BIOINFORMATICS ORIGINAL PAPER Vol. 25 no. 9 2009, pages 1118–1124
doi:10.1093/bioinformatics/btp131

Sequence analysis

A consistency-based consensus algorithm for *de novo* and reference-guided sequence assembly of short reads

Tobias Rausch[1,2,*], Sergey Koren[3], Gennady Denisov[3], David Weese[2], Anne-Katrin Emde[1,2], Andreas Döring[2] and Knut Reinert[2]

[1]International Max Planck Research School for Computational Biology and Scientific Computing, Ihnestr. 63 - 73, [2]Algorithmische Bioinformatik, Institut für Informatik, Takustr. 9, 14195 Berlin, Germany and [3]J. Craig Venter Institute, 9704 Medical Center Drive, Rockville, MD 20850, USA

Received on November 3, 2008; revised on January 23, 2009; accepted on March 2, 2009

Advance Access publication March 5, 2009

Associate Editor: Limsoon Wong

ABSTRACT

Motivation: Novel high-throughput sequencing technologies pose new algorithmic challenges in handling massive amounts of short-read, high-coverage data. A robust and versatile consensus tool is of particular interest for such data since a sound multi-read alignment is a prerequisite for variation analyses, accurate genome assemblies and insert sequencing.

Results: A multi-read alignment algorithm for *de novo* or reference-guided genome assembly is presented. The program identifies segments shared by multiple reads and then aligns these segments using a consistency-enhanced alignment graph. On real *de novo* sequencing data obtained from the newly established NCBI Short Read Archive, the program performs similarly in quality to other comparable programs. On more challenging simulated datasets for insert sequencing and variation analyses, our program outperforms the other tools.

Availability: The consensus program can be downloaded from http://www.seqan.de/projects/consensus.html. It can be used stand-alone or in conjunction with the Celera Assembler. Both application scenarios as well as the usage of the tool are described in the documentation.

Contact: rausch@inf.fu-berlin.de

1 INTRODUCTION

Gene prediction, genome comparison or phylogenetic studies rely on accurate reference sequences. These sequences are assembled in sequencing projects that require a robust and versatile consensus tool. Most of the present consensus tools are an integral part of fragment assemblers such as the Celera (Myers *et al.*, 2000), Arachne (Batzoglou *et al.*, 2002) or Atlas (Havlak *et al.*, 2004) assembler. These assemblers follow the established three-phase assembly methodology: Overlap-Phase, Layout-Phase and Consensus-Phase. In this scenario, the layout module forwards for each contig a set of reads with their approximate positions to the consensus module. The consensus module then computes a multi-read alignment and a final consensus sequence. A multi-read alignment is, however, also required in other applications, including reference-guided genome assembly projects (Pop *et al.*, 2004) and variation analyses (Denisov *et al.*, 2008). A reference-guided genome assembly uses an already sequenced reference genome to assemble a new genome. The new genome might have insertions or deletions with respect to the reference genome. These insertions or deletions can be detected using mate pair information as shown in Figure 1. The mate pairs themselves can be used to infer the layout of the reads within an insertion. The paired-end libraries, however, exhibit high deviations from the mean mate pair library length and hence determining the correct layout for such an insert sequence is difficult. Furthermore, a consensus algorithm together with mate pair information can be used to bridge the gaps between a scaffold's contigs or to compute the consensus sequence of a resolved repetitive region. To support both, reference-guided and *de novo* genome assemblies, our proposed algorithm works in two running modes. The first one allows insert sequencing and the second one, as a component of the Celera Assembler, *de novo* assembly.

Ad hoc techniques that incorporate reads one-by-one into a growing multiple alignment are highly error-prone and inadequate for the data produced by the new sequencing platforms such as 454 Life Sciences (www.454.com), Illumina's Solexa sequencing technology (www.illumina.com) and Applied Biosystems SOLiD Sequencing (www.appliedbiosystems.com). In contrast to Sanger reads, next generation sequencing platforms have higher error rates and produce high volumes of short read, deep coverage data. Therefore, new consensus methods are greatly needed. Especially for many downstream analyses such as repeat resolution or variation analysis accurate multi-read alignments showing single nucleotide polymorphisms (SNPs), indels and sequencing errors are indispensable. Unfortunately, multiple sequence alignment programs such as Clustal W (Thompson *et al.*, 1994), T-Coffee (Notredame *et al.*, 2000), MUSCLE (Edgar, 2004) or MAFFT (Katoh *et al.*, 2002) cannot be directly used to compute a multi-read alignment because they all assume globally related sequences or at least a shared local region. This is, of course, inappropriate for the alignment of reads where only about c reads overlap a given nucleotide with c being the assembly coverage. However, established multiple sequence alignment techniques such as consistency (Notredame *et al.*, 2000) and progressive alignment (Feng and Doolittle, 1987) can still be used and are the basis of the proposed method.

*To whom correspondence should be addressed.

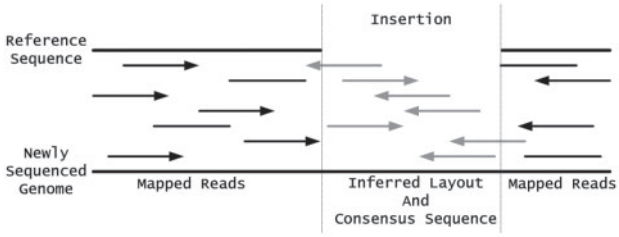

Fig. 1. A newly sequenced genome with an unknown insertion with respect to a reference genome. The mapped reads (black lines) can be used to infer the layout of the mate pairs (grey lines). Mate pairs are indicated by arrows pointing to each other. From this inferred layout, the consensus is computed.

