we can compute the desired estimates for all values of a at the same time, from the same queries to the index. This is possible thanks to the fact that a k-mer x is a node in all graphs $\mathsf{DB}_{k,a}$ with $a \leqslant \alpha(x)$. See Algorithm 1 for a snippet of pseudo-code on how to compute the out-degrees of x for all values of a, by just four queries to the index.

Algorithm 1: Computing the out-degrees $\delta^+_{k,a}(x)$ of a k-mer x, for all abundances $a \in [A_1, A_2]$; RLCSA is the index over the reads.

```
for $a = A_1$ to $A_2$ do
    $\delta^+_{k,a}(x) = 0$;
foreach $b \in \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$ do
    $y = x[2..k]b$;
    $\alpha(y) = \mathsf{RLCSA}.count(y)$;
    for $a = A_1$ to $\min(\alpha(y), A_2)$ do
        $\delta^+_{k,a}(x) = \delta^+_{k,a}(x) + 1$;
return $\delta^+_{k,a}(x)$.
```

We also recall here the Central Limit Theorem, as we will use it to guarantee a confidence interval of our estimations. Let Y be a random variable, from a distribution with finite mean μ and finite non-zero variance σ^2. By the Central Limit Theorem, the $100(1-\alpha)\%$ two-sided confidence interval of the sample mean $\overline{y}$ approaches

$$\left[\overline{y} - z_{\frac{\alpha}{2}} \frac{\sigma}{\sqrt{m}}\ ,\ \overline{y} + z_{\frac{\alpha}{2}} \frac{\sigma}{\sqrt{m}}\right]$$

as the number of samples m increases. In the above, $z_{\frac{\alpha}{2}}$ denotes the $\alpha/2$ quantile from the normal distribution. Notice that σ is the standard deviation of the random variable Y, while $\frac{\sigma}{\sqrt{m}}$ is the standard deviation of $\overline{y}$, since $\overline{y}$ is also a random variable varying from one sample to another. Below we will focus primarily on the distributions of sample statistics.

3.2 Sampling algorithms for the number of nodes and unitigs

Estimating the number of nodes of a dBG. We start by discussing how to obtain an accurate estimate for the number of nodes of $\mathsf{DB}_{k,a}$. This best illustrates our method, and this strategy will be further developed for the other estimates. We first express the number of nodes of $\mathsf{DB}_{k,a}$ as

$$|V(\mathsf{DB}_{k,a})| = \sum_{x \in \mathsf{K}_k} \frac{1}{\alpha(x)} \mathbb{I}(x,a).$$

Since $V(\mathsf{DB}_{k,a})$ is a subset of the multiset K_k, we can consider the proportion

$$p_{k,a} := \frac{|V(\mathsf{DB}_{k,a})|}{|\mathsf{K}_k|} = \frac{\sum_{x \in \mathsf{K}_k} \frac{1}{\alpha(x)} \mathbb{I}(x,a)}{|\mathsf{K}_k|} \in [0,1].$$

Algorithm 2: Computing the estimate $\hat{p}_{k,a}$ needed for the number of k-mers in the de Bruijn graph $\mathsf{DB}_{k,a}$, for all $a \in [A_1, A_2]$. The input is also a multiset $\{x_1, \ldots, x_m\}$ of k-mers from K_k.

```
for $a = A_1$ to $A_2$ do
    $sum[a] = 0$;
for $i = 1$ to $m$ do
    $\alpha(x_i) = \mathsf{RLCSA}.count(x_i)$;
    for $a = A_1$ to $\min(\alpha(x_i), A_2)$ do
        $sum[a] = sum[a] + 1/\alpha(x_i)$;
for $a = A_1$ to $A_2$ do
    $\hat{p}_{k,a} = sum[a]/m$;
return $\hat{p}_{k,a}$, for all $a \in [A_1, A_2]$.
```

