An Integrated Approach to Comparative Assembly

John Healy[1], Desmond Chambers[2]

[1] Department of Computing & Mathematics, Galway-Mayo Institute of Technology, Ireland {john.healy@gmit.ie}, [2] Department of Information Technology, National University of Ireland Galway, Ireland {des.chambers@nuigalway.ie}

Abstract. We describe a novel approach to comparative assembly that directly integrates anchoring alignments into the contig assembly process, enabling the extension of contig construction through the boundaries of repeat nodes in a compressed de Bruijn graph. Our method exploits anchoring alignments, paired-read constraints and read threading as path selection heuristics while an assembly graph is transversed during contig construction. Tests and benchmarks against preeminent implementations of both comparative and *de novo* assembly models demonstrate that the approach can significantly increase the contiguity of an assembly without inducing a large number of misjoins and structural errors.

Keywords: Integrated assembly, comparative assembly, anchoring.

1 Introduction

De novo models of genome assembly are characterised by the identification of repetitive sequences and the merging of unambiguous nodes in a graph up to the edges of repeat boundaries [1, 2]. In both the *Overlap-Layout-Consensus* and de Bruijn graph models, read length constrains contig assembly by placing limits on the ability of an assembler to resolve repeat-induced conflicts by circumnavigating repetitive regions during graph transversal [3]. Consequently, a separate scaffolding phase, involving the application of paired-read and other constraints, is typically used to resolve the ambiguities induced by repeats and to direct the merging of unitigs.

Comparative assembly models attempt to resolve the conflicts effectuated by repeats by utilising the nucleotide or protein sequences of a closely related species to guide the contig assembly process. The *Alignment-Layout-Consensus* model, employed in AMOS [4] and Mosaik [5], eschews the overlap phase of assembly completely, relying on alignment and anchoring techniques to provide positional information to the layout phase. While this approach directly integrates anchoring information into the contig assembly process, it requires a high degree of structural fidelity between the genome being assembled and the reference sequence.

Alternative *post hoc* comparative assembly techniques, such as those described by Gnerre *et al* [6] and others [7-10] use *de novo* assemblers to generate contigs and scaffolds, which are subsequently reified by their alignment against a reference genome. Such approaches to comparative assembly do not integrate anchoring or

alignment information directly into the *de novo* contig assembly process and have no native mechanism for identifying and circumnavigating repeat boundaries.

This paper describes how the direct incorporation of anchoring information enables the assembly of contigs up to and through the boundaries of repetitive nodes in an assembly graph. In contrast with existing approaches, including work previously published by the authors [11], the comparative assembly approach presented in this paper directly integrates anchoring information into the contig assembly process and can accommodate factors such as weak homology, repetitive sequences and structural variations between the target and reference genomes.

The remainder of this paper includes a description of the integrated assembly method in the next section. This is followed by the presentation and discussion of the results obtained from benchmarking and tests against preeminent examples of alternative assembly models.

2 Methods

The assembly of a set of shotgun sequence reads into a collection of contigs is both a multifaceted and complex process. The methods described in this section have been implemented in a prototype assembler written in Java. At a general level, the integrated approach requires the decomposition of each sequence read into a set of overlapping *k*-mers. Each *k*-mer is anchored to a reference genome using the fuzzy approach described previously by the authors [12] and added to a de Bruijn graph. In contrast with existing comparative techniques, anchoring information is directly exploited during the construction of contigs, by utilising read alignments as a path selection heuristic during transversal of the de Bruijn graph. The kernel of our method is to use anchoring information, paired-read and read threading to extend the construction of contigs in a compressed de Bruijn graph. As the anchoring mechanism is used in tandem with conventional *de novo* techniques when expanding graph nodes, structural differences between the target and reference genomes are less likely to stymie contig processing.

