Genome Assembly with Long Noisy Reads

Amir Kadivar

December 2015

abstract *We consider the problem of de novo genome assembly with reads characterized by relatively large length, relatively high error rate, especially in homopolymeric regions. We propose two solutions within the overlap-layout-consensus framework which can be put to use in tandem. First, we consider an algorithm to perform alignments in a "condensed" alphabet where homopolymeric stretches are reduced to single letters. Second, we consider an algorithm for deciding whether two reads are overlapping without any alignment computation but only using the "shift" distribution of exactly matching k-mers. We demonstrate that these two algorithms, respectively improve the accuracy and running time of overlap discovery.*

Background

Second generation sequencing technologies are characterized by relatively short reads (on average 300 bps), low error rates, and under 10X coverage with the main challenge of de novo assembly being the resolution of repeat regions. Third generation sequencing technologies (Pacific Biosciences' SMRT, Oxford Nanopore, and the like) are characterized by longer reads (on average 5-15 kbps), high error rates (up to 20 %), and up to 40X coverage. As such, single-molecule sequencing drastically simplifies the problem posed by repetitive structures. However, established assembly schemes do not scale well to the dimensions and low accuracy levels of such technologies: overlap-layout-consensus schemes suffer due to the high error rates (specifically the high indel rates) and de-Bruijn graph schemes suffer from the large read lengths. As a final note on the latter family of assembly schemes we note that the main challenge for applying the Eulerian path approach to assembly with long reads is to solve the *spectral alignment problem* (Pavel A. Pevzner 2001). The dynamic programming algorithm that solves this problem could be modified (non-trivially) to suit assembly with long reads and high error rates (Chaisson, Pevzner, and Tang 2004).

There has been successful attempts at incorporating SMRT reads into assembly pipelines. Most commonly, long reads are used in tandem with second generation techniques (Koren et al. 2012) mainly in the finishing process and to resolve repetitive structures. Here we concern ourselves with the problem of assembly using only SMRT reads. Proposals for this problem include the HGAP (Chin et al. 2013) and MHAP (Berlin et al. 2015) processes. Similar to our first algorithm, compressing the homopolymeric stretches has also been suggested (Au et al. 2012).

Problem Definition and Scope

In this section we formulate the scope of this report and briefly consider its boundary with the rest of the assembly pipeline.

Given a sequence of reads $(R_n)_{n=1}^N$ we wish to find the *overlap* graph $G = (V, E)$ which is a *weighted directed acyclic* graph whose vertices V is $\{R_n\}_{n=1}^N$. Two reads R_i and R_j are overlapping with score w, denoted by $R_i \overset{w}{\mapsto} R_j$, if a suffix of R_i aligns with a prefix of R_j.[1] Our goal is to find an approximate $\hat{E}$ to the set E of all such edges and we define the sensitivity and specifity of our results in terms of the number of edges recovered from the *true* overlap graph (built using a known reference genome).[2] That is we wish to minimize both of:

$$f.n. = \frac{|E \setminus \hat{E}|}{|E|} \text{ , and } f.p. = \frac{|\hat{E} \setminus E|}{|E|}$$

Suppose once a reasonable $\hat{E}$ is found we proceed as follows to complete the assembly problem: **1.** find a *layout path* by

[1]This is accurate as long as the alignment between the R_i and R_j is not substring. The accurate formulation is this: $R_i \overset{w}{\mapsto} R_j$ if the high-scoring alignment between the two reaches the boundary of both sequences and starts at the beginning of R_j.

[2]Since we will also be referring to the sensitivity of the rest of the assembly pipeline to different kinds of error introduced in the overlap stage, we reserve the term "sensitivity" for such purposes and refer to false negative and false positive rates instead.

solving a heaviest path problem over the overlap DAG, **2.** build a "scaffold" by solving a small consensus problem over the above layout path, **3.** align all vertices that were left out of the layout path onto the scaffold and return a complete layout path, **4.** solve the consensus problem as usual.

This report is only marginally concerned with the above steps. However, we wish to classify the sensitivity of the above steps on different kinds of errors introduced in the overlap stage.