We can estimate $p_{k,a}$ by sampling a multiset $\{x_1, \ldots, x_m\}$ of k-mers from K_k, and taking

$$\hat{p}_{k,a} := \frac{\sum_{i=1}^{m} \frac{1}{\alpha(x_i)} \mathbb{I}(x_i, a)}{m}.$$

Therefore, we get an estimate of $X_{k,a} := |V(\mathsf{DB}_{k,a})|$ as $\hat{X}_{k,a} = \hat{p}_{k,a}|\mathsf{K}_k|$. Notice that if we sample all k-mers exactly once, we get $\hat{X}_{k,a} = \frac{|V(\mathsf{DB}_{k,a})|}{|\mathsf{K}_k|}|\mathsf{K}_k| = |V(\mathsf{DB}_{k,a})|$. Analogously to Algorithm 1, we can implement this procedure for all given abundances with just m queries to the RLCSA index: see Algorithm 2 for a pseudo-code.

In order to obtain a confidence interval for $\hat{p}_{k,a}$, and thus for $\hat{X}_{k,a}$, we need to decide what is the value of m. Denote for brevity $p_{k,a}$ by just p, and the sample estimate $\hat{p}_{k,a}$ obtained from m samples by just $\hat{p}_m$. The nodes of $\mathsf{DB}_{k,a}$ and K_k correspond to binomial data with a proportion p, and population standard deviation $\sigma = \sqrt{p(1-p)}$. By the Central Limit Theorem recalled in Sec. 3.1, we have that the $100(1-\alpha)\%$ confidence interval of $\hat{p}_m$ is

$$\left[\hat{p}_m - z_{\frac{\alpha}{2}} \sqrt{\frac{p(1-p)}{m}}, \hat{p}_m + z_{\frac{\alpha}{2}} \sqrt{\frac{p(1-p)}{m}}\right].$$

During our sampling procedure, we compute the sample estimate $\hat{p}_m$ and we estimate its standard deviation σ as $\hat{\sigma} := \sqrt{\hat{p}_m(1-\hat{p}_m)}$. If the margin of error $z_{\frac{\alpha}{2}}\hat{\sigma}$ is less than $\varepsilon \hat{p}_m$, where $\varepsilon \in [0, 1)$ is an input accuracy parameter, we stop and report $\hat{p}_m$ as estimate for p; otherwise we increase the sample size m.

Estimating the number of unitigs of a dBG. Let $\mathsf{U}_{k,a}$ denote the set of all unitigs of $\mathsf{DB}_{k,a}$. We now derive a simple combinatorial expression for $|\mathsf{U}_{k,a}|$, which is key in this sampling phase. Let $\mathsf{ST}_{k,a}$ denote the set of start nodes of the unitigs of $\mathsf{DB}_{k,a}$. Since every node v in $\mathsf{ST}_{k,a}$ is either an isolated node, or it is a start node of some unitig(s) (each of these unitigs start with v and then continue to each of its out-neighbors), we can write

$$|\mathsf{U}_{k,a}| = \sum_{v \in \mathsf{ST}_{k,a}} \max(1, \delta^+_{k,a}(v)).$$

It is easy to tell if a node of $\mathsf{DB}_{k,a}$ is a start node of some unitigs: either it has at least two out-neighbors, or it has one out-neighbor, but at least two in-neighbors, or it is an isolated vertex. For every $x \in \mathsf{K}_k$, let the boolean variable $\mathsf{isStart}_{k,a}(x)$ be defined as

$$\begin{aligned}\mathsf{isStart}_{k,a}(x) :=& \delta^+_{k,a}(x) \geqslant 2 \text{ or} \\ & (\delta^+_{k,a}(x) = 1 \text{ and } \delta^-_{k,a}(x) \neq 1) \text{ or} \\ & (\delta^+_{k,a}(x) = 0 \text{ and } \delta^-_{k,a}(x) = 0).\end{aligned}$$