The graph transversal process requires the identification of all potential starting nodes in a compressed de Bruijn graph. The set of starting nodes, each of which is a potential contig, corresponds to all of the source and compressed nodes in the graph. The contig assembly process commences by processing a queue of candidate starting nodes, sorted by their anchoring index relative to the reference genome. Where a compressed node is aligned to more than one anchor, either by spanning separate anchors or though the alignment of read complements, its anchor index is computed from a majority count of the lowest indexed anchor supported by the underlying sequence reads. The process of contig extension involves the expansion of nodes along a path through the compressed graph and the addition to a contig of the unique sequence information encapsulated by each node. It is noteworthy that, as a consequence of node compression, the remaining nodes in the de Bruijn graph of size *k* are either junctions or bifurcations, representing alternative assembly paths.

Resolving the ambiguity induced by these nodes is imperative for the correct assembly of a genome.

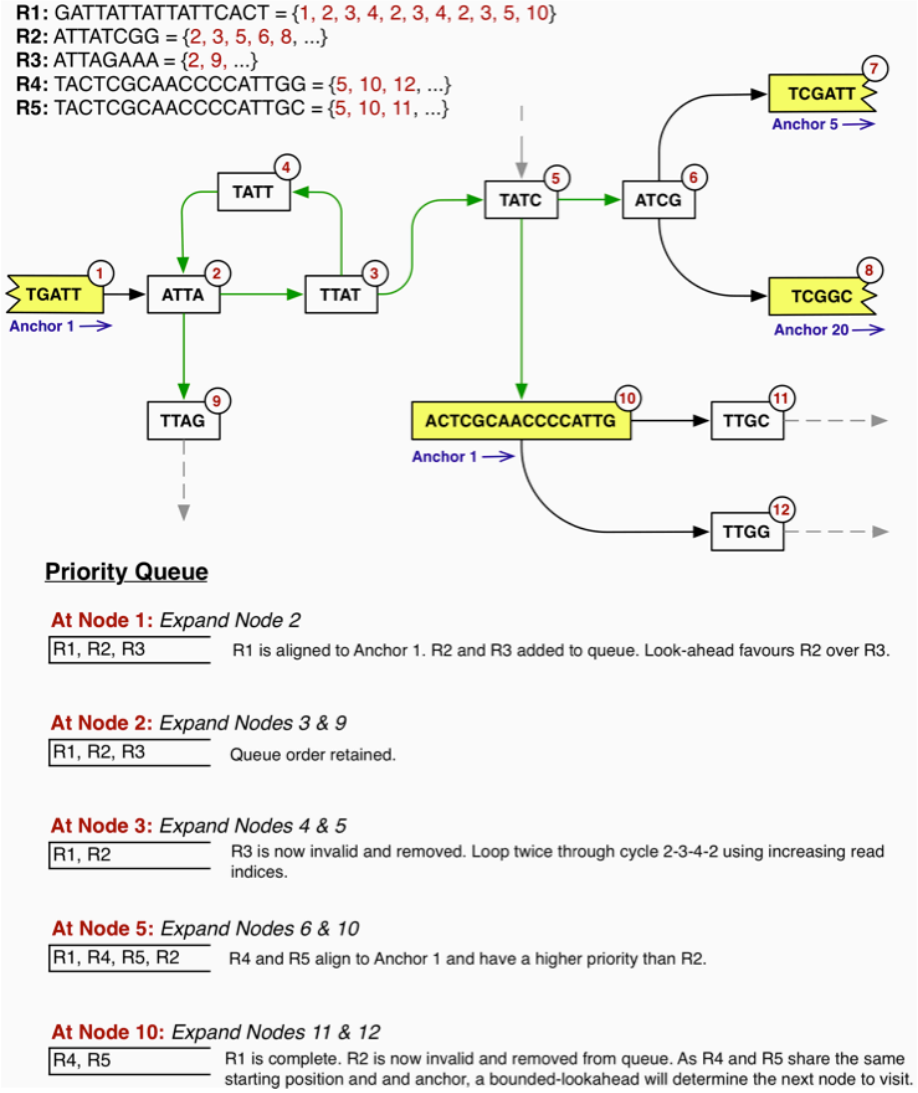

Fig. 1. Expanding nodes during graph transversal. Compressed nodes are depicted in yellow. Repetitive nodes of size k are shown in white. A bounded look-ahead from Node 2, with a depth of 3, is depicted by the green edges.

