Vol. 00 no. 00 2011
Pages 1–9

# AlignGraph: algorithm for secondary *de novo* genome assembly guided by closely related references

Ergude Bao [1], Tao Jiang [1] and Thomas Girke [2*]

[1]Department of Computer Science and Engineering, [2]Department of Botany and Plant Sciences, University of California, Riverside, California 92521

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Associate Editor: XXXXXXX

**ABSTRACT**
**Motivation:** *De novo* assemblies of genomes remain one of the most challenging applications in next generation sequencing. Usually, their results are incomplete and fragmented into hundreds of contigs. Repeats in genome sequences and sequencing errors are the main reasons for these complications. With the rapidly growing number of sequenced genomes, it is now feasible to improve genome assemblies by guiding them with genomes from related species.
**Results:** Here we introduce AlignGraph, an algorithm for extending and joining *de novo* assembled contigs or scaffolds guided by closely related reference genomes. It aligns paired-end (PE) reads and pre-assembled contigs or scaffolds to a close reference. From the obtained alignments, it builds a novel data structure, called the *paired-end multi-positional de Bruijn graph*. The incorporated positional information from the alignments and PE reads allows to extend the initial assemblies, while avoiding incorrect extensions and early terminations. In our performance tests, AlignGraph was able to substantially improve the contigs and scaffolds from several assemblers. For instance, 26.1-79.4% of the contigs of *Arabidopsis thaliana* and human could be extended, resulting in improvements of common assembly metrics, such as an increase of the N50 of the extendable contigs by 81.2-172.2% and 1.8-44.6%, respectively. In another test, AlignGraph was able to improve the assembly of a published genome (*Arabidopsis* strain Landsberg) by increasing the N50 of its extendable scaffolds by 104.4%. These results demonstrate AlignGraph's efficiency in improving genome assemblies by taking advantage of closely related references.
**Availability:** The AlignGraph software can be downloaded for free from this site: https://github.com/baoe/AlignGraph.
**Contact:** thomas.girke@ucr.edu

## 1 INTRODUCTION

Recent advances in next generation sequencing (NGS) have made it possible to sequence new genomes at a fraction of the time and cost required only a few years ago. These improvements allow now experimental scientists to integrate genome sequencing approaches into their daily research. In the absence of a close reference genome, whole-genome shotgun NGS sequencing is the most common approach where a *de novo* assembly algorithm is used to join reads into longer continuous contigs and scaffolds. Most NGS *de novo* assemblers create an overlap or de Bruijn graph representing the connections among the reads and output the paths in the graph as assembled contigs. Examples of these algorithms include Edena (Hernandez *et al.* 2008), Velvet (Zerbino and Birney 2008), ABySS (Simpson *et al.* 2009), ALLPATHS-LG (Gnerre *et al.* 2011), SOAPdenovo (Li *et al.* 2010; Luo *et al.* 2012), MaSuRCA (Zimin *et al.* 2013), CABOG (Miller *et al.* 2008), Euler-USR (Chaisson *et al.* 2009), and IDBA (Peng *et al.* 2010). This *de novo* sequencing approach fundamentally differs from genome resequencing approaches, where the NGS reads are not assembled but aligned against a very similar reference genome using a variant tolerant short read alignment algorithm (Ossowski *et al.* 2008). The sequence of the target genome can then be inferred from the mismatches and indels observed in the alignment results. *De novo* assemblies tend to be computationally much more challenging than alignment-based approaches. Additional limitations include: (i) the assembly results are often fragmented into large numbers of contigs; (ii) the coverage of the genome by the assembled contigs/scaffolds is commonly incomplete; and (iii) the frequency of falsely assembled contigs can be high, due to chimeric joins. The most important reasons for these complications are usually sequencing errors, repeat sequences in the target genome, non-uniform sequencing depth, and limited read length of NGS data. These error sources result in false positive, incomplete and branched paths in the assembly graph, and thus greatly limit the lengths and completeness of the final contigs (Zerbino and Birney 2008; Chaisson and Pevzner 2008; Peng *et al.* 2010). Combining both *de novo* assembly and alignment-based approaches presents a powerful alternative when a closely related reference genome sequence is available, but its genetic differences relative to the target genome are too pronounced to resolve them with an alignment approach alone (Schneeberger *et al.* 2011; Phillippy *et al.* 2008; Schatz *et al.* 2013). In this case, one can first assemble the reads into contigs and then align them together with the reads to the reference. The much longer contigs facilitate the identification of complex rearrangements, while the read alignments are useful for detecting smaller variations in regions that are not covered by contigs. Due to the rapidly increasing number of reference genomes becoming available for most organism groups, this reference-guided assembly approach will soon become the default option for many genome sequencing projects. Compared to *de novo* assemblies, reference-guided assemblies have many advantages. First, the alignments of the contigs and reads against the close reference provide valuable

[*]to whom correspondence should be addressed

proximity information that can be used to extend contigs with additional reads and to join contigs even if they overlap only by a few nucleotides. Second, the proximity information in the alignment can also be used to orient and order contigs along the reference to build a scaffold map of the entire assembly. Third, the alignment map can be used to evaluate the quality of contigs and pinpoint potential mis-assemblies.