Since the real information content of the overlap graph is its heaviest path, false negatives, esp. those with weak edges (i.e not recognizing weakly overlapping reads), are relatively tolerable: a missing edge can only cost us anything only if it belongs to the true heaviest path. Similarly, false positives are more dangerous, esp. those with large weights, since they may change the order of vertices in the heaviest path in their favor.

We say that $\hat{E}$ is *consistent* with E if its heaviest path $(\hat{v}_n)_{n=1}^{\hat{k}}$ is a *subsequence* of the true heaviest path $(v_n)_{n=1}^{k}$ of E. As long as the layout path produced by the first step is large enough and consistent with the true graph, the overlap stage is successful. This reasonably corresponds to our formulation of f.p.'s and f.n.'s.

Cycle breaking

The resulting overlap graph $\hat{G} = (V, \hat{E})$ may not be acyclic. However, a very low f.p. rate implies that $\hat{G}$ is either acyclic or not far from it. In our formulation, cycle breaking becomes the *feedback arc set* problem: find the lightest subset $\hat{E}^\circ$ of edges in $\hat{E}$ such that $(V, E \setminus \hat{E}^\circ)$ is acyclic. In all the work presented here cycle breaking is delegated to the igraph package which supports an optimal, but slow (exponential complexity), integer programming algorithm (Festa, Pardalos, and Resende 1999) and a suboptimal, but fast, algorithm relying on Eades' heuristic (Eades, Lin, and Smyth 1993).

Framework and Terminology

Our overlap discovery framework relies fully on k-mer methods (also known as l-tuples or words). That is, all analysis begins with indexing all words of length k observed in all reads and proceeds by finding exactly-matching segments between each pair of reads. A *segment* z for reads R_i and R_j is any pair of substrings of R_i and R_j (with potentially unequal lengths) together with an alignment between the two substrings. Two segments overlap if both their corresponding substrings in R_i and R_j are overlapping. A segment is *exactly-matching*, referred to as a *seed*, if its alignment contains no mismatches or indels. A seed is *maximal* if it cannot be extended in either direction by an exact match. Any set of seeds can be reduced to a maximal set of segments whose members are necessarily non-overlapping. A segment for reads R_i and R_j is *fully extended* if its alignment is an overlap (i.e suffix-prefix) alignment of R_i and R_j. The *shift* of a segment z for reads R_i and R_j with starting positions z_i and z_j in the corresponding sequences is the integer $z_i - z_j$.

Our first algorithm builds the overlap graph by trying to find, for any given pair of reads, a fully-extended segment with reasonable score. The scheme for extending segments in reasonable time and space constraints is fairly simple. The only twist is that all alignments during seed extension are performed in a "condensed" alphabet, referred to as the *h.p.-condensed* alphabet.

The second algorithm builds the overlap graph by calculating, for each pair of reads, a measure of "peakedness" of the distribution of the shifts of all their observed seeds. The intuition is that if two reads are overlapping there probably is a concentration of seeds with very close shift values. Since the two approaches are not mutually exclusive a natural hybrid of the two is also proposed.

Alphabet translation

The idea behind performing alignments in the h.p.-condensed alphabet is to capture the fact that indels are significantly more likely in homopolymeric stretches in SMRT sequencing. In order to avoid higher order Markov interactions in the dynamic programming algorithm we translate the reads and their seeds into the condensed alphabet, symbolically represented as

$$\{A, C, G, T\} \times \mathbb{N} = \{x_n; x \in \{A, C, G, T\}, n \in \mathbb{N}\}.$$

However, for the computational problem to be well-posed we need the condensed alphabet to be finite. Therefore, the translation process requires a parameter, referred to as the *h.p. max. length* h, which reduces the condensed alphabet to $\{A, C, G, T\} \times \mathbb{N}_{\leq h}$. That is, all homopolymeric stretches longer than h are considered to have length h.[3] We let $h = 5$ hereafter.