Our contig assembly method is based on the premise that anchoring information, in conjunction with read and paired read threading, provides sufficient information to

resolve the ambiguities effectuated by junctions and bifurcations in the compressed de Bruijn graph. The contig assembly process employs a priority queue when determining the next node to expand during graph transversal.

Procedure: offer

Input: Node node, ReadIndex r_i, int contigIndex, Queue Q

Output: void

1. **IF** read(r_i).processed **OR** read(r_i).complement().processed **THEN RETURN**
2. **IF** read(r_i) ∈ Q **THEN**
3. **IF** ∃ $r_j \in Q$ | read(r_j).complete **THEN RETURN**
4. **IF** ∃ $r_j \in Q$ | $r_i = r_j + len(r_j.\text{node}) - k - 1$ **OR**
   ∃ $r_j \in Q$ | $r_i = (len(r_j.\text{node}) - k - 1) - r_j^{mi} + 2$ **THEN**
5. **IF** ∃ $r_j \in Q$ | $r_i < r_j$ **THEN RETURN**
6. read(r_i).validated ← TRUE
7. **IF** $r_i^{ci} + len(\text{read}(r_i)) - 1 \leq \text{contigIndex} + len(\text{node})$ **THEN**
8. read(r_i).complete ← TRUE
9. **END IF**
10. **END IF**
11. **ELSE**
12. **IF** $r_i = 1$ **THEN**
13. read(r_i).validated ←TRUE
14. **IF** $r_i^{mi} + len(\text{read}(r_i)) - 1 \leq len(\text{node})$ **THEN** read(r_i).complete ← TRUE
15. add read(r_i) to Q
16. **END IF**
17. **END IF**

Fig. 2. Constraints employed when expanding nodes in a compressed de Bruijn graph. The index of an existing read in the priority queue is denoted by r_j, with r_j^{mi} and r_i^{ci} denoting the index of an existing read in a merged node and the contig index of the current read respectively.

The priority queue exploits read indices, along with paired read constraints and anchoring information to determine queue membership and priority respectively. At

an abstract level, this general approach exhibits the behaviours associated with both a constraint-satisfaction search and a best-first search.

For each child node expanded, the indices of its constituent reads are offered to the priority queue, together with the current index of the contig being extended. Membership of the priority queue is constrained to those nodes containing read indices that are consistent with the underlying set of reads already added to a contig. As shown on lines 12-15 of Fig. 2, newly encountered reads, with a starting index of one, are labelled as validated and are added to the priority queue. Those reads that are entirely contained by a merged node are marked as complete, triggering the reverse complement of a read to be also flagged as processed. This prevents the subsequent processing of uncontested sub-paths of the graph that correspond to the reverse complement of a processed sequence and forestalls segments of a genome from being assembled twice.

For reads that have already been added to the priority queue, their continuing membership is predicated on the satisfaction of the disjunction of the two constraints described on line 4 of Fig. 2. This first of these predicates is only true where the value of a read index is equivalent to the sum of the previous read index and the length of the nucleotide sequence of the last processed node. The second predicate is necessary to handle the condition where the starting index of a read aligns to the middle of a compressed node. A further constraint is applied to read indices that comply with these predicates, ensuring that the lowest read index is selected from a node where the multiplicity of indices for a given read is greater than one. As this condition is only true where a cycle exists in the graph, selecting read indices in increasing order enables the assembler to reliably transverse loops. Consistent with the technique described for processing newly encountered contained reads, both completed reads and their reverse complements are labelled as processed.

The semantics for computing the comparability of elements in the priority queue are shown in Fig. 3. A salient feature of the comparator heuristics is the exploitation of anchoring information when determining the priority of nodes in the queue. The method *hasValidAnchor()* checks if an anchored read aligns to either the current anchor or its two unprocessed preceding or succeeding anchors. This technique enables the prototype to apply anchoring heuristics to both genomes that exhibit a strong synteny and those that contain segmental inversions. In the event of a tie between two reads, the original anchor score is used to determine priority.

