Consensus Module

Karin A. Remington

1. Overview

The Consensus Module (CNS) generates a consensus sequence for each contig in the layout by first multi-aligning the read fragments and surrogates contained within the contig and then assigning a base to each position based on some weighted scoring of the appropriate column in the multi-alignment. The initial prototype will perform naïve multi-alignment and consensus scoring, while subsequent implementations will provide an refinement process to improve the multi-alignment and more sophisticated consensus scoring based on the quality values of the reads.

2. Memory Usage

Consider 500Mbp to be the length of the longest possible contig (based on estimate of longest chromosome). At 10x coverage, we expect 5 Gbp of read data to process. To include both base and quality-value data then requires at least 10Gb of memory.

3. Interface

In addition to the conventional call of the Consensus Module at the end of the Assembly pipeline, Consensus may also be called in some fashion from other stages of the pipeline (for example, as a function call within the Chunk Graph Builder, and/or as a separate process between the Chunk Graph Builder and the Chunk Graph Walker). The output of Consensus must vary according to the input available to it at these various stages of the assembly process. We describe here both a functional interface and a command line interface for the Consensus module. The functional interface requires only a list of fragments, their layout positions, and a pointer to a valid fragStore for retrieving sequence information. The command line version can be called following a) the CGB stage, where the input stream contains Chunk messages and the output is Unitig messages, and b) the CGW stage, where the input stream contains Surrogate and IntContig messages, and the output is Unitig and Contig messages. In scenario b), the input message are subject to the ordering requirement that any Surrogate message referenced in a given IntContig message appears in the stream prior to its reference. For each Surrogate unitig, Consensus expects a Surrogate message specifying the unitig's internal ID along with a list of LayoutPos for the fragments contained in the surrogate. For each contig in the assembled layout, the CGW module is expected to emit an IntContig message that specifies the contig identifier, contig length, a count of reads, guides and surrogate unitigs contained within the contig layout, and lists layout position messages, one for each such item. The layout position information for each item is encoded as LayoutPos (see below). The scalar `label` in the LayoutPos is set to `NOKEEP` in the case of resolved fragments contained in a Surrogate to avoid duplication, and these fragments are not itemized in the output Unitig message; otherwise, `label` is set to `KEEP`. In addition, each IntContig message contains a count of good and bad links, the number of unitigs contained and a list of their positions, and the number and list of predecessors and successors in the scaffold graph as specified by ScaffLink_IDS which refer to ScaffLink messages. All messages not specifically referenced here will be passed through CNS unchanged.

3.1. Functional Interface

```
char * getConsensus_AS(int num_reads, LayoutPos *reads, FragStoreHandle frag_store);
```

The input arguments are `num_reads`, giving the number of reads/guides to multialign, a list of `LayoutPos` for the reads/guides and a `FragStoreHandle` pointing to the persistent fragment store generated by some previous stage of the Assembly system.

3.2. Command Line Interface

Command line interface for AS_CNS module:

```
consensus [-q] [-[ra]] -P FragStorePath CGWStream
```

`-q`	Levels 1 and 2: requests that quality values be used in breaking ties when determining consensus base calls. Level 3: requests that quality values be used to determine consensus base calls and in scoring re-alignments during the refinement
`-r`	Levels 2 and 3: requests that alignments be refined via pass through a round-robin realignment process.
`-a`	Levels 2 and 3: requests that alignments be refined via pass through a window based (abacus) realignment process.