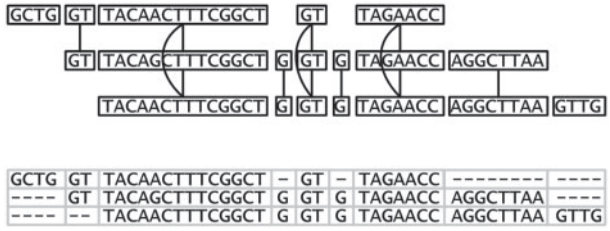

Fig. 2. An alignment graph and the corresponding alignment matrix for three reads. Vertices represent non-overlapping sequence segments, edges represent un-gapped aligned sequence segments and gaps are implicitly represented by the topology of the graph.

2 METHODS

2.1 An alignment graph of sequence segments

Throughout the consensus construction, an alignment graph is used to represent a multi-read alignment as shown in Figure 2. Vertices represent non-overlapping sequence segments, edges connect vertices and represent ungapped aligned sequence segments and gaps are implicitly represented by the topology of the graph. For example, the *GCTG* vertex in Figure 2 has no outgoing edges (degree zero) and thus, it is aligned to gaps in all other sequences. The properties of an alignment graph G are given below.

- For a set $\mathcal{S}=\{S^0,S^1,...,S^{n-1}\}$ of n reads, the alignment graph $G=(V=\{V^0\cup V^1\cup\cdots\cup V^{n-1}\},E)$ is an n-partite graph.
- Each vertex $v_p^i\in V^i$ represents a sequence segment in S^i of arbitrary length. We also say that v_p^i covers all positions of the segment. For instance, v_p^i might cover the sequence segment $S^i_{u1,u2}=s^i_{u1}s^i_{u1+1}\cdots s^i_{u2-1}$.
- Every position in $S^i=s^i_0s^i_1\cdots{}^i_{|S^i|-1}$ is covered by one and only one vertex $v_p^i\in V^i$.
- Three integers are associated with each vertex: (i) the sequence identifier it belongs to; (ii) the beginning of the segment; and (iii) the length of the segment.
- An edge $e=\{v_p^i,v_q^j\}\in E$ with $i\neq j$ indicates that vertex v_p^i can be aligned with vertex v_q^j. In other words, the sequence substring in S^i covered by v_p^i can be aligned without gaps to the substring in S^j covered by v_q^j.
- The benefit of aligning v_p^i with v_q^j is given by an edge-weight w_e.

The alignment graph is a compact and versatile description of an alignment. Large-scale alignments can be efficiently stored because long segments are represented by only a single vertex. Furthermore, the extension and direction of an alignment is completely defined by alignment edges. That is, the graph formulation is equally suitable to align globally related sequences or thousands of reads where only subsets are related by mutual overlaps (Fig. 3). The algorithm to convert an alignment graph G into an ordinary alignment matrix is given below.

(1) Compute the connected components C_i of G. If G is an actual alignment, each $C_i\in\mathcal{C}=\{C_0,C_1,...,C_{k-1}\}$ has at most one vertex from every sequence.

(2) A directed component graph G_C is constructed with $G_C=(V_C=\{v_{C_0},v_{C_1},...,v_{C_{k-1}}\},E_C)$. A directed edge $e=(v_{C_u},v_{C_v})\in E_C$ with $u\neq v$ is inserted into the graph if and only if there is a vertex in component C_u that precedes a vertex in component C_v in one of the sequences. Informally, the edges in G_C mirror the order of the vertices along a sequence.

(3) A topological sort is computed on the vertices of G_C. Note that this operation does not impose an order on adjacent indels and thus, there is a many-to-one relationship between alignment matrices and alignment graphs.

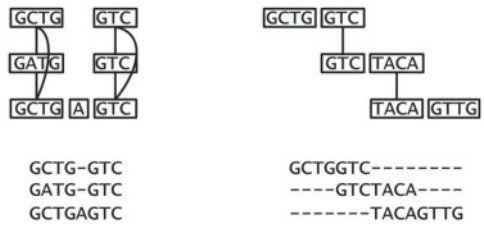

Fig. 3. The alignment on the left shows globally related sequences whereas the one on the right shows a simplified multi-read alignment. Note that the direction of the alignment is solely dependent on the edges.

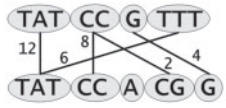

Fig. 4. A general alignment graph of two sequences with weighted match information. Only a subset of the edges can be realized in an alignment.

Besides representing actual alignments, the graph can also be used to store arbitrary match information as illustrated in Figure 4. The set of edges E is now solely a set of possible alignment edges and only a proper subset $E'\subset E$ constitutes a valid alignment. This subset E' is called a trace (Kececioglu, 1993; Sankoff and Kruskal, 1983). The best alignment is the set of edges of maximum weight (maximal trace). In the multiple case, this problem is known to be NP-hard (Wang and Jiang, 1994) and the optimal dynamic programming algorithm has exponential complexity in the number of sequences. In the Section 2.2, we present an efficient and accurate heuristic to compute such a trace for an alignment graph of thousands of reads.