In the absence of anchoring information, the comparator favours nodes connected by paired reads that are consistent with their distance constraints and orientation. Failing this, a bounded look-ahead is performed, that searches for succeeding merged nodes in the graph containing reads aligned to either the current anchor or its adjacent anchors. The bounded look-ahead is implemented as a depth-limited depth-first search using the algorithm described by Coppin [13], with a default look-ahead depth of four. Restricting the look-ahead to compressed nodes is necessary, as existing nodes of size k represent junctions and bifurcations and, even if anchored, only provide ambiguous positional information to the assembler.

The contig extension process and the expansion of graph nodes continue while validated nodes remain in the priority queue. If the priority queue returns a null object reference, the next unprocessed candidate source node is selected and a new contig is

started. This process of contig assembly iterates until the remaining list of unprocessed candidate source nodes has been exhausted.

Procedure: compare

Input: NodeQueueElement current, NodeQueueElement next, Queue Q

Output: -1, 1, 0 denoting less than, greater than, or equal to.

```
1.  IF current.validated AND NOT next.validated THEN RETURN -1;
2.  IF NOT current.validated AND next.validated THEN RETURN 1;
3.  IF hasValidAnchor(current) AND NOT hasValidAnchor(next) THEN RETURN -1
4.  ELSE IF NOT hasValidAnchor(current) AND hasValidAnchor(next) THEN RETURN 1
5.  ELSE IF hasValidAnchor(current) AND hasValidAnchor(next) THEN
6.      IF anchorScore(current) > anchorScore (next) THEN RETURN -1
7.      ELSE RETURN 1
8.  ELSE
9.      IF current.mate ∈ Q AND next.mate ∉ Q THEN RETURN -1
10.     ELSE IF current.mate ∉ Q AND next.mate ∈ Q THEN RETURN 1
11.     ELSE
12.         IF lookAhead(current) <= lookAhead(next) THEN RETURN -1
13.     ELSE RETURN 1
14. END IF
```

Fig. 3. Priority heuristics used in the node queue. The semantics of comparability exploit anchoring information to guide the assembler when expanding nodes. Paired read constraints and a bounded look-ahead are also used to determine node priority.

3 Results and Discussion

The integrated approach was benchmarked using synthetic error-free paired-end reads, extracted from a set of candidate genomes at 10X coverage, corresponding to a 99.995% sampling of each genome [14]. Sequence reads of 800bps, 400bps and 80bps were used to test the prototype, with the varying read lengths corresponding to Sanger, 454 and Illumina sequencing platforms.

The tests also included the benchmarking of the prototype against the preeminent implementations of *de novo* and comparative assembly, capable of utilising both

Sanger and short sequence reads. Cabog 7.0 [15], Velvet 1.2.06 [16] and AMOS 3.0.1 [4, 17] were selected as reference implementations of the de Bruijn graph, *Overlap-Layout-Consensus* and *Alignment-Layout-Consensus* assembly models respectively.

In addition to computing the N50 contig size, the contig sets produced by the different assemblers were subject to a quality analysis using the QUAST tool [18]. This was achieved by comparing each set of contigs assembled from synthetic reads against the original genome that they were extracted from. As noted by Narzisi and Mishra [19], the N50 metric emphasises contig size and is a poor metric for capturing contig quality. An aggressive or greedy assembly strategy may produce a large N50 value, but may also increase the number of misassemblies in contigs. Conversely, a conservative approach to contig extension may produce a higher quality assembly, but with a lower N50 value. Alternative assembly metrics such as those described by Vezzi *et al* [20], Salzberg *et al* [21] and, more recently by Gurevich *et al* [18] emphasis both the size and quality of contigs. In addition to the N50 contig size, QUAST computes an NGA50 metric using a reference genome, where the lengths of aligned blocks are counted instead of contig lengths. If a contig contains a misassembly with respect to the reference, the contig is broken into smaller sequences and the N50 value recomputed. QUAST detects a relocation misassembly if the left flanking sequence of a contig overlaps or aligns more than 1Kbps away from the right flanking sequence on a reference genome. An inversion misassembly is detected where flanking sequences align on different strands of each genome.