Therefore, we can obtain the number of unitigs also by summing over all k-mers in the reads, as done for the number of nodes:

$$|\mathsf{U}_{k,a}| = \sum_{\substack{x \in \mathsf{K}_k \text{ such that} \\ \mathsf{isStart}_{k,a}(x)}} \max\left(\frac{1}{\alpha(x)}\mathbb{I}(x,a), \frac{1}{\alpha(x)}\mathbb{I}(x,a)\delta^+_{k,a}(x)\right). \tag{1}$$

Consider the ratio $q_{k,a}$ between the number of unitigs and all k-mers in the reads

$$q_{k,a} := \frac{|\mathsf{U}_{k,a}|}{|\mathsf{K}_k|}.$$

Observe that $q_{k,a} \in [0,1]$ since every unitig contains at least one k-mer, thus $|\mathsf{U}_{k,a}| \leqslant |\mathsf{K}_k|$. We can analogously estimate $q_{k,a}$ as above, after sampling a multiset $\{x_1, \ldots, x_m\}$ of k-mers from K_k, as

$$\hat{q}_{k,a} := \frac{1}{m} \sum_{\substack{i \in [1,m] \text{ such that} \\ \mathsf{isStart}_{k,a}(x_i)}} \max\left(\frac{1}{\alpha(x_i)}\mathbb{I}(x_i,a), \frac{1}{\alpha(x_i)}\mathbb{I}(x_i,a)\delta^+_{k,a}(x_i)\right).$$

The estimate of $Y_{k,a} := |\mathsf{U}_{k,a}|$ is then $\hat{Y}_{k,a} = \hat{q}_{k,a}|\mathsf{K}_k|$. Similarly to $X_{k,a}$, sampling all k-mers will give $\hat{Y}_{k,a} = |\mathsf{U}_{k,a}|$. As in Algorithm 2, for a given value of k, we can compute all values $\hat{q}_{k,a}$ for all abundances a in a given interval $[A_1, A_2]$ at the same time. Since $q_{k,a} \in [0,1]$, we can obtain the sample size m identically as done above for the number of nodes of $\mathsf{DB}_{k,a}$.

3.3 Sampling algorithms for the mean length and E-size of the unitigs

We first discuss how to estimate the E-size of the unitigs of $\mathsf{DB}_{k,a}$, and then briefly show how this technique provides an estimate for the mean length of the unitigs. The E-size [5] of the set $\mathsf{U}_{k,a}$ of unitig strings of $\mathsf{DB}_{k,a}$ is defined as the expected length of the unitig strings of $\mathsf{DB}_{k,a}$. More precisely, this is the expected unitig string length covering any position on the concatenation of all unitig strings. Formally,

$$\mathsf{E}_{\mathsf{size}}(\mathsf{U}_{k,a}) := \sum_{w \in \mathsf{U}_{k,a}} |w| P(w) = \sum_{w \in \mathsf{U}_{k,a}} |w| \frac{|w|}{\sum_{w' \in \mathsf{U}_{k,a}} |w'|} = \frac{\sum_{w \in \mathsf{U}_{k,a}} |w|^2}{\sum_{w \in \mathsf{U}_{k,a}} |w|}, \tag{2}$$

where $|w|$ denotes the length of the string spelled by the unitig w and $P(w)$ the probability of sampling a position in the concatenation of all unitigs.

In the ideal setting when the set of unitigs partitions the genome, the E-size corresponds to the expected unitig length covering any position of the genome. In a de novo assembly this might not be true, due to unsequenced regions, allele splitting and overlapping unitig ends. However, the variation of E-size across different assemblies of a given genome is an informative metric of the assembly contiguity [5].

To estimate the E-size, we will sample m unitigs, and use their string lengths $x_1, x_2, \ldots, x_m$ to estimate it. However, notice that the E-size metric is independent of the abundances of the k-mer of the unitigs. Since our sampling procedure is based on sampling k-mers from K_k, we need to remove the bias introduced by their different abundances.