[3]This comes with a caveat: if the source alphabet sequence contains homopolymeric stretches that are longer than h, the condensing process is lossy in that expanding a condensed sequence does not necessarily give the original sequence back. This turns out to not be a limiting problem.

Alignment in the condensed alphabet

The library of sequence alignment methods implemented for this project operates on sequences of arbitrary alphabets to accommodate for h.p.-condensed sequences and thus any typical dynamic programming alignment algorithm may proceed as usual in the condensed alphabet.

Calculation of scores

To obtain substitution and gap scores in the condensed alphabet there are, in principle, two alternatives: **1.** calculate substitution *probabilities* from substitution probabilities in the original alphabet together with gap probabilities within homopolymeric stretches and take their log odds against a null hypothesis as scores, or **2.** calculate substitution scores in the original alphabet first using substitution probabilities. Then use substitution *scores* in the original alphabet together with a homopolymeric gap score to derive substitution scores in the condensed alphabet.

It is not yet clear how substitution probabilities in the condensed alphabet can be systematically derived from those in the original sequence.[4] Therefore, we use the second approach.

The conversion of substitution probabilities to scores in the original alphabet is the usual log odds calculation. That is, score of substituting letter x by letter y where $x, y \in \{A, C, G, T\}$ is

$$S(x \to y) = \log[(1-g)\Pr(x \to y)] - \log[\Pr(y)]$$

where g is the gap probability and $\Pr(y)$ is the null-hypothesis probability of y appearing at any position of a random sequence.

To calculate substitution scores in the condensed alphabet we proceed as follows: let x_i, y_j denote any two condensed alphabet letters. When $x = y$ the substitution cost is

$$S(x_i \to x_j) = \min(i,j) S(x \to x) + G_h(|i-j|)$$

where $G_h(n) = ng_h$ is the linear homopolymeric gap penalty and when $x \neq y$ it is

$$S(x_i \to y_j) = \min(i,j) S(x \to y) + G(|i-j|)$$

where $G(n) = g_o + ng_e$ is the usual affine/linear gap penalty in the original alphabet.

[4]The problem definition may very well be ill-posed. For example, consider the problem of aligning `AAAAAA` and `AAACCC`. The corresponding condensed sequences are A_6 and A_3C_3 with no way of out of an alignment with 3 indels in the original alphabet.

Content-dependent gap scores

When aligning sequences in the condensed alphabet the usual linear/affine gap penalty which only takes into consideration the length of a gap has undesirable consequences.[5] To overcome this, we use gap scores that are *content-dependent* in that the extension score of gaps depends on the substring that is inserted or deleted. The cost of a gap of length n inserting or deleting a substring $s = (s_i)_{i=1}^{n}$ is

$$G(s) = g_o + \sum_{i=1}^{n} g(s_i)$$

where $g(\cdot)$ is the content-dependent gap score of letter s_i. For letter x_k in the condensed alphabet we have $g(x_k) = kg_e$ where g_e is the usual gap extend score in the original alphabet.

Seed extension

At first, if possible, we can rule out a portion of read pairs based on the observation that they have "too few" seeds in common. The threshold m_z of the minimum number of seeds for a pair of reads to be considered *potentially homologous* is naturally dependent on k, the word length and only applicable when k is large enough (see section on parameters).

Seed extension is performed by sliding a window along the two sequences going forward and backward from a given seed until either the score begins to deteriorate or a sequence boundary is reached. For this purpose, two special variations of the Smith-Waterman algorithm are implemented: *start/end-anchored overlap* alignments. A start(end)-anchored alignment is one that is required to begin (end) at the top left (bottom right) corner and to end (begin) somewhere on the bottom or right (top or left) edges of the dynamic programming table. A seed is extended forward (backward) by repeated start(end)-anchored overlap alignments with a fixed *window* size w. Each window is a $w \times w$ box in the dynamic programming table starting (ending) at the exact position that the last alignment ended (started).

Detecting deteriorating scores is done by means of two parameters (both dependent on w):

- The *drop threshold* D_w which is the score threshold for alignments over a window $w \times w$ to be considered "bad",

[5]For example, the sequences $A_2C_7T_3$ and A_1T_4 can get a positive score despite having a 7-character long gap.