Previous studies on reference-guided assemblies include the AMOScmp software (Pop *et al.* 2004a), an add-on tool for the ARACHNE assembler (Gnerre *et al.* 2009), and custom workflows largely based on existing assembly software (*e.g.* Schneeberger *et al.* 2011). The first two were designed primarily for Sanger reads, while the latter has been used for NGS genome assembly. Downstream of the primary assembly, scaffolding algorithms, such as RACA (Kim *et al.* 2013), can be used that order and orient pre-assembled contigs to a connection map by incorporating additional sequence information from mate pair or paired-end (PE) reads and/or from closely related genomes (Pop *et al.* 2004b; Boetzer *et al.* 2011; Dayarian *et al.* 2010; Gao *et al.* 2011; Salmela *et al.* 2011; Gritsenko *et al.* 2012). The resulting scaffolds contain often gaps, which are unresolved sequence areas between the original contigs. Dedicated gap filling algorithms can be used to partially fill these gaps (Boetzer and Pirovano 2012; Luo *et al.* 2012; Tsai *et al.* 2010). More recently, components of reference-based strategies have also been incorporated into some of the *de novo* assembly suites themselves such as the *cheat* mode option of ALLPATHS-LG (Gnerre *et al.* 2011) and IDBA-hybrid (unpublished).

This study proposes a novel algorithm, called AlignGraph, for improving the lengths and completeness of contigs or scaffolds by reassembling them with help provided by a reference genome of a closely related organism. In contrast to existing reference-assisted methods, AlignGraph is a *secondary assembly algorithm* that loads the alignment information of PE reads and pre-assembled contigs/scaffolds against the reference into a novel assembly graph, called the *PE multi-positional de Bruijn graph*, that we specifically designed for facilitating secondary assemblies. By traversing this graph, the contigs or scaffolds of the primary assembly can be extended and joined.

AlignGraph's functionalities are unique by solving several challenges in improving assembly results. As a de Bruijn graph-based method it solves limitations typical for many heuristic extension methods that are often used in the *de novo* assembly area (Warren *et al.* 2007; Jeck *et al.* 2007; Dohm *et al.* 2007). For instance, if there are multiple solutions how to extend a contig, then finding the correct one can be challenging with most heuristic methods. Those ambiguous solutions, that correspond to branched paths in the de Bruijn graph, are usually caused by repetitive sequences in genomes, and frequently lead to early terminations of the contig extension process. The de Bruijn graph method is often more efficient in finding the correct solution here, because the contextual information, required for resolving these ambiguities, is maintained in the graph (Zerbino and Birney 2008; Chaisson and Pevzner 2008). This issue is not as pronounced in assemblies with much longer Sanger reads, as those are much more likely to span non-repetitive regions with repetitive regions in between (Gnerre *et al.* 2009). Thus, it is particularly important to address this problem in assemblies with short reads. In comparison to the conventional de Bruijn graph, our PE multi-positional de Bruijn graph has several additional advantages. First, many branched paths can be eliminated directly in the graph with help of the additional PE read and alignment information. This simplifies the identification of correct paths. Second, many false positive paths, caused by sequencing errors, can be eliminated by correcting erroneous reads with correct reads that align to the same position in the reference genome. Third, guided by the alignment information to the reference genome, the PE multi-positional de Bruijn graph is less affected by regionally low read coverage that often gives rise to incomplete paths in the conventional de Bruijn graph. As a result, many incorrect extensions and early terminations can be avoided.

## 2 METHODS

### 2.1 AlignGraph Algorithm

This section describes the AlignGraph algorithm. Its workflow can be divided into the following three major steps. Figure 1B illustrates these steps with an example.

(i) *Alignment maps*. The PE reads are aligned against both the pre-assembled contigs and the close reference genome; and the contigs are aligned against the reference.

(ii) *Contig reassembly*. The alignment mapping results are used to construct a positional variant of the de Bruijn graph, called the *PE multi-positional de Bruijn graph*.

(iii) *Graph traversal*. The resulting graph is edited and traversed to obtain extended contigs.

Throughout the text, the source genome of the PE reads and the pre-assembled contigs is referred to as the *target genome*, whereas the genome of the closely related species for guiding the contig improvement steps is referred to as the *reference genome*. For simplicity, the following description of AlignGraph refers mostly to contigs, but it also applies to scaffolds containing a limited amount gaps.

*Prerequisites.* Prior to the above steps, the user is expected to generate genomic PE reads for the target genome of interest and to assemble them with a *de novo* NGS genome assembler. Since most genome assemblers perform better with PE than single end data, AlignGraph also depends on this sequence type. A major advantage of AlignGraph is its design to work with most genome assemblers, but the quality of the initial *de novo* assembled contigs is expected to impact the final results (see 3.2). For optimal results, it is also important to follow the recommendations of the chosen *de novo* assembler with respect to insert length of the sequencing library, minimum coverage of the target genome with PE reads and other recommendations. If scaffolds are inputted, it is usually beneficial to fill them with a gap filling algorithm prior to processing them with AlignGraph (*e.g.* Boetzer and Pirovano 2012). Another requirement for AlignGraph is the availability of a closely related reference genome sequence. Nearly complete reference genomes of high quality will yield the best results, but partially sequenced genomes can be used as well. Based on our experience, AlignGraph can make solid improvements when at least 50% of the PE reads can be aligned to the reference genome using the alignment protocol outlined below.