2.2 Multi-read alignment algorithm

The multi-read alignment algorithm has four processing steps. step 1: computation of pairwise overlap alignments; step 2: alignment graph construction; step 3: consistency extension; and step 4: a graph-based progressive alignment.

Step 1. The input of the algorithm is a set of reads in FASTA format with their estimated begin and end positions. These estimated layout positions are either forwarded from the layout module of a *de novo* assembler or inferred from mate pair information (Fig. 1). In the first case, the positions are estimated during the layout computation of the assembler and thus, these positions tend to be very accurate. In the second case, the layout positions are

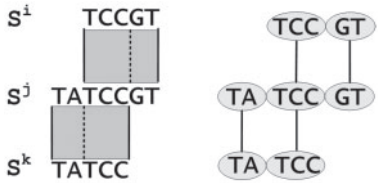

Fig. 5. The segment match refinement algorithm cuts recursively all overlapping segments until all segments are disjoint. In the above example, the two overlapping segment matches are refined into four distinct matches.

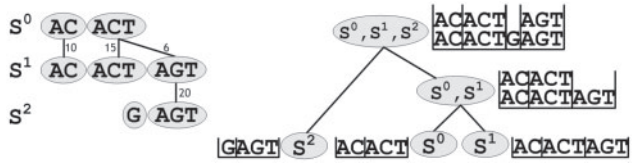

Fig. 6. The progressive alignment proceeds along a guide tree shown on the right. On the left is the original alignment graph with edge-weights indicating the quality of a match. Next to each internal guide, tree node a profile of the already aligned subtree is shown.

derived from a paired-end library and hence, they deviate strongly from the true positions depending on the standard deviation of the paired-end library. Because of these differences, our proposed algorithm works in two running modes.

The first running mode assumes accurate start and end positions. These positions are used to detect potential overlaps and to estimate the alignment diagonal of a banded overlap alignment with affine gap costs (Gotoh, 1982). A banded alignment initializes the dynamic programming matrix with zeros and computes only a band of size k around the estimated alignment diagonal where k is the bandwidth. Note that the bandwidth k depends on two key characteristics—the sequencing error rate and the length of the overlap. Hence, the baseline for k is a configurable parameter that is adjusted linearly based on the length of the overlap. Usually, it is unnecessary to compute all possible pairwise overlaps, especially for deep coverage sequencing projects. For that reason, we provide a parameter that adjusts how many overlaps are computed per given read. The more error-prone the reads are, the more overlaps one should compute per read. The second running mode assumes layout positions estimated from a paired-end library. In this case, the positions tend to be unreliable and hence, all overlaps of reads in a user-defined window are computed with a standard dynamic programming algorithm.

Subsequent to the computation of overlap alignments, we select all overlaps of significant quality and length. Similar to the bandwidth, both parameters are adaptable from the command line. The selected overlaps are then subdivided into ungapped alignments, called segment matches. A segment match $M_{ij}=(S^i_{u1,u2},S^j_{w1,w2})$ represents an ungapped alignment of $S^i_{u1,u2}=s^i_{u1}s^i_{u1+1}\cdots s^i_{u2-1}$ with $S^j_{w1,w2}=s^j_{w1}s^j_{w1+1}\cdots s^j_{w2-1}$.

Step 2. All collected segment matches $\mathcal{M}$ are used to construct an alignment graph. Unfortunately, segment matches can overlap and intersect each other. This violates the alignment graph property that sequence segments are covered by one and only one vertex. For instance, the set $\mathcal{M}$ might contain two matches M_{ij} and M_{jk} where $M_{ij}=(S^i_{u1,u2},S^j_{x2,x4})$ and $M_{jk}=(S^j_{x1,x3},S^k_{y1,y2})$. Then M_{ij} and M_{jk} intersect each other in sequence S^j if $x1<x2<x3<x4$ as illustrated in Figure 5. To avoid a greedy method that would choose one of the conflicting matches, we implemented a multiple segment match refinement algorithm (Rausch *et al.*, 2008). This algorithm subdivides overlapping segment matches into distinct submatches so that no segment match begins or ends within another segment match as shown in Figure 5. This procedure ensures that we can ultimately divide each sequence into non-overlapping sequence segments so that every segment match begins or ends on the boundary of a sequence segment. We then define vertices for each sequence segment and we connect two vertices v_1 and v_2 with an edge $e=\{v_1,v_2\}$ if and only if there is a segment match between the corresponding sequence segments. The weight of this edge depends on the quality of the segment match.