The `FragStorePath` directs the module to the location of the persistent fragment store generated by previous stages of the Assembly system. The `CGWStream` consists of messages of type `SurrogateMesg` and `IntContigMesg` from the Assembly module (as described in the CNS I/O specification document), with the ordering requirement that any `SurrogateMesg` referenced in a given `IntContigMesg` appears in the stream prior to its reference. For each Surrogate unitig, Consensus expects a Surrogate message specifying the unitig's internal ID along with a list of LayoutPos for the fragments contained in the surrogate. For each contig in the assembled layout, the Assembly module is expected to emit a IntContig message that specifies the contig identifier, contig length, and lists of reads/guides and surrogates contained within the contig layout. The reads, guides and surrogates are encoded as LayoutPos (see ProtoSpec document for details). The boolean `label` in the LayoutPos is set to `NOKEEP` in the case of resolved fragments in the Surrogate, and these fragments are not itemized in the output Chunk message.

4. Design

CNS processes input on a contig by contig basis. Messages are read from the CGWstream, which may be an ASCII or binary file or Unix stream, and processed according to type. Messages other than IntContig and Surrogate are passed through unmodified. Surrogate and IntContig messages are processed as follows.

Surrogate messages initiate a multi-alignment subprocess in which a consensus sequence (with quality values) is generated to serve effectively as a read fragment in the multi-alignment of the contig(s) containing the surrogate unitig.

IntContig messages initiate the main multi-alignment process. The process requires that any surrogates referenced in the IntContig message have been previously processed by CNS and resulting consensus sequences have been stored for reference. The combined set of read/guide sequences and surrogate consensus sequences is then aligned using a pair-wise induced multi-alignment (see 4.1). Once this initial multi-alignment is achieved, a first approximation consensus sequence is generated. This will be the prototype output. After a functioning prototype is in place, subsequent efforts will be directed toward a) refining the initial multi-alignment by the round-robin heuristic (see 4.2), and b) improving the consensus scoring scheme by weighting with quality values (see 4.3). The remainder of this section describes the three "levels" of development of the CNS module.

4.1. Initial Multi-Alignment/Consensus

The stages of the Assembly process prior to CNS provide an initial "layout" of the read fragments within a given contig/surrogate. Using this information, encoded as `LayoutPos` messages, CNS first orders the read fragments by starting position within the contig. Once ordered, the reads are assembled into a multi-alignment structure in a pair-wise fashion, using `AS_ALN_DP_Compare()` to calculate the pair-wise alignments. Each read is aligned to the previously aligned read with maximal overlap length based on the `LayoutPos` information. After all of reads in a given contig/surrogate have been included in the multi-alignment, a consensus sequence is built by calling bases in each column of the multi-alignment. Ties can be broken using the sum of quality-value scores (with the `-q` command line option) or purely randomly. Quality values are generated for each called based of the consensus sequence by averaging the quality values of the matching bases of the reads/guides/surrogates covering the given base, though these should be treated only as placeholders for an eventual quality value assignments based on error probability interpretations of the sequence quality values.

4.2. Round-Robin Refinement of Multi-Alignment

The initial pair-wise induced multi-alignment is potentially only a crude approximation to an optimal multi-alignment. To improve this approximation, the Level 2 version of CNS employs a round-robin realignment algorithm [Anson and Myers, 1997]. This algorithm preserves the global structure of the initial alignment, but improves the local alignment locally.

The refinement proceeds by removing each fragment read in turn from the multi-alignment and realigning it to the consensus of the remaining multi-aligned reads, repeating the process until the alignment score does not decrease. The refinement scoring function, which scores the alignment of a single character *x* with a column of characters *X*, is motivated and detailed in [Anson and Myers, 1997]. It is an evenly weighted linear combination of a consensus score and a normalized or fractional mismatch score, given by: [Note: careful symbol definition not yet included.]