- The *max. successive drops* M_w which is the number of successive windows where the seed extends "badly" after which a seed is not further pursued.

Naturally, we need to adjust D_w and M_w to the value of w. In the results presented here this tuning is only done empirically. An alternative formulation is discussed in future work.

Safe margins

A common category of f.p.'s is that of those caused by *mostly-overlapping* sequences. Two reads are mostly-overlapping if their correct overlap alignment starts and/or ends very close to the diagonal of the dynamic programming table. In such cases the direction of the overlap is not robustly determined by the optimal alignment hence potentially reversing the direction of a heavy edge in the overlap graph. Due to the high sensitivity to f.p.'s and the small information content of such read pairs, we choose to not add an edge (in either direction) to the graph if two reads are mostly-overlapping.[6]

Seed extension in the condensed alphabet

The assembly line can be modified in two places to use condensed alphabets. During *indexing* reads can be indexed in a condensed alphabet and thus seeds are in the condensed alphabet to begin with. *Seed extension* can be performed in the condensed alphabet regardless of whether indexing, too, was done in the same fashion (and regardless of the h.p. max. length during indexing). The motivation of seed extension in the condensed alphabet as been already discussed. We now consider the consequences of indexing in the condensed alphabet.

First, suppose indexing is done using an h.p. max. length of 1. This has the potential added benefit that we do not miss any seed because of homopolymeric indels. It also has the downside of introducing a large amount of noise (all the seeds that are only seeds because they are h.p.-condensed). In practice the latter negative force was dominant and since there is no smooth way of balancing this trade-off (increasing the h.p. max. length to 2 drops an arbitrary number of seeds) this idea was dropped.

Second, suppose indexing is done in the original alphabet (as it is a requirement for shift distribution analysis discussed below). Then for seed extension, the seeds themselves must be condensed which is a non-trivial problem of its own: **1.** seed boundaries may not, and typically do not, coincide with boundaries of homopolymeric stretches, and **2.** condensing seeds requires recalculating their coordinates in the condensed sequence which is a computationally demanding task.

[6]An alternative would be to add both edges to the graph or to remove one of the reads entirely from the graph.

The first problem is dealt with by the following lossy rules:

- If the seed does not *begin* at a homopolymeric boundary (e.g. the second 4-mers of `AAACC` and `TAACC`) the seed is ignored.
- If a seed does not *end* at a homopolymeric boundary (e.g the first 4-mers of `AACCC` and `AACCT`) the longest possible segment is reported which is exactly-matching up to its last letter in the condensed alphabet (i.e in previous example A_2C_2 and A_2C_3 is reported as the condensed segment).

The second problem is dealt with by another lossy rule which guarantees only a single pass of each read pair to condense all their seeds: seeds are condensed "in order" and those that conflict the order are ignored. The following section describes what this ordering is and what it means for a seed to conflict the order.

Seed ordering Fix any two reads R_i and R_j. All seeds for R_i and R_j have a natural *partial order*: Let z and z' be two seeds with starting positions (z_i, z_j) and (z'_i, z'_j) in R_i and R_j, respectively. Then z and z' are *comparable* if:

$$(z_i - z'_i)(z_j - z'_j) \geq 0$$

that is, both coordinates have the same order. We can forcefully extend this partial order to a *total order* by letting $z \leq z'$ if:

$$z_i < z'_i$$

or if

$$z_i = z'_i \text{ , and } z_j \leq z'_j$$

Consequently, seeds that are not partial-order-comparable with their immediate total-order predecessors *conflict* with the order in the sense that their coordinates cannot be calculated trivially within the single pass of R_i and R_j.[7]

[7]For example, in the seeds with coordinates $(0, 1), (1, 4), (2, 2), (2, 5)$ are in increasing order according to the total-order but the third seed is not partial-order comparable to the second seed and is thus ignored.