We first describe this sampling procedure, which produces a multiset W of unitigs of $\mathsf{DB}_{k,a}$, as follows. We choose a k-mer $x \in \mathsf{K}_k$ at random. If x is a start node of some unitig of $\mathsf{DB}_{k,a}$ (that is, $\mathsf{isStart}_{k,a}(x)$ holds), then we output all the unitigs starting at x. These unitigs can be obtained by traversing the graph by following each of the out-neighbors of x as long as the traversed path is unary. With the RLCSA data structure, we are able to organize this visit so that we obtain the unitigs for all abundances simultaneously. In Algorithms 3 and 4 we give pseudo-codes of this procedure.

Given a unitig $w = (v_1, v_2, \ldots, v_t)$ of $\mathsf{U}_{k,a}$, let $\alpha(w) := \alpha(v_1)$ be the defined as the *abundance* of w. Observe that if every k-mer in K_k is sampled exactly once and W_{all} denotes the resulting multiset of sampled unitigs, then each unitig w of $\mathsf{U}_{k,a}$ appears $\alpha(w)$ times in W_{all}. Therefore, we can express the E-size of the set $\mathsf{U}_{k,a}$ by normalizing the probability of w with $1/\alpha(w)$. This gives the following equivalent expression for the E-size:

$$\mathsf{E}_{\mathsf{size}}(\mathsf{U}_{k,a}) = \sum_{w \in W_{all}} |w| \frac{|w|\frac{1}{\alpha(w)}}{\sum_{w' \in W_{all}} |w'| \frac{1}{\alpha(w')}} = \frac{\sum_{w \in W_{all}} \frac{|w|^2}{\alpha(w)}}{\sum_{w \in W_{all}} \frac{|w|}{\alpha(w)}}. \tag{3}$$

We now discuss how many unitig samples m we need and how to combine their string lengths into an estimate for the E-size. We first give another expression of the E-size of the unitigs of $\mathsf{DB}_{k,a}$, which allows obtaining confidence interval for its estimate. If $x \in \mathbb{N}$, then the string lengths of all unitigs in $\mathsf{DB}_{k,a}$ induces a distribution $f(x)$, and let X be random variable over $f(x)$. From the definition (2), we get

$$\mathsf{E}_{\mathsf{size}}(\mathsf{U}_{k,a}) = \frac{\sum_{w \in \mathsf{U}_{k,a}} |w|^2}{\sum_{w \in \mathsf{U}_{k,a}} |w|} = \frac{\sum_{x \in \mathbb{N}} x^2 f(x)}{\sum_{x \in \mathbb{N}} x f(x)} = \frac{\mathrm{E}[X^2]}{\mathrm{E}[X]}. \tag{4}$$

We sample m unitigs with the procedure described above: let $x_1, x_2, \ldots, x_m$ be their string lengths and let $a_1, \ldots, a_m$ be their abundances. Denote $A_1 := \sum_{i=1}^{m} \frac{1}{a_i}$ and $A_2 := \sum_{i=1}^{m} \frac{1}{a_i^2}$. Equation (4) shows that we can estimate

$$\mathrm{E}[X] \text{ as } \overline{x}_m := \frac{1}{A_1} \sum_{i=1}^{m} \frac{x_i}{a_i} \qquad \text{and} \qquad \mathrm{E}[X^2] \text{ as } \overline{x^2}_m := \frac{1}{A_1} \sum_{i=1}^{m} \frac{x_i^2}{a_i}.$$

Recall that the estimates $\overline{x}_m$ and $\overline{x^2}_m$ are random variables themselves, since for a different sample we will get different estimates. For a given sample size m, let X_m^1 and X_m^2 be random variables over the sample distribution of $\overline{x}_m$ and of $\overline{x^2}_m$, respectively. Let also Y_m be the random variable defined as

$$Y_m := \frac{X_m^2}{X_m^1}.$$

Since $\mathrm{E}[Y_m] = \mathsf{E}_{\mathsf{size}}(\mathsf{U}_{k,a})$ (which holds for any given unbiased estimator), we can report the sample estimate $\overline{y}_m$ for Y_m as our estimate for the E-size.