$$\delta_{a+c}(x,X) = \frac{1}{2}\delta_c(x,X) + \frac{1}{2}\delta_a(x,X)$$

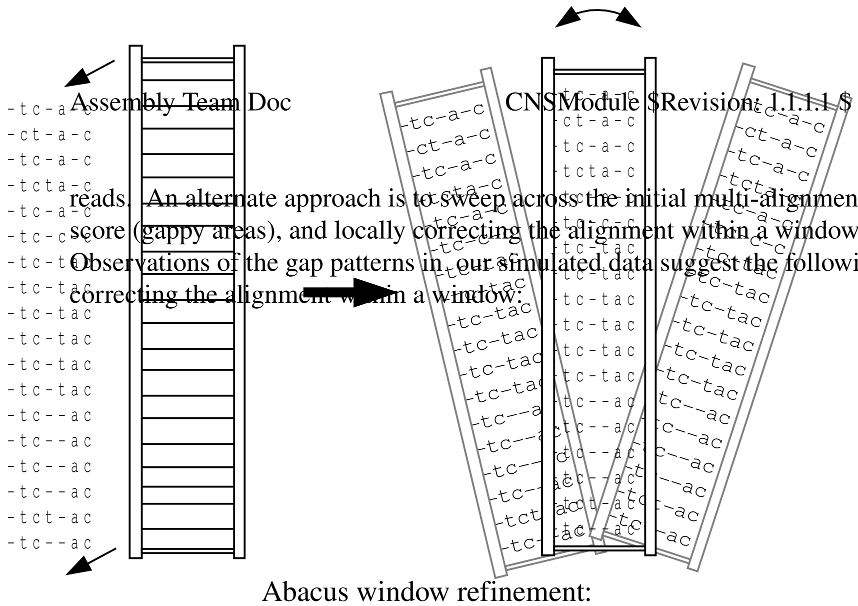
where the scoring functions δ_c and δ_a are defined as follows:

$$\delta_c(x,X) = \begin{cases} 0 & \text{if } x \in \bar{c}(X) \text{ or } X = \varepsilon \\ 1 & \text{otherwise} \end{cases}$$

$$\delta_a(x,X) = \delta_a(x,(a_1,a_2,\ldots,a_n)) = \begin{cases} \dfrac{\left|\{a_i : a_i \neq x\}\right|}{\text{n}} & \text{if } n > 0 \\ 0 & \text{if } n = 0 \end{cases}$$

4.3. Abacus Window Sweep Refinement of Multi-Alignment

The round-robin procedure described in 4.2 can be time consuming, particularly for long contigs with many reads. ~~An alternate approach is to sweep across the initial multi-alignment, looking for regions of high score (gappy areas), and locally correcting the alignment within a window containing such a region. Observations of the gap patterns in our simulated data suggest the following “abacus” heuristic for correcting the alignment within a window:~~



Abacus window refinement:

1) Introduce gap column
2) Sweep through original columns from left to right
3) Left shift if match or lower score available
4) Repeat from right to left
5) If right sweep score is lower, keep
   else take left sweep result

4.4. Incorporating Quality Values in Multi-Alignment and Consensus

In the first two release stages, the CNS multi-alignment is performed based on the read sequences only, not on their associated quality values~~. The quality values could be used as a simple weighting in the determination of the consensus base of each column in the alignment, say by choosing the base with the highest quality value sum or by breaking simple consensus vote ties with quality values. In addition (or instead ?), the quality values could be used to locally refine the alignment, with more weight being given to data with higher associated confidence levels. This strategy is envisioned for~~ The Level 3 version of CNSwill employ an appropriate scoring of the alignments with respect to quality values associated with each column of the multi-alignment. Still to be addressed is how to deal with data which has no associated quality values on input. Assembly’s current data compression scheme prohibits negative quality values and strictly limits the upper bound on the consensus quality values, which should be taken into consideration on output, though computations need not be similarly restricted.

Based on initial assembly design documents, we will assume that quality values are integer values in the interval [0,60], though Phred and Phrap documentation indicates a Phred range of [0,40] with the additional special values of 98 and 99**. Before a proposal is implemented, we will need to clarify the specification of the quality value input to Assembly**. We further assume that the quality values have an interpretation similar to Phred values q_i, i.e. $q_i = -10\log_{10}(p_i)$, where p_i represents the probability of an error in the base call at position i.