Shift analysis

Now we introduce the second algorithm for overlap discovery with long, low-accuracy reads. Let R_i and R_j be any two reads and let them have a set of n seeds $\{z^k\}_{k=1}^n$ where seed z^k has coordinates (z_i^k, z_j^k). Let the shifts of the seeds be $\{d_k\}_{k=1}^n$ where

$$d_k = z_i^k - z_j^k$$

Each seed, through its coordinates, implies a certain offset for the correct overlap alignment of R_i and R_j. The core idea is that if R_i and R_j are actually overlapping reads, there must be concentration of shift values. That is, if one looks at the histogram of $\{d_k\}_{k=1}^n$ there is a certain "peakedness" close to the correct shift between the sequences. In fact, one can visually verify that this is typically the case.[8]

The advantage of this scheme is that one can, in principle, decide whether two reads are overlapping or not without a single cycle of computation spent on alignment. The only difficulty is in formulating a statistical measure of this "peakedness" that successfully differentiates the shift distribution of strongly overlapping read pairs from the rest. In what follows we present a simple such measure. There are potential alternatives discussed in future work.

First note that given the set of shifts $\{d_k\}_{k=1}^n$ for two reads R_i and R_j we must have, for every k

$$-|R_j| \leq d_k \leq |R_i|$$

We now build the distribution $f(\cdot)$ of shifts by sliding a window of length L along the set of values such that for any $d \in \{-L - |R_j|, \ldots, |R_i|\}$ we have:

$$f(d) = \Big| \{k; d \leq d_k \leq d + L\} \Big|$$

We than calculate the *peak ratio* $r_{i,j}$ for reads R_i and R_j defined as

$$r_{i,j} = \frac{\max f(d)}{\sum f(d)}$$

where both the max and the sum are taken over all $d \in \{-L - |R_j|, \ldots, |R_i|\}$. We then classify overlapping and non-overlapping reads based on a lower cutoff m and an upper cutoff M on $r_{i,j}$ such that $r_{i,j} < m$ implies the reads are not overlapping, $r_{i,j} > M$ implies the reads are overlapping, and otherwise the test is non-conclusive.

[8]The most common exception is when two reads are only weakly overlapping which is neither surprising nor problematic.

Shift analysis in the condensed alphabet

The "peakedness" of shift distributions is not always sharp, the main contributing factor to which is the high indel rates of SMRT sequencing. Therefore, we expect that if seeds are found in the condensed alphabet the shift analysis approach cannot be of much use since the coordinates of seeds has even less of a bearing on the overlap offset of the true alignment. For this reason, it is preferable to perform seed indexing in the original alphabet and seed extension in the condensed alphabet. The caveats of condensing seeds and ways to deal with them have already been discussed.

Hybrid algorithm

A natural extension of the above scheme is to use seed extension for those read pairs for which the shift analysis tests is non-conclusive. We can additionally use the information obtained in the first step to guide the seed extension phase by selecting those seeds that are most likely, in the shift-peak sense, to belong to a correct alignment. Therefore, the hybrid algorithm can be described as follows:

i. Find the peak ratio $r_{i,j}$ and conclude if $r_{i,j} > M$ or if $r_{i,j} < m$.
ii. If the ratio is neither small or large enough, the seeds within radius L of the mode shift are considered for extension.

Implementation and Results

The dataset used to test the algorithms is PacBio generated reads for chromosome 1 of Leishmania Donovani where Correct position of reads are inferred using BLAST against a known reference genome.

Source Code

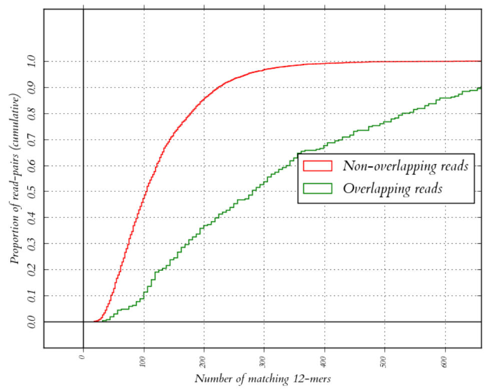
All code is open source and available at github.com/amirkdv/align.py and documentation as well as library API can be found at alignpy.readthedocs.org. The dynamic programming algorithm for sequence alignment as well as the seed extension algorithm are implemented in C. All core k-mer handling logic (indexing, disk IO, etc.) are delegated to SQLite which is a fast, serverless, SQL database implemented in C. All graph handling (cycle breaking, topological sorting of DAG's, and drawing graphs) are delegated to igraph which is implemented in C/C++.