In order to get a confidence interval of $\overline{y}_m$, we need to derive the standard deviation of Y_m, which we denote by σ_{Y_m}. If ... is ..., then the first order Taylor expansion gives a good estimation of $\sigma_{Y_m}^2$ [1]:

$$\sigma_{Y_m}^2 = \mathrm{Var}\left[Y_m\right] = \mathrm{Var}\left[\frac{X_m^2}{X_m^1}\right] \approx \frac{\mathrm{Var}\left[X_m^2\right]}{\mathrm{E}\left[X_m^1\right]^2} - 2\frac{\mathrm{E}\left[X_m^2\right]}{\mathrm{E}\left[X_m^1\right]^3}\mathrm{Cov}\left[X_m^2, X_m^1\right] + \frac{E\left[X_m^2\right]^2}{E\left[X_m^1\right]^4}\mathrm{Var}\left[X_m^1\right].$$

We estimate σ_{Y_m} by $\hat{\sigma}_{Y_m}$, obtained by estimating the expressions in the above formula as:

- $\mathrm{E}[X_m^1]$ as $\overline{x}_m$, and $\mathrm{E}[X_m^2]$ as $\overline{x^2}_m$;
- $\mathrm{Var}[X_m^1]$ as
$$\frac{A_2}{(A_1)^2 - A_2} \frac{\sum_{i=1}^{m} \frac{1}{a_i}(x_i - \overline{x}_m)^2}{A_1},$$
- $\mathrm{Var}[X_m^2]$ as
$$\frac{A_2}{(A_1)^2 - A_2} \frac{\sum_{i=1}^{m} \frac{1}{a_i}(x_i^2 - \overline{x^2}_m)^2}{A_1};$$

Algorithm 3: Computing the lengths of all unitigs starting at a k-mer x in $\mathsf{DB}_{k,a}$, for all abundances in an interval $[A_1, A_2]$. The output is an array *length* of lists such that $length[a]$ is the list of lengths of all unitigs starting at x in $\mathsf{DB}_{k,a}$, for all $a \in [A_1, A_2]$. The sub-routine **extendUnitig**(y, A_1, A_2) is described in Algorithm 4.

```
for $a = A_1$ to $A_2$ do
    $length[a] = \emptyset$;
// We compute the set of abundances for which $x$ is a start node
$\alpha(x) = \mathsf{RLCSA}.count(x)$;
$start = \emptyset$;
for $a = A_1$ to $A_2$ do
    if $\mathsf{isStart}_{k,a}(x)$ and $a \leqslant \alpha(x)$ then
        $start.append(a)$;
if $start \neq \emptyset$ then
    // We extract the min and max abundance in the set $start$
    $A'_1 = \min(start)$;
    $A'_2 = \max(start)$;
    // For each possible out-neighbor $y$ of $x$
    foreach $b \in \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$ do
        $y = x[2..k]b$;
        $\alpha(y) = \mathsf{RLCSA}.count(y)$;
        // We try extending the path starting with $x, y$ if $y$ exists in the graph
           and as long as there is an abundance for which this path is unary
        $extension\_length = \mathbf{extendUnitig}(y, A'_1, \min(\alpha(y), A'_2))$;
        foreach $a \in start$ do
            if $a \leqslant \min(\alpha(y), A'_2)$ then
                $length[a].append(1 + extension\_length[a])$
return $length[a]$, for all $a \in [A_1, A_2]$.
```

- $\mathrm{Cov}[X_m^2, X_m^1]$ as

$$\frac{A_2}{(A_1)^2 - A_2} \frac{\sum_{i=1}^{m} \frac{1}{a_i}(x_i - \overline{x}_m)(x_i^2 - \overline{x^2}_m)}{A_1}.$$