~~To achieve the desired affect, we are looking for a scheme which respects the following:~~

~~1. relatively high positive score for a mismatch of high quality data~~

~~2. relatively low score for mismatches of low quality data~~

3. ~~smallest score for a match of high quality data~~
4. ~~more weight given to column consensus than to single base~~

~~For comparison, in Phrap, Phred quality values are "adjusted" based on mutually confirming read pairs (see Appendix for documentation). At a given alignment position, if a given read is confirmed by an "independent" read (derived from a different chemistry or from an opposite strand), the quality value of that base call is adjusted to the sum of the quality values of the confirming pair of base calls. The quality value of the consensus is then calculated as the maximum of the adjusted quality values offset by the quality values of any mismatches in that column. The following proposal does not require confirming reads to be "independent" in the Phrap sense. We consider all to be independent, and so the quality values are summed across all agreeing reads, not just the maximum pair.~~

QV Scoring Strawman Proposal:

~~Consider the following scoring scheme:~~

$$\delta^q(x, X) = \delta^q(x, (a_1, a_2, \ldots, a_n)) = \sum_{a_i \neq x} q_i - \sum_{a_i = x} q_i - q_x$$

~~Under δ^q, the quality of the base being aligned is used to offset the penalty imposed by the accumulated qualities of the column. The column gap penalty is the sum of the quality values of the candidate column, and the base gap penalty is the quality value of the candidate base. This seems to have the qualities desired. After alignment, regions of low quality appearing within higher quality data may indicate a possible site for discriminating repeat instances or SNPs.~~

Consider for the sake of discussion the case where a single pair of haplotypes is being assembled. Then, for each column of the multi-alignment, the potential underlying events can be enumerated :

$$= E_a \quad E_\rho = \{ \quad : \quad \in A\} \quad \{\rho = ab : a, b \in A, a \neq b\}$$

Now, for each potential underlying event e in E, calculate the conditional probability of observing the column data given the underlying event.

$$\tau_e = \begin{cases} \prod_{i=1}^{n} ((b_i == e)\,?(1 - x_i) : \frac{1}{4} x_i) & \text{for } e \in E_a \\ \tau_a + \tau_b & \text{for } e \in E_\rho \end{cases}$$

Then, find the underlying event with greatest probability, using the formula from Churchill and Waterman:

$$\pi(e) = \frac{p(e)\tau_e}{\sum_{x \in E} p(x)\tau_x}$$

AUTHORS

Karin A. Remington

Created January 29, 1999

Modified March 11, 1999

Modified April 8, 1999

Sections on Round-Robin and QV realignment substantially altered.

Modified May 14, 1999

Section on Abacus refinement added, and QV scoring method is outlined.

Appendix:

For convenience, the following is quoted from Phrap documentation available at URL
http://bozeman.mbt.washington.edu/phrap.docs/phrap.html

Phrap adjusted quality values/error probabilities: Phrap computes adjusted quality values for each read on the basis of read-read confirmation information, as follows. If a read is confirmed by an opposite-strand or different-chemistry (dye terminator vs. dye primer) read at a given position, that position is given a quality which is the sum of the two input read qualities; when more than one opposite-strand read confirms a given position, only the single highest quality from all opposite-strand matching reads is used. If qualities are related to error probabilities as described above, this procedure can be interpreted as computing an error probability for each base which is the product of the error probabilities for the two reads; since error profiles for opposite-strand or different chemistry reads are essentially independent, this is reasonable, although it is somewhat conservative in that it does not take into account same-strand matches (which clearly should count for something, although they certainly are not independent).

Contig positions are assigned quality values equal to the highest adjusted quality of any read at that position, and then adjusted downward to take into account any discrepancies with other reads. As a result, when phred input qualities are used, the output phrap qualities associated to each contig position have a natural interpretation as (conservative) error probabilities. Such error probabilities provide an extremely useful guide to where editing or additional data collection is needed.