Step 3. The initial alignment graph was constructed using solely *pairwise* sequence alignment information. We now extend the graph to incorporate multiple alignment information using the triplet extension from T-Coffee (Notredame *et al.*, 2000). The triplet extension is one example of a consistency method (Gotoh, 1990) and relies on the following observation—the two segment matches $M_{ij}=(S^i_{u1,u2},S^j_{x1,x2})$ and $M_{jk}=(S^j_{x1,x2},S^k_{y1,y2})$ induce a putative transitive segment match $M_{ik}=(S^i_{u1,u2},S^k_{y1,y2})$ that is either consistent or inconsistent with a segment match from the precomputed pairwise alignment of S^i and S^k. If it is consistent, greater confidence in this segment match is established. In terms of the alignment graph, we thus traverse all pairs of alignment edges $\{e_{ij}=\{v^i_p,v^j_q\},e_{jk}=\{v^j_q,v^k_r\}\}$ that share a common vertex. If the transitive edge $e_{ik}=\{v^i_p,v^k_r\}$ is present we found a three-way clique and add to $w_{e_{ik}}$ the minimum weight of the other two edges $\min(w_{e_{ij}},w_{e_{jk}})$. If the transitive edge is not present, we create it with the minimum weight of the other two edges.

Step 4. In the last step, this consistency-enhanced alignment graph is used to progressively align the reads according to a guide tree (Fig. 6). This guide tree is constructed from the overlap alignment scores using a sparse distance matrix and the fast UPGMA algorithm (Sokal and Michener, 1958). A sparse distance matrix is used instead of an ordinary matrix because for each read only c other reads are expected to overlap where c is the assembly coverage. The guide tree ensures that the best quality overlaps are aligned first whereas the difficult and error-prone overlaps caused by reads with many sequencing errors come in late, when partial alignments along subtrees are already quite large and fixed. The progressive alignment itself is completely independent of the nature of the sequences. Given an input alignment graph, it builds a multiple alignment along the guide tree simply by aligning strings of vertices. In the pairwise case, we can directly apply the heaviest common subsequence algorithm (Jacobson and Vo, 1992) to compute the maximal trace. If we progress up in the guide tree, we have profiles of vertices. We can always project these profiles onto a string of nodes and estimate the edge weights between the profiles from the original edge weights by taking the average weight. At each internal vertex of the guide tree, we thus compute another heaviest common subsequence of these profiles. The procedure is summarized in Figure 6. On average, the final profile will have about c vertices at each position where c is again the coverage. Thus, the amount of required memory depends on the coverage and the source sequence length. It is, however, largely independent of the number of reads. This is a key distinction of our method to current multiple sequence alignment tools where the profiles grow linearly with the number of sequences. In a final post-processing step, we compute the consensus sequence and convert the final profile into a multi-read alignment. The consensus sequence can be computed using a simple majority vote in each column or using a Bayesian method (Churchill and Waterman, 1992). The final multi-read alignment can be visualized in Hawkeye (Schatz *et al.*, 2007) or it can be written to a simple text file.

3 RESULTS

In the introduction, we proposed the applicability of our tool in three main scenarios: (i) as the consensus tool in a *de novo* assembly; (ii) as a consensus tool in a reference-guided genome assembly for insert sequencing; and (iii) as a prerequisite for variation analyses.

3.1 Consensus generation during *de novo* assembly

On simulated data, the true source sequence is known in advance and consensus errors can be counted and compared among

different tools. We tried to compare our tool to as many as possible present consensus tools. Unfortunately, most assemblers are monolithic and were not designed to run their consensus algorithm as a stand-alone process. Either because of this lack of modularity or because of a lack of documentation, we did not succeed for the Atlas (Havlak *et al.*, 2004), PCAP (Huang *et al.*, 2003) and Phusion assembler (Mullikin and Ning, 2003). However, we could run the consensus module of the Minimus assembler (Sommer *et al.*, 2007) from the AMOS consortium (http://amos.sourceforge.net) and the consensus module of the Celera Assembler (Myers *et al.*, 2000). Reads were simulated under various settings and different error rate assumptions. The settings and results are summarized in Table 1. For the old Sanger-style reads, the difference in the results is small and our tool outperforms the other tools only for high error rates. The AMOS consensus module performs very well on reads of length 200 even for a relatively high coverage. For these kind of reads, our own tool delivers similar results but is not as fast as the AMOS consensus module. This increased running time can be seen in all experiments and is caused by the pairwise banded overlap alignments and the graph-based progressive alignment. However, it clearly pays off in terms of consensus quality, especially in the case of very short reads of length 35.

As mentioned in the Section 1, the consensus program was also integrated in the Celera Assembler as an optional consensus module. To evaluate this integration, a short read assembly was carried out with data from the new NCBI Short Read Archive (SRA, www.ncbi.nlm.nih.gov/Traces/sra/). The 454 sequencing data of the *P. gingivalis W83* strain (Accession: SRA001027) was obtained. It included two mated (Accession: SRR001351) and two non-mated read datasets (Accession: SRR001352) from a 454 FLX half-plate. The assembler was tested on all four datasets E8YURXS01, E8YURXS02 (mated reads), E9T0MN001 and E9T0MN002 (non-mated reads) separately. We also obtained the provisional reference sequence of *P. gingivalis W83* from NCBI (Accession: NC_002950). In Table 2, we compared the assembly statistics of the Celera Assembler using the default consensus with the version of the Celera Assembler using the new consensus module. The results of both versions are very similar over all datasets. The number of contigs slightly differs because both tools are used twice during unitig consensus and scaffold consensus. Aligning the largest contig from each assembly with the provisional reference sequence revealed, however, minor differences. We counted these differences and report the results in Table 2. Unfortunately, the 454 sequencing data was generated from a different organism of the same strain than the original assembly, so it remains unclear if these differences are actual consensus errors, true polymorphisms or errors in the provisional reference sequence. For a few differences, we inspected the multi-read alignment manually. Some of the common differences were in well-assembled, non-repeat areas and hence, they are probably true polymorphisms. Most of the discrepancies between the two consensus methods occurred in low-coverage regions where two nucleotides occur equally often.