Applying the Central Limit Theorem, we obtain a confidence interval for our E-size estimate $\overline{y}_m$ as

$$\left[\overline{y}_m - z_{\frac{\alpha}{2}} \sigma_{Y_m} \ , \ \overline{y}_m + z_{\frac{\alpha}{2}} \sigma_{Y_m}\right],$$

where we estimate the standard deviation of $\overline{y}_m$ with $\hat{\sigma}_{Y_m}$. If the margin of error $z_{\frac{\alpha}{2}} \hat{\sigma}_{Y_m}$ is less than $\varepsilon \overline{y}_m$, where $\varepsilon \in [0, 1)$ is an input accuracy parameter, we stop and report $\overline{y}_m$ as estimate for the E-size; otherwise we continue sampling more unitigs.

The above immediately lead to a confidence interval for the mean unitig length $\mathrm{E}[X]$: we estimate it as $\overline{x}_m$, and obtain its confidence interval by estimating its standard deviation as the square root of the above estimation for $\mathrm{Var}[X_m^1]$.

4 Results and discussion

5 Conclusions

References

[1] Haym Benaroya, Seon Mi Han, and Mark Nagurka. *Probability Models in Engineering and Science*. CRC Press, 2005.

[2] Erwan Drezen, Guillaume Rizk, Rayan Chikhi, Charles Deltel, Claire Lemaitre, Pierre Peterlongo, and Dominique Lavenier. GATB: Genome Assembly & Analysis Tool Box. *Bioinformatics*, 30(20):2959–2961, 2014.

[3] John Dimitri Kececioglu. *Exact and approximation algorithms for DNA sequence reconstruction*. PhD thesis, Tucson, AZ, USA, 1992.

[4] Paul Medvedev, Konstantinos Georgiou, Gene Myers, and Michael Brudno. Computability of Models for Sequence Assembly. In Raffaele Giancarlo and Sridhar Hannenhalli, editors, *Algorithms in Bioinformatics, 7th International Workshop, WABI 2007, Philadelphia, PA, USA, September 8-9, 2007, Proceedings*, volume 4645 of *Lecture Notes in Computer Science*, pages 289–301. Springer, 2007.

[5] S. L. Salzberg, a. M. Phillippy, a. V. Zimin, D. Puiu, T. Magoc, S. Koren, T. Treangen, M. C. Schatz, a. L. Delcher, M. Roberts, G. Marcais, M. Pop, and J. a. Yorke. GAGE: A critical evaluation of genome assemblies and assembly algorithms. *Genome Research*, 22(3):557–567, December 2012.

Algorithm 4: Extending a unitig. The input is a k-mer y and an interval $[A_1, A_2]$ such that $\alpha(y) \in [A_1, A_2]$, and the output is, for every $a \in [A_1, A_2]$, the length of the longest path starting with y such that all of its nodes, except for the last, are unary in $\mathsf{DB}_{k,a}$.

```
extendUnitig($y, A_1, A_2$)
    for $a = A_1$ to $A_2$ do
        $length[a] = 0$;
    $A'_1 = A_1$; $A'_2 = A_2$;
    while $A'_1 \leqslant A'_2$ do
        $advanced = $ false;
        for $a = A'_1$ to $A'_2$ do
            $length[a] = length[a] + 1$;
            if $\mathsf{isUnary}_{k,a}(y)$ and (not $advanced$) then
                update $y$ so that it equals its unique out-neighbor;
                $advanced = $ true;
            if $\delta^+_{k,a}(y) = 0$ then
                $A'_2 = a - 1$;
            if $\delta^+_{k,a}(y) > 1$ or $\delta^-_{k,a}(y) > 1$ then
                $A'_1 = a + 1$;
    return $length[a]$, for all $a \in [A_1, A_2]$.
```