The results presented in this section were compiled from executing the suite of assemblers on an OSX 10.6.8 platform, with a single 3.2 GHz Intel Core i3 processor and 16GB of RAM. The prototype assembler required the additional provisioning of a Java HotSpot 1.6 64-bit virtual machine.

3.1 Comparison with the *Alignment-Layout-Consensus* Model

The effectiveness of the *Alignment-Layout-Consensus* model is predicated on accurately aligning sequence reads against a reference genome and then using alignment and paired-read information to create a layout graph and compute a consensus sequence. The results show a direct correlation between the DDH distance [22, 23] of the target and reference genomes and the contig sizes produced by AMOS. This is not unexpected, as the DDH distance metric and the alignments employed by AMOS are both computed from the high scoring pairs produced by MUMmer alignments. While the AMOS assemblies of B. *suis*, C. *pneumonia* and S. *aureus* produced contigs with a high NGA50 value, the target and reference genomes are highly homologous, with DDH values of 0.0861, 0.0260 and 0.1095 respectively. The contig sizes produced by AMOS rapidly reduce as the DDH distance increases.

In contrast with the integrated assembly approach used in the prototype, AMOS failed to produce sizable contigs for the assemblies of K. *pneumoniae*, M. *genitalium*, S. *typhimurium* and Synechococcus, where the high DDH distance stymied the alignment phase of assembly. Despite the weak homology between these genomes and their reference sequence, the prototype anchored a significant number of sequence reads and produced assemblies with a large NGA50 value and a small

number of errors. Using the same DDH metric to compute a required percentage identity match, the prototype anchored 46.65% of the K. *pneumoniae* reads, 64.1% of the M. *genitalium* reads and 56.62% of the 800bp reads used to assemble S. *typhimurium*. Furthermore, the low NGA50 value computed for the AMOS assemblies of S. *typhimurium* and Synechococcus demonstrate the vulnerability of conventional comparative assembly to structural variations between the target and reference genomes, as the reference genomes used in both assemblies contain significant inversions and rearrangements.

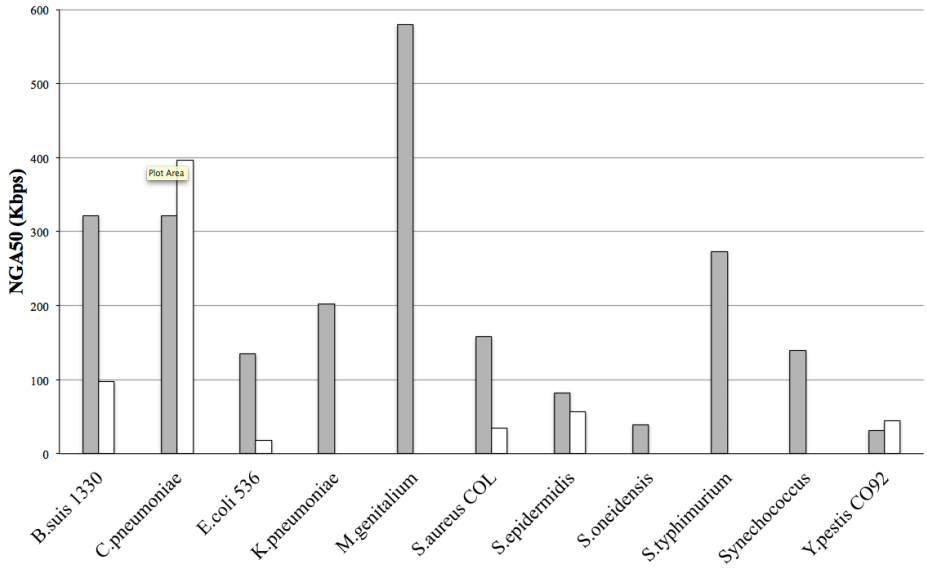

Fig. 4. Comparison of NGA50 values of the integrated (grey) and AMOS (white) assemblies.