3.2 Insert sequencing

In Figure 1, we illustrated the insert sequencing scenario. In contrast to *de novo* assembly, the layout positions in this case are rather imprecise, because they are derived from mate pair information.

Table 1. Consensus generation during *de novo* assembly: results of a simulation study for consensus computation

Setting			Tool	Error rate (0.5%)		Error rate (1%)		Error rate (2%)		Error rate (4%)	
Source length	Read length	Coverage		Errors	Time (s)	Errors	Time (s)	Errors	Time (s)	Errors	Time (s)
50 000	800	10×	AMOS-Cons	0	0.14	0	0.19	2	0.29	8	14.65
			CA-Cons	1	1.62	0	1.70	5	2.01	>20	2.68
			Seq-Cons	0	3.31	1	3.71	1	5.08	1	8.48
50 000	200	20×	AMOS-Cons	0	0.26	0	0.38	0	0.70	0	11.02
			CA-Cons	0	3.71	0	4.36	2	6.00	>20	10.08
			Seq-Cons	0	16.65	0	19.98	0	28.77	1	43.40
50 000	35	30×	AMOS-Cons	0	0.94	1	1.06	0	2.27	10	5.08
			CA-Cons	0	9.35	4	15.01	–	–	–	–
			Seq-Cons	0	173.49	0	185.21	0	203.17	0	230.27
100 000	800	10×	AMOS-Cons	0	0.32	1	0.45	3	0.58	>20	26.57
			CA-Cons	1	3.22	1	3.57	7	4.23	>20	5.60
			Seq-Cons	0	6.60	1	8.11	0	11.10	11	18.95
100 000	200	20×	AMOS-Cons	0	0.61	0	0.86	0	1.54	1	20.20
			CA-Cons	0	7.69	0	10.26	10	14.73	>20	24.45
			Seq-Cons	0	37.87	0	46.42	0	67.40	0	103.81
100 000	35	30×	AMOS-Cons	0	1.86	0	2.35	4	4.46	8	11.21
			CA-Cons	0	28.08	7	48.06	–	–	–	–
			Seq-Cons	0	643.22	0	685.85	0	776.26	1	930.23

Reads were simulated under various settings detailed above. We compared the tools using various error rates and report the number of errors in the consensus sequence with respect to the true source sequence for all positions with coverage > 2. AMOS-Cons is the consensus module of the AMOS library, CA-Cons is the consensus module of the Celera Assembler and Seq-Cons is the proposed consensus tool. '–' values indicate that the consensus module did not produce a consensus.

To simulate such a scenario, we assumed all mate pairs were sequenced from a fragment of a given length that was sampled from a Normal distribution N(μ,σ). In Table 3, we show the results for different parameters of the Normal distribution. Since the layout positions are now unreliable, we compute all pairwise overlaps in a given window and rely on the consistency-enhanced alignment graph to pull the reads into the right position. The results strongly support the assumption that this procedure is more robust than classical consensus approaches. This could be confirmed by aligning the consensus sequence of each tool with the source insert sequence using MUMmer (Kurtz *et al.*, 2004) and the NUCmer program (Delcher *et al.*, 2002). Two example alignments are shown in Figure 7. These results confirm the assumption that the consistency extension seems to be superior to the other approaches because it differentiates between random overlaps among two reads

Table 2. *Porphyromonas gingivalis W83*: results for all four sequencing datasets

Dataset	Assembler	Number of contigs	N50 size	Max size	Time (s)	Number of differences
E8YURXS01	CA	243	79 797	242 219	3272	26 (15)
	CA+Seq	252	79 787	242 223	7540	24 (15)
E8YURXS02	CA	244	78 965	241 877	3451	15 (12)
	CA+Seq	242	78 969	241 879	7956	16 (12)
E9T0MN001	CA	337	32 692	154 508	4177	7 (7)
	CA+Seq	335	32 691	154 509	5344	9 (7)
E9T0MN002	CA	328	35 105	131 244	4038	13 (9)
	CA+Seq	332	35 104	131 243	5161	14 (9)

The number of contigs, the N50 contig size, the largest contig size and the total running time is shown. The last column indicates the number of differences of an alignment of the largest contig to the provisional reference sequence. In brackets are the number of common differences. The N50 size was calculated by sorting all contigs according to length and computing the incremental sum until this sum exceeds 50% of the total length of all contigs. The last contig size used in this summation is the N50 size. CA is the Celera Assembler with the default consensus and CA+Seq is the Celera Assembler version using the proposed consensus tool.