*(i) Alignment maps.* In the initial preprocessing step of AlignGraph, the PE reads, used for the *de novo* assembly in the *Prerequisite* section, are aligned to the contigs and to the reference genome, and the contigs are also aligned to the reference genome. Aligning the reads to the contigs simplifies their alignments to the reference by guiding them with the much longer contigs as backbone (see below). Generating reliable alignments among the PE reads and the contigs is relatively straightforward, because both are from the same genetic background, thus requiring a low level of variant tolerance in the alignments. Aligning the contigs to the reference genome demands a higher level of variant tolerance. However, due to the relatively large length of the contigs, their alignments to the reference can also be done reliably, as long as the evolutionary distance between the target and reference genome is

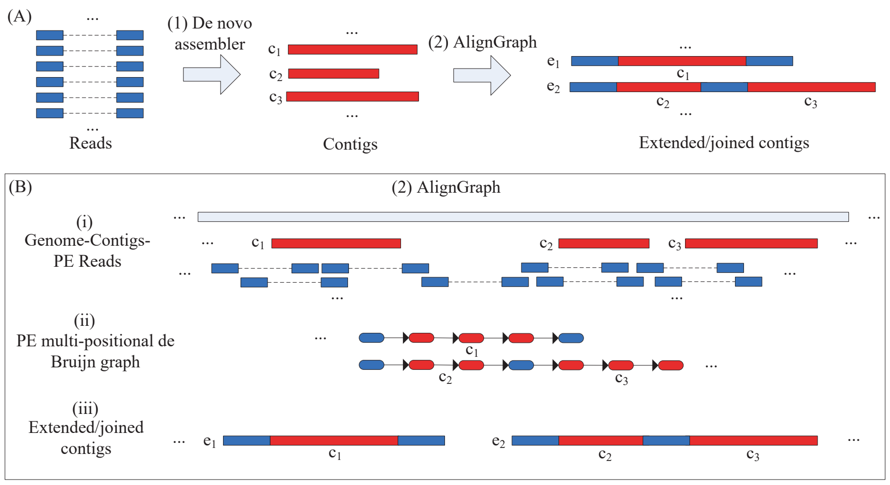

Fig. 1: *Overview of the AlignGraph algorithm.* The outline on the top (A) shows AlignGraph in the context of common genome assembly workflows, and the one on the bottom (B) illustrates its three main processing steps. (A) In step 1, the PE reads from a target genome are assembled by a *de novo* assembler into contigs (here $c_1$, $c_2$ and $c_3$). Subsequently (step 2), the contigs can be extended (blue) and joined by AlignGraph ($e_1$ and $e_2$). (B) The workflow of AlignGraph consists of three main steps. (i) the PE reads are aligned to the reference genome and to the contigs, and the contigs are also aligned to the reference genome. (ii) the PE multi-positional de Bruijn graph is built from the alignment results, where the red and blue subpaths correspond to the aligned contigs and sequences from PE reads, respectively. (iii) the extended and/or joined contigs (here $e_1$ and $e_2$) are generated by traversing the graph.

not too large. The current implementation of AlignGraph uses Bowtie2 and BLAT for these two alignment steps, respectively (Langmead and Salzberg 2012; Kent 2002). In contrast to this, aligning the relatively short PE reads to the reference genome is a much more challenging task, due to the difficulty of generating reliable short alignments containing larger numbers of mismatches and gaps. This problem does not apply to the reads aligning to the contigs since their alignment positions to the reference genome can be inferred from the more robust contig alignments. For the PE read to reference genome alignment, it is important to choose a highly variant tolerant short read aligner that is able to reliably align most of the short reads to their *true* source locations in the reference genome while minimizing the number of false positive read placements. Clearly, the latter would negatively impact the precision performance of AlignGraph by leading to chimeric joins in the downstream contig extension steps. Although a wide range of short read aligners has been developed over the past years (Li and Homer 2010), none of them has been specifically designed or optimized for aligning short reads against reference genomes with sequence differences more pronounced than those observed among genomes within the same species. To minimize the above challenges, we have chosen for this critical step the highly tunable Bowtie2 aligner with parameter settings that we optimized for aligning PE reads from a target genome to a reference genome sharing variable degrees of sequence similarity. The use of PE read alignments in this step is also important, because the additional sequence information, provided by the second read in a PE, increases the specificity of the alignment process compared to single end reads, and thus reduces the number of false read placements. To account for rearrangements among the two genomes, we use for the alignments of the PE reads against the reference genome more relaxed insert length variation settings than in the alignments against the contigs (details are below).

*(ii) Contig reassembly with PE multi-positional de Bruijn graph.* The core functionality of AlignGraph is the extension of the contigs by re-assembling them using the alignment results obtained in the previous step. To achieve this efficiently, we build from the alignment maps a variant of the de Bruijn graph, here called the *paired-end multi-positional de Bruijn graph*. This method combines the PE de Bruijn graph (Medvedev *et al.* 2011) and the positional de Bruijn graph (Ronen *et al.* 2012) where we incorporate both PE read information and alignment positions into the graph (Pevzner *et al.* 2001). The former was designed to generate more complete contigs in *de novo* assemblies, and the latter to correct contig errors in secondary assemblies. Our approach solves several problems in improving assembly results that we briefly discussed in the Introduction (see also Table 1). The following describes our modified de Bruijn graph in more details, where we first introduce important concepts of conventional de Bruijn graph-based assembly methods.