The assemblies of E. *coli*, S. *aureus*, and Y. *pestis* illustrate the difficulty of comparative assembly with short reads. Despite producing both high contiguity and quality assemblies for these genomes with 400 and 800bp sequences, the NGA50 value of the AMOS assembly of E. *coli* 536 with 80bp reads is less than 1% of the N50 value. There were similar large discrepancies between the N50 and NGA50 values for the AMOS assemblies of S. *aureus* and Y. *pestis*. Given the relatively low DDH distances between these genomes and their reference sequence, structural variations, exacerbated by the small cluster size used by *AMOScmp-shortReads*, are the likely cause of the misassemblies reported. This explanation is supported by the high quality AMOS assembly of C. *pneumonia*, where 99.72% of the genome was assembled into a single contig with an NGA50 value of 1216Kbps. Read length also stymied the assembly of 80bp sequence reads using the prototype. The heuristics employed to sort and constrain membership of the priority queue used by the contig assembler, rely heavily on read threading and anchor alignments, both of which are significantly influenced by read length.

3.2 Comparison with *de novo* Assembly Models

The number of contigs and the corresponding NGA50 quality values shown in Table 1 illustrate the versatility of integrated assembly. In contrast with the *Alignment-Layout-Consensus* model, which breaks down in the presence of weak homology and structural rearrangements, the integrated approach exhibits levels of contiguity and quality commensurate with the results produced by preeminent *de novo* assemblers based on the *Overlap-Layout-Consensus* and de Bruijn graph models. When expanding graph nodes during contig assembly, anchoring information facilitates the extension of contigs through repeat boundaries, compensating for the shortcomings of the paired read and read threading techniques used by *de novo* assemblers. In the context of a weak homology between the genome being assembled and the reference sequence, the *de novo* aspects of the integrated approach compensate for the shortcomings of a purely comparative assembly.

Table 1. Comparison of assembly results using 400bp sequence reads.

Organism	**Integrated**		**Velvet**		**Cabog**	
	Contigs	*NGA50*	*Contigs*	*NGA50*	*Contigs*	*NGA50*
B.*suis* 1330	16	321	91	163	25	241
C.*pneumoniae*	9	321	39	693	15	168
E.*coli* 536	68	135	493	132	72	197
K.*pneumoniae*	49	202	325	140	63	144
M.*genitalium*	1	580	168	80	7	371
S.*aureus* COL	35	158	359	110	36	236
S.*epidermidis*	24	82	302	49	47	77
S.*oneidensis*	90	39	623	47	171	54
S.*typhimurium*	36	273	349	171	47	185
Synechococcus	9	139	138	161	26	175
Y.*pestis* CO92	81	32	511	38	145	52

Vezzi *et al* [24] argue that short reads have made the assembly problem more difficult, due to the complexity involved in resolving long repeats. Despite its successful adaptation to hybrid assemblies [3, 25], testing and benchmarking revealed the inability of Cabog to generate substantial contigs sizes for 80bp reads, highlighting a limitation of the Best Overlap Graph (BOG) [15] used by the assembler. In a hybrid assembly, the transitive reduction of the BOG creates a bias towards longer sequences, as shorter Illumina-sized reads are more likely to be discarded from the graph. Consequently, the contig sizes produced by a hybrid assembly will be characteristic of that expected from longer sequence reads. As the N50 metric is a measure of the fragmentation of an assembly [26], the small N50 and

NGA50 values computed for the Cabog assemblies with 80bp reads demonstrate that the assembler is better suited for use with the longer sequence reads that it is designed for. This is consistent with the findings of Li *et al* [3], who noted the shortcomings of *Overlap-Layout-Consensus* approach in assembling short Illumina-length reads.