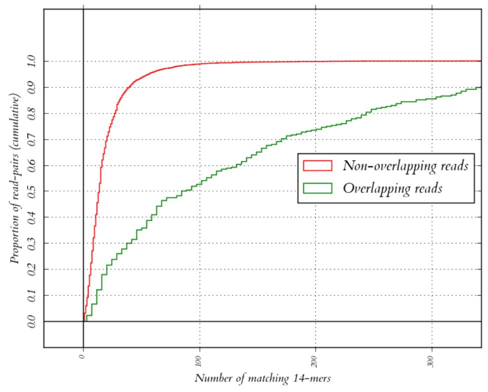
Everything else is implemented in Python interfacing the C component via a foreign-function interface and interfacing SQLite and igraph via existing open source python modules.

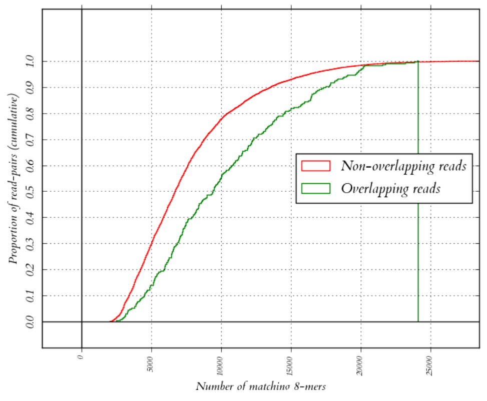
Parameters

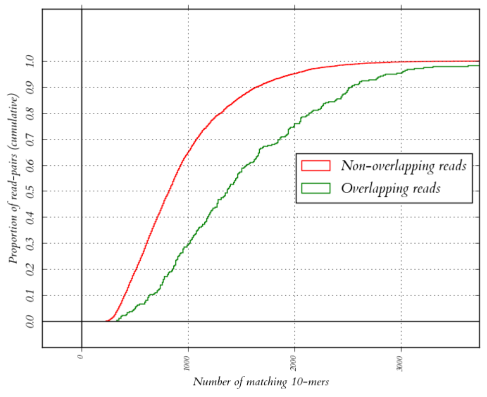
In all results the following parameters have been empirically found and fixed:

- Substitution probabilities are 0.02 for all pairs of distinct nucleotides.
- Gap open and extend probabilities are 0.15 and 0.2, respectively.
- The homopolymeric gap score is -0.2.
- $m = 1000$, minimum required score for overlap alignments as well as the safety margin to guard against mostly-overlapping reads,
- $h = 5$, the h.p. max. length,
- $w = 50$, window size for seed extension
- $D_w = -20$, the drop threshold for seed extension,
- $M_w = 3$, max. successive drops for seed extension,
- $L = 100$, the rolling sum window size for shift distribution.