**Table 1.** *Problems the PE multi-positional de Bruijn graph solves in comparison to the conventional de Bruijn graph.*

| Problem | Consequence | Solution |
|---|---|---|
| Repeat sequences | Branched paths | Distinguishes paths for repetitive regions by incorporating PE read and alignment position information |
| Sequencing errors | False positive paths | Corrects paths from erroneous reads with correct reads aligned to the same position |
| Low sequencing depth | Incomplete paths | Builds paths from reads of low sequencing depth with reference support |

*Background*
The most widely used method for genome assemblies from short reads is the de Bruijn graph method (Pevzner *et al.* 2001). A de Bruijn graph is a directed graph: two connected nodes represent $k + 1$ bases where the first node represents the first $k$ bases and the second node the second $k$ bases (called *k-mer*). To construct a de Bruijn graph, $l - k + 1$ connected nodes are constructed from each read of length $l$ and two nodes from different reads are joined if they share the same k-mers. In theory, this graph contains all information required to reconstruct the full sequence of the underlying genome by traversing it properly. However, such an ideal result

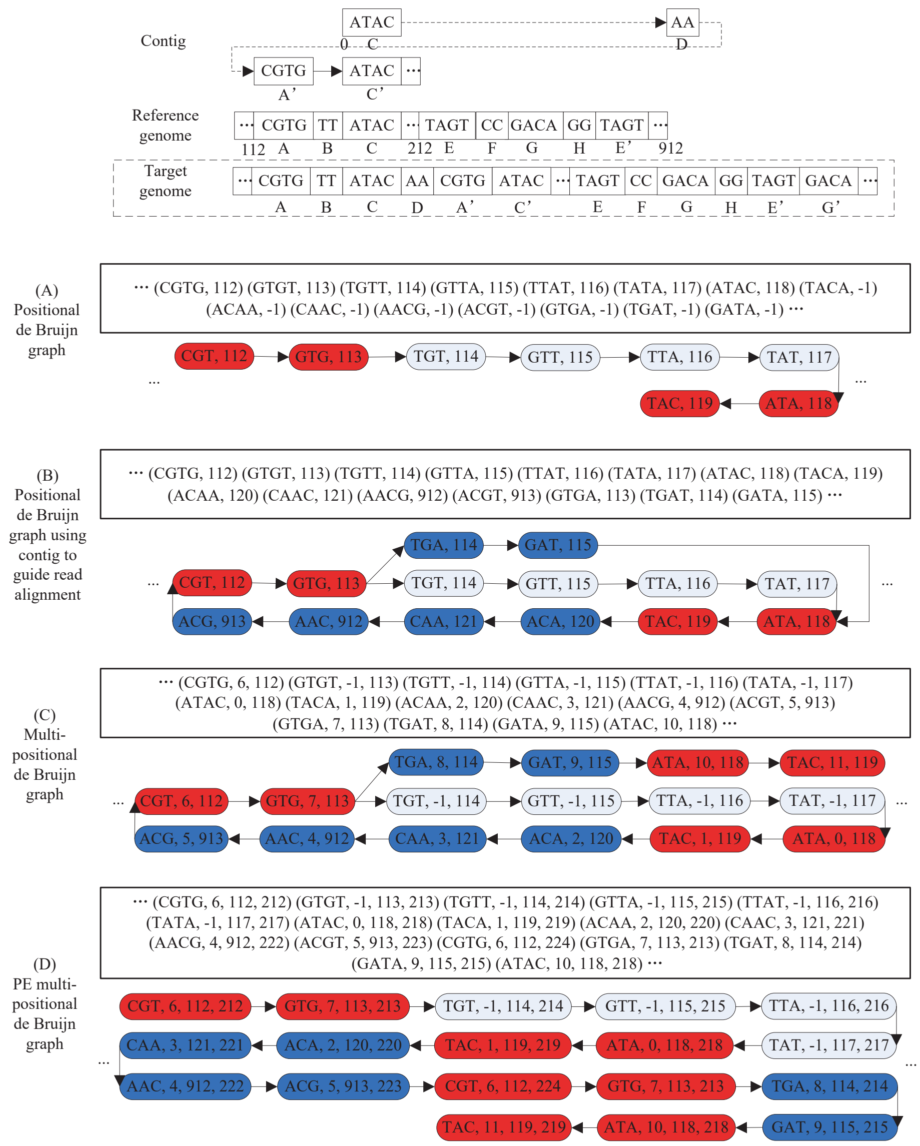
Contig
ATAC
0 C
AA
D
CGTG
A'
ATAC
C'
Reference genome
··· CGTG TT ATAC ··· TAGT CC GACA GG TAGT ···
112 A B C 212 E F G H E' 912
Target genome
··· CGTG TT ATAC AA CGTG ATAC ··· TAGT CC GACA GG TAGT GACA ···
A B C D A' C' E F G H E' G'
(A) Positional de Bruijn graph
··· (CGTG, 112) (GTGT, 113) (TGTT, 114) (GTTA, 115) (TTAT, 116) (TATA, 117) (ATAC, 118) (TACA, -1) (ACAA, -1) (CAAC, -1) (AACG, -1) (ACGT, -1) (GTGA, -1) (TGAT, -1) (GATA, -1) ···
CGT, 112
GTG, 113
TGT, 114
GTT, 115
TTA, 116
TAT, 117
ATA, 118
TAC, 119
(B) Positional de Bruijn graph using contig to guide read alignment
··· (CGTG, 112) (GTGT, 113) (TGTT, 114) (GTTA, 115) (TTAT, 116) (TATA, 117) (ATAC, 118) (TACA, 119) (ACAA, 120) (CAAC, 121) (AACG, 912) (ACGT, 913) (GTGA, 113) (TGAT, 114) (GATA, 115) ···
TGA, 114
GAT, 115
CGT, 112
GTG, 113
TGT, 114
GTT, 115
TTA, 116
TAT, 117
ACG, 913
AAC, 912
CAA, 121
ACA, 120
TAC, 119
ATA, 118
(C) Multi-positional de Bruijn graph
··· (CGTG, 6, 112) (GTGT, -1, 113) (TGTT, -1, 114) (GTTA, -1, 115) (TTAT, -1, 116) (TATA, -1, 117) (ATAC, 0, 118) (TACA, 1, 119) (ACAA, 2, 120) (CAAC, 3, 121) (AACG, 4, 912) (ACGT, 5, 913) (GTGA, 7, 113) (TGAT, 8, 114) (GATA, 9, 115) (ATAC, 10, 118) ···
TGA, 8, 114
GAT, 9, 115
ATA, 10, 118
TAC, 11, 119
CGT, 6, 112
GTG, 7, 113
TGT, -1, 114
GTT, -1, 115
TTA, -1, 116
TAT, -1, 117
ACG, 5, 913
AAC, 4, 912
CAA, 3, 121
ACA, 2, 120
TAC, 1, 119
ATA, 0, 118
(D) PE multi-positional de Bruijn graph
··· (CGTG, 6, 112, 212) (GTGT, -1, 113, 213) (TGTT, -1, 114, 214) (GTTA, -1, 115, 215) (TTAT, -1, 116, 216) (TATA, -1, 117, 217) (ATAC, 0, 118, 218) (TACA, 1, 119, 219) (ACAA, 2, 120, 220) (CAAC, 3, 121, 221) (AACG, 4, 912, 222) (ACGT, 5, 913, 223) (CGTG, 6, 112, 224) (GTGA, 7, 113, 213) (TGAT, 8, 114, 214) (GATA, 9, 115, 215) (ATAC, 10, 118, 218) ···
CGT, 6, 112, 212
GTG, 7, 113, 213
TGT, -1, 114, 214
GTT, -1, 115, 215
TTA, -1, 116, 216
CAA, 3, 121, 221
ACA, 2, 120, 220
TAC, 1, 119, 219
ATA, 0, 118, 218
TAT, -1, 117, 217
AAC, 4, 912, 222
ACG, 5, 913, 223
CGT, 6, 112, 224
GTG, 7, 113, 213
TGA, 8, 114, 214
TAC, 11, 119, 219
ATA, 10, 118, 218
GAT, 9, 115, 215