A notable feature of the Cabog assemblies with longer reads is the effective use of scaffolding to reduce the multiplicity of contigs and to boost contig size. The explicit support for read coherence in the overlap graph facilitates the identification of repeats during scaffolding, as repeat contigs will contain more reads than the level of coverage permits. Despite aggressively merging unitigs using mate-pair constraints as a greedy heuristic, Cabog produced assemblies of high contiguity and high accuracy with longer reads.

Although Velvet utilises read threading during graph construction to compute the level of coverage at each node [16, 27], the conventional de Bruijn graph approach lacks read coherence [28]. Consequently, de Bruijn graph assemblers have a limited capacity to extend contigs through repetitive nodes, without resorting to greedy heuristics. Li *et al* [3] contend that the ability of a de Bruijn graph assembler to resolve repeats it primarily determined by the size of k. Their view is supported by Simpson and Durbin [29] who argue that decomposing a sequence into k-mers results in a collapse of repetitive sequences whose size > k. This observation is borne out by the results, as the NGA50 value of the assemblies produced by Velvet is generally significantly lower than that produced by Cabog, particularly for the assemblies of Y. pestis, S. *oneidensis* and S. *epidermidis*. In contrast with the conventional de Bruijn graph approach, the incorporation of read threading and anchoring information into the integrated assembly process added read coherence to the de Bruijn graph used by the prototype. Using read indices, mate-pair links and anchor alignments as path selection heuristics enabled the prototype assembler to boost the NGA50 value, by extending contigs through repeat nodes in the de Bruijn graph.

4 Conclusions

An integrated approach to comparative assembly enables the extension of contigs through repeat boundaries in a de Bruijn graph, by exploiting anchoring, paired read and read threading information. In contrast with current approaches to comparative assembly that are constrained by a requirement for both strong homology and synteny, an integrated assembly functions well in the presence of structural rearrangements and weak homology, by combining key aspects of both *de novo* and comparative assembly models. The results of tests and benchmarks against the preeminent implementations of alternative assembly approaches suggest that integrating comparative techniques directly into the contig assembly process can significantly increase both the contiguity and quality of an assembly.

References

1. Idury, R., Waterman, M.: A new algorithm for DNA sequence assembly. Journal of Computational Biology 2, 291-306 (1995)
2. Myers, E., Sutton, G., Delcher, A., Dew, I., Fasulo, D., Flanigan, M., Kravitz, S., Mobarry, C., Reinert, K., Remington, K.: A whole-genome assembly of Drosophila. Science 287, 2196 (2000)
3. Li, Z., Chen, Y., Mu, D., Yuan, J., Shi, Y., Zhang, H., Gan, J., Li, N., Hu, X., Liu, B.: Comparison of the two major classes of assembly algorithms: overlap-layout-consensus and de-bruijn-graph. Briefings in Functional Genomics 11, 25-37 (2012)
4. Pop, M., Phillippy, A., Delcher, A., Salzberg, S.: Comparative genome assembly. Briefings in bioinformatics 5, 237 (2004)
5. Strömberg, M., Lee, W.: MOSAIK read alignment and assembly program. (2009)
6. Gnerre, S., Lander, E., Lindblad-Toh, K., Jaffe, D.: Assisted assembly: how to improve a de novo genome assembly by using related species. Genome biology 10, R88 (2009)
7. Cattonaro, F., Policriti, A., Vezzi, F.: Enhanced reference guided assembly. pp. 77-80. IEEE, (Year)
8. Nijkamp, J., Winterbach, W., van den Broek, M., Daran, J.M., Reinders, M., de Ridder, D.: Integrating genome assemblies with MAIA. Bioinformatics 26, i433-i439 (2010)
9. Reinhardt, J., Baltrus, D., Nishimura, M., Jeck, W., Jones, C., Dangl, J.: De novo assembly using low-coverage short read sequence data from the rice pathogen Pseudomonas syringae pv. oryzae. Genome Research 19, 294 (2009)
10. Schneeberger, K., Ossowski, S., Ott, F., Klein, J.D., Wang, X., Lanz, C., Smith, L.M., Cao, J., Fitz, J., Warthmann, N.: Reference-guided assembly of four diverse Arabidopsis thaliana genomes. Proceedings of the National Academy of Sciences 108, 10249 (2011)
11. Healy, J., Chambers, D.: De Novo Draft Genome Assembly Using Fuzzy K-mers. In: BIOTECHNO 2011, The Third International Conference on Bioinformatics, Biocomputational Systems and Biotechnologies, pp. 104-109. (2011)
12. Healy, J., Chambers, D.: Fast and Accurate Genome Anchoring Using Fuzzy Hash Maps. In: 5th International Conference on Practical Applications of Computational Biology & Bioinformatics (PACBB 2011), pp. 149-156. Springer, (2011)
13. Coppin, B.: Artificial intelligence illuminated. Jones & Bartlett Learning (2004)
14. Lander, E., Waterman, M.: Genomic mapping by fingerprinting random clones: a mathematical analysis. Genomics 2, 231-239 (1988)
15. Miller, J.R., Delcher, A.L., Koren, S., Venter, E., Walenz, B.P., Brownley, A., Johnson, J., Li, K., Mobarry, C., Sutton, G.: Aggressive assembly of pyrosequencing reads with mates. Bioinformatics 24, 2818-2824 (2008)
16. Zerbino, D., Birney, E.: Velvet: Algorithms for de novo short read assembly using de Bruijn graphs. Genome Research 18, 821 (2008)
17. Treangen, T.J., Sommer, D.D., Angly, F.E., Koren, S., Pop, M.: Next generation sequence assembly with AMOS. Curr Protoc Bioinformatics (2011)
18. Gurevich, A., Saveliev, V., Vyahhi, N., Tesler, G.: QUAST: quality assessment tool for genome assemblies. Bioinformatics 29, 1072-1075 (2013)
19. Narzisi, G., Mishra, B.: Comparing de novo genome assembly: The long and short of it. PloS one 6, e19175 (2011)
20. Vezzi, F., Narzisi, G., Mishra, B.: Feature-by-feature, evaluating de novo sequence assembly. PloS one 7, e31002 (2012)