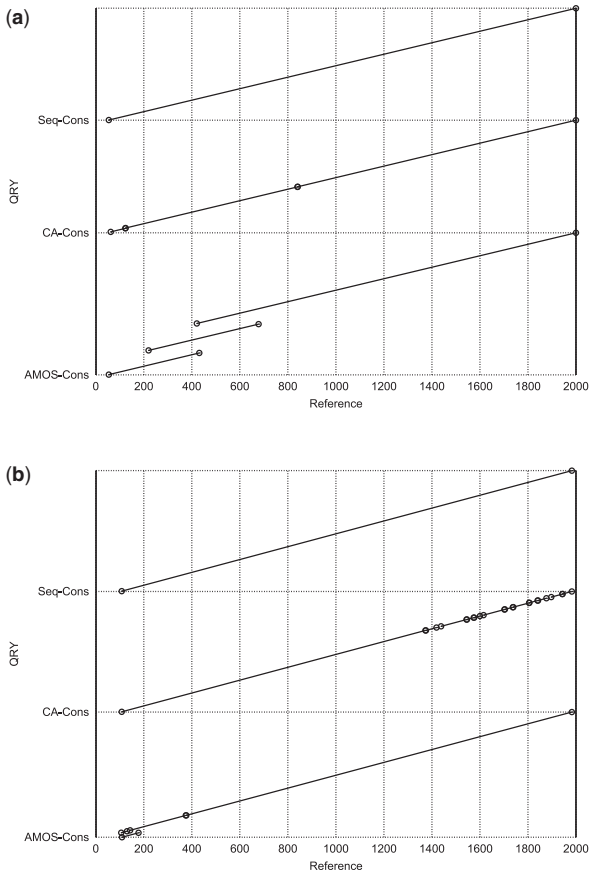

Fig. 7. Alignment of each consensus with the source insert sequence where coverage >2. Straight lines indicate matching segments and line endpoints are circled. Errors and gaps at the beginning and at the end of the source insert sequence are due to an insufficient sampling of reads at these positions (coverage $\leq$ 2). (**a**) Consensus to reference alignment: Read length 200, N(2000, 50). (**b**) Consensus to reference alignment: Read length 800, N(2000, 200).

Table 3. Insert sequencing: given a set of mapped reads and a set of unmapped mate pairs we can approximate the positions of the unmapped mate pairs from the library size and standard deviation parameters

Setting				Tool	N (2000, 50)		N (2000, 100)		N (2000, 200)		N (2000, 400)	
Source length	Read length	Coverage	Error rate (%)		Errors	Time (s)	Errors	Time (s)	Errors	Time (s)	Errors	Time (s)
2000	800	10×	2	AMOS-Cons	0	1.01	3	2.03	>20	4.02	>20	7.63
				CA-Cons	1	0.22	3	0.28	14	1.55	–	–
				Seq-Cons	0	4.58	2	4.24	0	4.80	0	4.84
2000	200	20×	2	AMOS-Cons	>20	1.39	>20	4.48	>20	12.90	>20	18.92
				CA-Cons	6	0.82	–	–	–	–	–	–
				Seq-Cons	0	7.12	0	7.79	0	7.37	>20	6.59
2000	35	30×	2	AMOS-Cons	>20	5.54	>20	9.03	>20	16.33	>20	10.41
				CA-Cons	–	–	–	–	–	–	–	–
				Seq-Cons	0	14.60	0	14.01	>20	13.10	>20	12.06

Given such a Normal distribution N(μ,σ) we simulated reads under the above settings. We report the number of errors in the consensus sequence with respect to the true insert sequence for all positions with coverage > 2. AMOS-Cons is the consensus module of the AMOS library, CA-Cons is the consensus module of the Celera Assembler and Seq-Cons is the proposed consensus tool. The '–' values indicate that the consensus module did not produce a consensus.

Table 4. Multi-read alignment: a clipped view of a multi-read alignment with two SNPs (bold font)

Alignment of haplotypes	
DQ453512	gcccgact**g**cc**c**ttctca
DQ453513	gcccgact**a**cc**t**ttctca
DQ453514	gcccgact**a**cc**t**ttctca
DQ453515	gcccgact**g**cc**c**ttctca