Fig. 2: *Advantages of the PE multi-positional de Bruijn graph compared to the positional de Bruijn graph.* In the target genome given on the top $A$ and $A'$, $C$ and $C'$, $E$ and $E'$, $G$ and $G'$ are repetitive regions. Each PE read of length 2 × 4bp is sequenced with one pair from region $ABCDA'C'$ and the other from the corresponding position of region $EFGHE'G'$ (the pair from $EFGHE'G'$ is omitted for simplicity). In comparison to the target genome, the reference genome has a repeat-free region $ABC$ similar to $ABCDA'C'$ and a region $EFGHE'$ similar to $EFGHE'G'$. The reads from region $ABCDA'C'$ are assembled with a *de novo* assembler into a contig starting from $CDA'C'$, but regions $A$ and $B$ are not assembled due to low sequencing depth, repeats or other problems. When aligning the contig to the reference genome, the repetitive regions $C$ and $C'$ are both aligned to $C$ in the reference genome and the insertion $D$ is assigned to the end of the reference. In (A) reads are aligned directly to the reference genome to build the initial positional de Bruijn graph; and in (B)-(D) the reads are aligned to the pre-assembled contigs and then aligned to the reference to build first the extended positional de Bruijn graph and then the PE multi-positional de Bruijn graph. (A) The initial positional de Bruijn graph is built here with 3-mers. Some reads cannot be aligned to the reference genome due to sequence differences in the target genome as indicated here by 3-mers with -1 as alignment position. The repetitive regions $A$ and $A'$ (or $C$ and $C'$) are collapsed into one path in red in the graph. (B) The initial positional de Bruijn graph is constructed with help from the read-to-contig alignment information. The read-to-reference genome alignment information yields a more complete positional de Bruijn graph, but the repetitive regions $A$ and $A'$ (or $C$ and $C'$) are still collapsed resulting in branch points. (C) An extended positional de Bruijn graph is built by incorporating into each 3-mer the read alignment position to the contig. As a result of this operation, the repetitive regions $C$ and $C'$ can be distinguished into two paths where the 3-mers have different alignment positions in the contig, but $A$ and $A'$ are still collapsed. (D) The PE multi-positional de Bruijn graph is constructed by incorporating into each 3-mer their PE read alignment positions to the reference genome (the right 3 bases and its alignment position to the contig is omitted here). With this information the repeats $A$ and $A'$ can be distinguished into two paths as the 3-mers have different PE alignment positions in the reference genome. The final graph contains only one single path allowing to output an extended contig corresponding to the region $ABCDA'C'$ in the target genome.