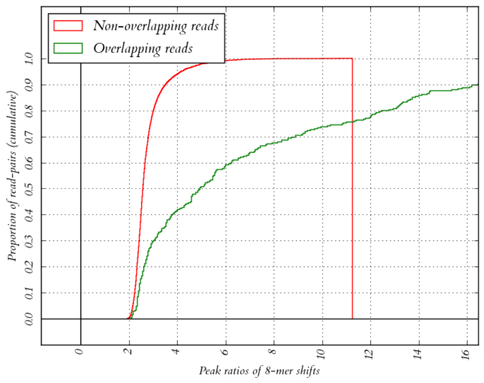


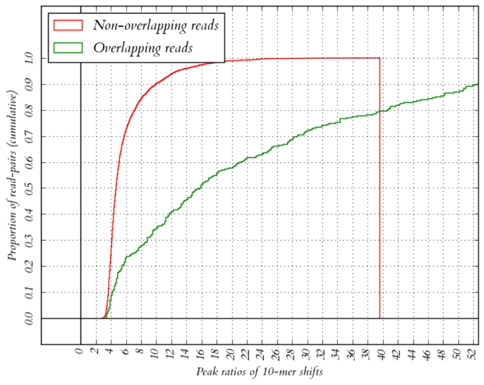


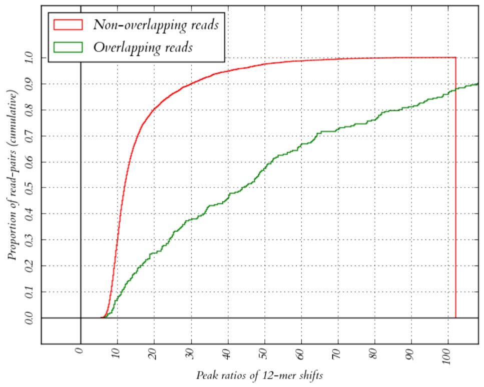


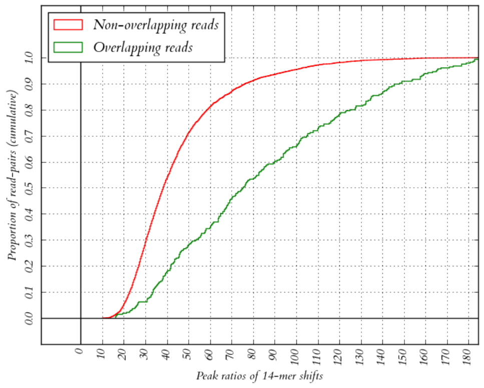
We now consider the behavior of k, the word length of tuples, and the minimum number of seeds m_z required for a pair of reads to be considered potentially homologous. The first set of diagrams show the distribution of number of seeds per read-pair for $k \in \{8, 10, 12, 14, 16\}$. We can see that for $k \leq 12$ there is no meaningful choice of m_z that does not immediately cause a baseline of considerable false negative. This is not surprising since only when using large enough word lengths can we discard some read pairs simply because they have too few seeds. For smaller word lengths *all* read pairs have a considerable amount of seeds in common.

The next set of diagrams show the distribution of $r_{i,j}$ per read-pair for $k \in \{8, 10, 12, 14, 16\}$. We can see that for $k \geq 12$ there is no meaningful choice of m and M, the lower and upper cutoffs, that can significantly reduce the amount of computation. This is not surprising since as we increase k the number of seeds per read-pair decreases and the statistical behavior of shifts is more and more dominated by the baseline noise to the point that at $k = 16$ our simple peak ratio scheme is unable to distinguish overlapping and non-overlapping reads.









Results

Table 1 contains preliminary results obtained from finding the overlap graph of the first 300 reads using various schemes. First, merely using the h.p.-condensed version of sequences in seed extension significantly decreases the false negative rate. Second, Using shift distribution analysis dramatically reduces the computation time. Third, Shift distribution analysis is not enough to produce usable results for later stages and the hybrid algorithm should be considered seriously.

Future work

Faster tuple handling Here are typical amounts of time spent on each of the 3 main parts of the algorithm (all numbers are per read pair): **1)** fetching seeds: 1ms, **2)** analyzing shift distribution: 1ms, **3)** seed extension: 100ms.

As long as seed extension is involved for a pair of sequences the dominant factor in the running time is CPU time spent on alignment. But when only considering shift distributions the disk IO time (spent on fetching seeds) is significant.

Seed extension in linear space and time A potential improvement to seed extension is to drop the 3 non-conventional tuning parameters w, D_w, and M_w and use the following scheme: **1.** Linear space optimization of the sequence alignment is trivial to implement as long as we don't wish to traceback the optimal alignment (which is the case in seed extension.[9]). This will allow us to drop the rolling window as the entire forward and backward extension problems can be solved at once. **2.** To enforce a certain quality of scores banded alignment (which is already implemented) can be used. This implies linear time complexity in read lengths.