21. Salzberg, S.L., Phillippy, A.M., Zimin, A., Puiu, D., Magoc, T., Koren, S., Treangen, T.J., Schatz, M.C., Delcher, A.L., Roberts, M.: GAGE: A critical evaluation of genome assemblies and assembly algorithms. Genome research 22, 557-567 (2012)
22. Auch, A.F., Klenk, H.-P., Goker, M.: Standard operating procedure for calculating genome-to-genome distances based on high-scoring segment pairs. Standards in Genomic Sciences 2, 142 (2010)
23. Auch, A.F., von Jan, M., Klenk, H.-P., Goker, M.: Digital DNA-DNA hybridization for microbial species delineation by means of genome-to-genome sequence comparison. Standards in Genomic Sciences 2, 117 (2010)
24. Vezzi, F., Narzisi, G., Mishra, B.: Reevaluating assembly evaluations with feature response curves: GAGE and assemblathons. PloS one 7, e52210 (2012)
25. Losada, L., Varga, J.J., Hostetler, J., Radune, D., Kim, M., Durkin, S., Schneewind, O., Nierman, W.C.: Genome sequencing and analysis of Yersina pestis KIM D27, an avirulent strain exempt from select agent regulation. PloS one 6, e19054 (2011)
26. Haiminen, N., Kuhn, D.N., Parida, L., Rigoutsos, I.: Evaluation of methods for de novo genome assembly from high-throughput sequencing reads reveals dependencies that affect the quality of the results. PloS one 6, e24182 (2011)
27. Zerbino, D.R., McEwen, G.K., Margulies, E.H., Birney, E.: Pebble and rock band: heuristic resolution of repeats and scaffolding in the velvet short-read de novo assembler. PloS one 4, e8407 (2009)
28. Myers, E.: The fragment assembly string graph. Bioinformatics 21, (2005)
29. Simpson, J.T., Durbin, R.: Efficient de novo assembly of large genomes using compressed data structures. Genome research 22, 549-556 (2012)