is usually hard to obtain, because the de Bruijn graph frequently contains many false positive, incomplete and branched paths, especially when the read quality is low or the target genome is repeat rich. The false positive and incomplete paths are due to false positive k-mers with sequencing errors and missing k-mers from regions of low sequencing depth, respectively. The branched paths are caused by joins of k-mers from repetitive regions. Several variations of the de Bruijn graph have been proposed to solve these limitations, especially the branched paths, while preserving all of its genome information (Medvedev *et al.* 2011; Ronen *et al.* 2012; Peng *et al.* 2010). The paired-end de Bruijn graph (Medvedev *et al.* 2011) is built from PE reads, where each k-mer contains $k$ bases from the left pair plus its corresponding $k$ bases from the right pair. In contrast to this, the positional de Bruijn graph (Ronen *et al.* 2012) incorporates read alignment information by including in each k-mer the $k$ bases plus its alignment position. With the additional information assigned to the k-mers, k-mers from repetitive regions can often be distinguished, and thus the number of branches in the graph can be reduced. In addition, because the positional de Bruijn graph is built from read alignments, false positive and incomplete paths can be largely avoided. We emphasize that the PE de Bruijn graph requires the left pair forward-strand read and the right pair reverse-strand read or vice versa, but it is difficult to know their orientation. This problem can be resolved, if the PE de Bruijn graph is built from aligned reads, where their orientation can be obtained from the alignments.

*PE multi-positional de Bruijn graph*
We derive the *paired-end multi-positional de Bruijn graph* as a combination of the PE de Bruijn graph and the positional de Bruijn graph. Each k-mer of the PE multi-positional de Bruijn graph is composed of three left/right element pairs: the $k$ bases of each the left and the right read pair (called *left or right $k$ bases*), the alignment position of each the left and the right $k$ bases to the contigs, and the alignment position of each the left and the right $k$ bases to the reference genome. Two k-mers can be joined if they have similar $k$ bases and close alignment positions within the constraints defined in the formulas below. Formally, let $s$ be the $k$ bases from the left read pair and $s'$ the corresponding $k$ bases from the right read pair, then the k-mer of PE multi-positional de Bruijn graph is a 6-tuple ($s$, $s'$, $c$, $g$, $c'$, $g'$), where $c$ is the alignment position of $s$ to the contigs, $g$ is the alignment position of $s$ to the reference genome, $c'$ is the alignment position of $s'$ to the contigs, and $g'$ is the alignment position of $s'$ to the reference genome. Two k-mers ($s_i$, $s'_i$, $c_i$, $g_i$, $c'_i$, $g'_i$) and ($s_j$, $s'_i$, $c_j$, $g_j$, $c'_j$, $g'_j$) can be joined if constrains (1)-(6) are met:

$$\text{mismatch}(s_i, s_j) < \delta \tag{1}$$

$$\text{mismatch}(s'_i, s'_j) < \delta \tag{2}$$

$$|c_i - c_j| < \epsilon \text{ or } c_i = -1 \text{ or } c_j = -1 \tag{3}$$

$$|g_i - g_j| < \epsilon \tag{4}$$

$$|c'_i - c'_j| < \epsilon + 2D \text{ or } c'_i = -1 \text{ or } c'_j = -1 \tag{5}$$

$$|g'_i - g'_j| < \epsilon + 2D \tag{6}$$

where $\delta$ and $\epsilon$ are small numbers with the default values 5 and 25, respectively, and $D$ is the variability of the insert length $I$ of the PE reads. The variability $D$ is equal to $\max\{I_u - I, I - I_l\}$ where $I_u$ and $I_l$ are the upper and lower limits of $I$, respectively. The variables in the above formulas are explained below.

$\delta$: To join two k-mers and tolerate sequencing errors, we allow a small number of mismatches $\delta$ between $s_i$ and $s_j$ and between $s'_i$ and $s'_j$ in (1) and (2), respectively.

$\epsilon$: We allow a small shift $\epsilon$ between each pair of alignment positions in (3)-(6), because the same $k$ bases $s_i$ and $s_j$ (or $s'_i$ and $s'_j$) from different reads may align to different but close positions in the contigs or genome as discussed in Ronen *et al.* (2012).