Measuring peakedness in shift distributions **1.** *Peak detection* is a well-known problem in time series analysis and such algorithms, for example wavelet-based methods, have been successfully applied in other areas of bioinformatics (Du, Kibbe, and Lin 2006). **2.** One can, in principle, calculate a p-value for each peak by a combinatorial analysis. The problem in this case reduces to calculating variations of the probability distribution of *success runs* in a sequence of independent Bernoulli trials (Feller 1971). This

[9]We do, however, need to know the starting position of the alignment but this information can be passed down the chain of choices to avoid a recursive traceback

Table 1: Overlap discovery results

Algorithm	k	f.n.	f.p.	CPU time	reads
seed extension in original alphabet	15	72%	1.2%	25,000 s	300
seed extension in condensed alphabet	15	37%	0.1%	25,000 s	300
shift distribution analysis	10	54%	27%	1,600 s	300

analysis yield complicated and numerically unstable formulae and good approximations are needed. A naive algorithm which approximates the distribution of success runs by a Binomial distribution and that by a Normal distribution in the limit was tested but the approximation was too weak to give significant p-values for any peak.

Alignment in condensed alphabet As mentioned above, the problem is not yet mathematically well-defined. A possible complicated formulation is to allow single letters to be matched to multiple letters in an alignment. This requires allowing nonstandard choices in the DP table but there is a way of maintaining polynomial time complexity (at most cubic) in the DP algorithm. Much of the complexity is due to the higher order Markov interaction between positions of sequences which, however, is already dealt with to implement the affine gap penalty scheme in quadratic time.

References

Au, Kin Fai et al. (2012). “Improving PacBio Long Read Accuracy by Short Read Alignment”. In: *PLoS ONE* 7.10, e46679. DOI: 10.1371/journal.pone.0046679.

Berlin, Konstantin et al. (2015). “Assembling large genomes with single-molecule sequencing and locality-sensitive hashing”. In: *Nat Biotech* 33.6. Research, pp. 623–630.

Chaisson, Mark, Pavel Pevzner, and Haixu Tang (2004). “Fragment assembly with short reads”. In: *Bioinformatics* 20.13, pp. 2067–2074. DOI: 10 . 1093 / bioinformatics / bth205.

Chin, Chen-Shan et al. (2013). “Nonhybrid, finished microbial genome assemblies from long-read SMRT sequencing data”. In: *Nat Meth* 10.6. Article, pp. 563–569.

Du, Pan, Warren A. Kibbe, and Simon M. Lin (2006). “Improved peak detection in mass spectrum by incorporating continuous wavelet transform-based pattern matching”. In: *Bioinformatics* 22.17, pp. 2059–2065. DOI: 10 . 1093 / bioinformatics/btl355.

Eades, Peter, Xuemin Lin, and W.F. Smyth (1993). “A fast and effective heuristic for the feedback arc set problem”. In: *Information Processing Letters* 47.6, pp. 319 –323. DOI: http://dx.doi.org/10.1016/0020-0190(93)90079-O.

Feller, W. (1971). *An introduction to probability theory and its applications*. Wiley series in probability and mathematical statistics: Probability and mathematical statistics. Wiley.

Festa, Paola, PanosM. Pardalos, and MauricioG.C. Resende (1999). “Feedback Set Problems”. English. In: *Handbook of Combinatorial Optimization*. Ed. by Ding-Zhu Du and PanosM. Pardalos. Springer US, pp. 209–258. DOI: 10 . 1007/978-1-4757-3023-4_4.

Koren, Sergey et al. (2012). “Hybrid error correction and de novo assembly of single-molecule sequencing reads”. In: *Nat Biotech* 30.7, pp. 693–700. DOI: 10.1038/nbt.2280.

Pavel A. Pevzner Haixu Tang, Michael S. Waterman (2001). “An Eulerian Path Approach to DNA Fragment Assembly”. In: *Proceedings of the National Academy of Sciences of the United States of America* 98.17, pp. 9748–9753.