Seq-Cons	CA-Cons	AMOS-Cons
gcccg-act**g**	gcccg-actg	gcccg-a-ct**g**
gcccgtact**g**-c-c**c**ttctca	gcccgtactg-cc-cttctca	gcccgta-ct**g**-c-c**c**ttctca
gcccg-act**g**-c-c**c**ttctca	gcccg-actg-cc-cttctca	gcccg-a-ct**g**-c-c**c**ttctca
gcccg-act**a**-c-c**t**ttctca	gcccg-act-acc-tttctca	gcccg-a-ct**a**-c-c**t**ttctca
gcccg-act**g**-c-c**c**ttctca	gcccg-actg-cc-cttctca	gcccg-a-ct**g**-c-c**c**ttctca
gcccg-act**a**-c-c**t**ttctca	gcccg-act-acc-tttctca	gcccg-a-ct**a**-c-c**t**ttctca
gcccg-act**g**-c-c**c**ttctca	gcccg-actg-cc-cttctca	gcccg-a-ct**g**-c-c**c**ttctca
gcccg-act**a**-c-c**t**ttctca	gcccg-act-acc-tttctca	gcccg-a-ct**a**-c-c**t**ttctca
gcccg-act**g**-c-c**c**ttctca	gcccg-actg-cc-cttctca	gcccg-a-ct**g**-c-c**c**ttctca
gcccg-act**a**-c-c**t**ttctca	gcccg-act-acc-tttctca	gcccg-a-ct**a**-c-c**t**ttctca
gcccg-act**g**-c-c**c**ttctca	gcccg-actg-cc-cttctca	gcccg-a-ct**g**-c-c**c**ttctca
gcccg-act**a**-c-c**t**ttc-ca	gcccg-act-acc-tttc-ca	gcccg-a-ct**a**-c-c**t**ttc-ca
gcccg-act**g**ac-c**t**ttctca	gcccg-actg-acc-tttctca	gcccg-a-ct**g**ac-c**t**ttctca
gcccg-act**g**-c-c**c**-tctca	gcccg-actgc-cc-tct-ca	gcccg-a-ct**g**-c-c**c**t-ctca
gcccg-act**a**-c-c**t**ttctca	gcccg-acta-cctttct-ca	gcccg-a-ct**a**-c-c**t**ttctca
tcccg-act**g**-c-c**c**ttctca	tcccg-actgc-cc-ttct-ca	tcccg-a-ct**g**-c-c**c**ttctca
gcccg-act**a**-a-c**t**ttctca	gcccg-acta-a-ctttct-ca	gcccg-a-ct**a**-a-c**t**ttctca
gcccgaact**g**-c-c**c**ttctca	gcccgaactgc-cc-ttct-ca	gcccg-aact**g**-c-c**c**ttctca
gcccg-act**g**-c-c**c**ttctca	gcccg-actgc-cc-ttct-ca	gcccg-a-ct**g**-c-c**c**ttctca
gcccg-act**a**-c-c**t**ttccca	gcccg-acta-cctttcc-ca	gcccg-a-ct**a**-c-c**t**ttccca
gcccg-act**a**-cac**t**ttctca	gcccg-actaca-ctttct-ca	gcccg-a-ct**a**-cac**t**ttctca
gcccg-actg-c-ccttctca	gcccg-acta-cc-tttctca	gcccg-a-ctg-c-ccttctca

The alignment of the haplotypes is shown at the top. Below are all the aligned reads spanning this region for a given consensus tool. The last row indicates the consensus letter.

and overlaps confirmed by multiple reads. Table 3 also clearly indicates that the insert sequencing task becomes more difficult for short reads because they induce an increasing number of random overlaps.

3.3 Variation analyses

A mere high-quality consensus without an accurate multi-read alignment, where all the sequencing errors and DNA polymorphisms can be readily identified, is insufficient for a sound variation analysis. Several applications such as separating haplotypes, calling SNPs, or repeat resolution rely on the multi-read alignment itself. The difficulty for the algorithms is that besides sequencing errors, now source sequence variation further complicates the problem. For the *Taeniopygia guttata* (zebra finch) mitochondrion the NCBI Genome database provides four submitted haplotypes (Accessions: DQ453512 - DQ453515). Although there is a reference genome for this species (Accession: NC_007897), we took the four haplotypes to sample reads in order to test our algorithm in case of sequence variation. To quantify the amount of variation, we first aligned all four haplotypes. Because of high-sequence similarity, this could be easily done with MAFFT (Katoh *et al.*, 2002). The final multiple sequence alignment of the four haplotypes revealed 104 SNP locations and one insertion. We then simulated reads of length 200 with 2% error rate from each single haplotype at 5-fold coverage. All reads were combined in a final testing set. On this set, all consensus tools computed a mixture of the four haplotypes as the consensus sequence. We then inspected the multi-read alignments at the SNP locations manually to identify potential misalignments. In Table 4, the multi-read alignments of all tools are shown for a short segment with two SNP. Both the Seq-Cons and AMOS-Cons alignments allow a simple column-based SNP identification and consensus calling, since all alleles of the SNP ended up in the same column. The CA-Cons alignment requires a more sophisticated approach to correctly call the consensus sequence: the haplotype confirmed by the largest number of ungapped reads is picked (Denisov *et al.*, 2008). The algorithm used by CA-Cons may have problems if the read error rate is very high and every haplotype is confirmed by only one read.

4 DISCUSSION

In contrast to the monolithic assemblers, our own approach favors a highly modular design. All components of the algorithm are part of the generic SeqAn library (Döring *et al.*, 2008) for sequence analysis and individual components can be exchanged. Current work, for instance, focusses on the development of a new overlap

computation method based upon seed and extend algorithms. The implemented dynamic programming-based solution has a quadratic runtime for each pair of overlapping reads. An index based all against all comparison is significantly faster in practice. Building the index takes a total of $\mathcal{O}(n)$ time where n is the total length of all reads. Then a q-gram filter such as SWIFT (Rasmussen *et al.*, 2006) can be used to efficiently identify potential overlaps. The runtime of this filter depends on the size of q and this in turn depends on the sequencing error rate. The verification phase simply extends the identified seeds from the filtration phase and checks whether the computed overlap is above a user-defined quality and length threshold.

The proposed consensus algorithm might also be helpful to determine expressed sequence tags (EST) consensus sequences (Malde *et al.*, 2005). For this kind of application, the tool should be integrated into a complete EST analysis pipeline that includes other important steps such as EST fragment clustering and splice variant detection. From our own experience of integrating our tool into the Celera Assembler, we believe that such an integration is rather simple given the modular design and simple interfaces of our tool. Additionally, the tool could also be used for realignment of a set of given contigs similar to the ReAligner program (Anson and Myers, 1997).