$2D$: We allow a shift $2D$ of $s'_i$ and $s'_j$'s alignment positions to the contigs in (5) and to the reference genome in (6). The maximum and minimum alignment distances between a read pair are $I - l + D$ and $I - l - D$, respectively, where $l$ is the read length, assuming the same read length for both members in a pair. Thus, the maximum alignment distance of two right reads with left reads aligned at the same position is $(I - l + D) - (I - l - D) = 2D$. This distance is equal to the distance between any two k-mers from the same position in the right read pairs, so the maximum distance between $s'_i$ and $s'_j$ will be $2D$.

$-1$: $s_i$ and $s_j$ (or $s'_i$ and $s'_j$) can be joined if one or both of them are aligned directly to the reference genome rather than guided by the *de novo* contigs. In those cases, we assign -1 as alignment position to the contigs. This is important because we allow contig extensions only if the alignable and unalignable bases to contigs can be joined.

It is important to guarantee that each k-mer corresponding to an insertion of a read alignment has a position in the reference genome. To achieve this, we append the inserted k-mer to the end of the genome sequence. In our implementation of the PE multi-positional de Bruijn graph, we first load iteratively sections of the reference genome into memory. Then we perform the following operation. We test for each k-mer in each aligned read at genome position $g$, if there is already a k-mer at $g$ and whether the new k-mer can be joined with it. If so then we join the two k-mers; otherwise we attach the new k-mer to position $g$. The connection between two k-mers is recorded by using pointers and the read coverage for each k-mer is stored along with it. Figure 2 illustrates the main advantages of the PE multi-positional de Bruijn graph compared to the positional de Bruijn graph with several examples (see also Table 1). This includes the contig-guided PE read alignment against the reference genome resulting in a larger number of alignable reads, and thus a more complete de Bruijn graph (Figure 2B); as well as the reduction of branched paths in the graph by distinguishing reads from different repetitive regions (Figures 2C and 2D). For space reasons, the advantages over the conventional de Bruijn graph in reducing false positive and incomplete paths are not shown.

*(iii) Graph traversal returns extended contigs.* To remove errors, the de Bruijn graph needs to be edited prior to its traversal. The three major types of errors are tips, bubbles and erroneous connections (Zerbino and Birney 2008; Chaisson and Pevzner 2008; Peng *et al.* 2010). Most of them are caused by errors in the reads. A tip is a short path with a dead end, while a bubble consists of two short paths sharing the same start and end nodes. The formation of tips and bubbles is relatively rare, mainly because k-mers with $< \delta$ mismatches are joined. The remaining errors can be removed by applying a coverage cut-off filter similar to the strategies employed by most *de novo* assemblers. Due to the additional information encoded in the modified de Bruijn graph, one can use here a relatively small coverage threshold. After these error removal steps, the PE multi-positional de Bruijn graph is traversed, using a broad-first strategy, to generate the final contigs. Each traversal stops at a branch position and an extended contig is returned. After returning the extended contigs, the remaining unextended contigs (identical with initial *de novo* contigs) are provided to the user in a separate file. Finally, contigs with sufficient PE read connections and a path between them can be joined. Occasionally, those connections can be missed by the above filtering step because of too low read coverage in local areas of the connecting path.

## 2.2 Software implementation

AlignGraph is implemented in C++ for Linux operating systems. Its expected input includes the PE reads, the pre-assembled *de novo* contigs, and the reference genome. Its output includes the extended contigs as well as the remaining non-extended contigs. AlignGraph runs the alignment steps with BLAT and Bowtie2 automatically, but both need to be installed on a system. AlignGraph's run time is currently 23-48 minutes per million aligned reads and its memory usage stays below 34-45 GB even for very large read sets and genomes. These requirements are much more moderate than those of most *de novo* assemblers (Luo *et al.* 2012).

## 3 EVALUATION

### 3.1 Experimental design

*Background.* To evaluate AlignGraph's efficiency in improving genome assemblies, we performed a series of systematic performance tests. For this, we assembled publicly available genomic PE read sets from two organisms of variable genome complexity with six widely used *de novo* assemblers, extended the resulting contigs with AlignGraph, and then evaluated the improvements with a set of standard metrics for comparing assembly results (Table 2). In these tests it was important to choose the NGS read samples from organisms where the genome sequence of both the target genome and a close reference genome are known. This way one can evaluate the completeness and correctness of the results against a true result rather than one that is unknown or only partially known. To also assure the improvements obtained by AlignGraph, are not simply the result of insufficient optimizations of the upstream *de novo* assembly, we included in some cases pre-assembled contig and scaffold sets that are widely accepted by the community as benchmark data sets for evaluating assembly software. Today's requirements for assembling genomes from NGS were met by choosing read samples with $\geq$75bp and paired-end read information. In total we performed assembly tests on the following three sample sets.

*3.1.1 A. thaliana sample.* The first sample set was from the model organism *A. thaliana*, which is a flowering plant with a compact genome of 130Mb in size. The PE read set, chosen for this test, is from a genomic Illumina NGS library with a read length of 2x 75 bp. As de novo assemblers, we included in this test Velvet and ABySS, which we chose here as software representatives performing well on single library data, and because of their good sensitivity and precision performance (Lin *et al.* 2011). The VelvetOptimiser tool was used to optimize the parameter settings for the Velvet assembly. ABySS was run with the same k-mer length as Velvet, while the remaining parameters were set to their defaults. To extend the preassembled contigs with AlignGraph, we used in one test the *A. thaliana* target genome as ideal reference, and in another test we used the publicly available genome sequence from the related *A. lyrata* species as reference (Table 12a). The latter was chosen, because it constitutes a more challenging reference genome for testing AlignGraph's performance in improving genome assemblies than the references used in the other tests below. This is the case for the following reasons (Hu *et al.* 2011): *A. lyrata* and *A. thaliana* diverged over 10 million years ago; their genomes differ by many regional rearrangements; the sequence similarity in the common regions of their genomes is only 80%; and the *A. lyrata* genome sequence is still incomplete and fragmented into many scaffolds.