The quality of the multi-read alignment provides several opportunities for future research. First, the alignment lends itself for an accurate genetic variation analysis, including an improved SNP calling or the retrieval of multiple consensus sequences for polyploid organisms (Denisov *et al.*, 2008). Second, the alignment might improve the performance of repeat resolution algorithms in DNA sequence assembly (Kececioglu and Ju, 2001). Third, as noticed in the Section 1, the approach is certainly fundamental for reference-guided assembly methods to infer newly inserted sequence segments with respect to a reference genome. Lastly, the approach could prove useful to close the gaps between a scaffold's contigs or to bridge small repeat regions.

ACKNOWLEDGEMENTS

We wish to thank the AMOS group at the CBCB at the University of Maryland for valuable input on the use of the AMOS library.

Funding: National Institute of General Medical Sciences (NIGMS) (grant R01-GM077117-02 to S.K. and G.D.)

Conflict of interest: none declared.

REFERENCES

Anson,E.L. and Myers,E.W. (1997). Realigner: a program for refining dna sequence multi-alignments. In *Proceedings of the first annual international conference on computational molecular biology, RECOMB '97*. ACM Press, New York, NY, USA, pp. 9–16.

Batzoglou,S. *et al*. (2002). ARACHNE: a whole-genome shotgun assembler. *Genome Res.*, **12**, 177–189.

Churchill,G.A. and Waterman,M.S. (1992). The accuracy of DNA sequences: estimating sequence quality. *Genomics*, **14**, 89–98.

Delcher,A.L. *et al*. (2002). Fast algorithms for large-scale genome alignment and comparison. *Nucleic Acids Res.*, **30**, 2478–2483.

Denisov,G. *et al*. (2008). Consensus generation and variant detection by Celera Assembler. *Bioinformatics*, **24**, 1035–1040.

Döring,A. *et al*. (2008). SeqAn - an efficient, generic C++ library for sequence analysis. *BMC Bioinformatics*, **9**, 11.

Edgar,R.C. (2004). MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Res.*, **32**, 1792–1797.

Feng,D.-F. and Doolittle,R.F. (1987). Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *J. Mol. Evol.*, **25**, 351–360.

Gotoh,O. (1982). An improved algorithm for matching biological sequences. *J. Mol. Biol.*, **162**, 705–708.

Gotoh,O. (1990). Consistency of optimal sequence alignments. *BMB: Bull. Math. Biol.*, **52**.

Havlak,P. *et al*. (2004). The atlas genome assembly system. *Genome Res.*, **14**, 721–732.

Huang,X. *et al*. (2003). PCAP: A whole-genome assembly program. *Genome Res.*, **13**, 2164–2170.

Jacobson,G. and Vo,K.-P. (1992). Heaviest increasing/common subsequence problems. In *Proceedings of the Third Annual Symposium on Combinatorial Pattern Matching*, *CPM '92*. Springer-Verlag, London, UK, pp. 52–66.

Katoh,K. *et al*. (2002). MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform. *Nucleic Acids Res.*, **30**, 3059–3066.

Kececioglu,J.D. (1993). The maximum weight trace problem in multiple sequence alignment. In *Proceedings of the Forth Annual Symposium on Combinatorial Pattern Matching*, *CPM '93*. London, UK. Springer-Verlag, pp. 106–119.

Kececioglu,J. and Ju,J. (2001). Separating repeats in DNA sequence assembly. In *Proceedings of the Fifth Annual International Conference on Computational Biology*, *RECOMB '01*. New York, NY, USA. ACM Press, pp. 176–183.

Kurtz,S. *et al*. (2004). Versatile and open software for comparing large genomes. *Genome Biol.*, **5**, R12.

Malde,K. *et al*. (2005). A graph based algorithm for generating EST consensus sequences. *Bioinformatics*, **21**, 1371–1375.

Mullikin,J.C. and Ning,Z. (2003). The Phusion assembler. *Genome Res.*, **13**, 81–90.

Myers,E.W. *et al*. (2000). A whole-genome assembly of Drosophila. *Science*, **287**, 2196–2204.

Notredame,C. *et al*. (2000). T-Coffee: a novel method for fast and accurate multiple sequence alignment. *J. Mol. Biol.*, **302**, 205–217.

Pop,M. *et al*. (2004). Comparative genome assembly. *Brief. Bioinform.*, **5**, 237–248.

Rasmussen,K. *et al*. (2006). Efficient q-gram filters for finding all epsilon-matches over a given length. *J. Comput. Biol.*, **13**, 296–308.

Rausch,T. *et al*. (2008). Segment-based multiple sequence alignment. *Bioinformatics*, **24**, i187–i192.

Sankoff,D. and Kruskal,J.B. (1983). *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*. Addison-Wesley, Reading, MA.

Schatz,M. *et al*. (2007). Hawkeye: an interactive visual analytics tool for genome assemblies. *Genome Biol.*, **8**, R34.

Sokal,R.R. and Michener,C.D. (1958). A statistical method for evaluating systematic relationships. *Univ. Kansas Sci. Bull.*, **38**, 1409–1438.

Sommer,D. *et al*. (2007). Minimus: a fast, lightweight genome assembler. *BMC Bioinformatics*, **8**, 64.

Thompson,J.D. *et al*. (1994). CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Res.*, **22**, 4673–4680.

Wang,L. and Jiang,T. (1994). On the complexity of multiple sequence alignment. *J. Comput. Biol.*, **1**, 337–348.