*3.1.2 Human sample from GAGE.* The second sample set is from the community project GAGE (Genome Assembly Gold-Standard Evaluations), from which we selected the human chromosome 14 sample (Salzberg *et al.* 2012). Its Illumina sequences consists of PE reads with a length of 76-101 bp from three different libraries. To assure in this test a high quality of the initial *de novo* contigs, we did not assemble them ourself. Instead, we downloaded the pre-assembled contig sets provided by the GAGE project for the four assemblers that ranked highest in the benchmark tests by Salzberg *et al.* (2012) in assemblies from multiple genome libraries with variable insert lengths. Those included ALLPATHS-LG, SOAPdenovo, MaSuRCA and CABOG. As reference sequence for guiding AlignGraph, we used in these tests the chimpanzee genome. Only for ALLPATH-LG in its *cheat* mode, we reassembled the contigs ourselves, because this assembler exhibits a better sensitivity and precision performance when providing a closely related reference genome. Here it was important to compare the performance of ALLPATHS-LG with AlignGraph when both are guided by the same reference genome.

In addition to contigs, we evaluated AlignGraph's performance in improving the scaffold sets provided by the GAGE project for the same human sample set. Prior to their reassembly with AlignGraph, we reduced the number of unresolved sequence regions (gaps filled with ambiguous N bases) in the scaffolds by applying the GapFiller algorithm, which is currently one of the most efficient gap filling methods (Boetzer and Pirovano 2012).

*3.1.3 Published genome.* In addition to the tests above, we were interested in evaluating to what extent AlignGraph can improve the genome sequence generated with another reference assisted assembly approach. For this test, we chose the published genome sequence from Landsberg *erecta* (L*er*-1; Schneeberger *et al.* 2011). The latter is a strain of *A. thaliana* with too severe differences in its genome to resolve its sequence with a simple resequencing approach alone where the *A. thaliana* genome could serve as reference. Thus, Schneeberger *et al.* (2011) assembled its genome with a reference assisted pipeline approach that included ALLPATHS-LG and several refinement steps.

*3.1.4 Data sources.* The genome sequences used in the above tests were downloaded from the following community sites: *A. thaliana* from TAIR, *A. lyrata* from JGI, Landsberg *erecta* from 1001 Genomes, and human and chimpanzee from Ensembl. From the GAGE site, we downloaded the PE read sets, and the pre-assembled contigs and scaffolds for the human chromosome 14 sample (Salzberg *et al.* 2012). The PE read sets from *A. thaliana* and Landsberg *erecta* were downloaded from NCBI's Sequence Read Archive (SRA) and the 1001 Genome site, respectively. The *A. thaliana* read set contained 32 million 2 $\times$ 75 bp PE reads (accession: SRR073127), the human read set contained 61 million 2 $\times$ 75-101 bp PE reads, and the L*er*-1 read set contained 65 million 2 $\times$ 101 bp PE reads.

*3.1.5 Performance Measurements.* Most of the performance measures used by this study are adapted from the GAGE project (Salzberg *et al.* 2012). To evaluate the completeness of the contigs, we aligned them to the target genome with BLAT. If a contig could not be aligned as a single global alignment, then it was split according to the local alignment results into the smallest possible number of sub-contigs. The resulting contigs are called *true contigs*. The precision measures include the number of misassemblies per million base pairs (MPMB) and the average identity between contigs and target genome. There are two types of misassemblies: misjoin errors and unjoin errors. Misjoin errors result in chimeric contigs. Their number can be calculated as the

**Table 2.** *Performance Evaluation of AlignGraph.* (a) Genomic PE reads from *A. thaliana* were assembled with Velvet and ABySS. The resulting contigs were extended with AlignGraph using as reference the genome sequence from *A. lyrata*, and as ideal reference genome the one from *A. thaliana* (*A.th.*). (b-d) The subsequent panels contain assembly results for the human chromosome 14 sample from the GAGE project where the chimpanzee genome served as reference. (b) Contig assembly results are given for the *de novo* assemblers ALLPATHS-LG, ALLPATHS-LGc (in cheat mode), SOAPdenovo, MaSuRCA and CABOG. (c) Scaffolded assembly results are given for SOAPdenovo, MaSuRCA and CABOG. The results are organized row-wise as follows: the number of initial contigs obtained by each *de novo* assembler[1], the 'extendable' subset of the initial contigs that AlignGraph was able to improve[2], and the extension results obtained with AlignGraph[3]. The additional columns give the number of contigs[4], N50 values[5], the length coverage of the genome by contigs[6], the average[7] and maximum[8] length of the contigs, the number of misassemblies per million base pairs (MPMB)[9], and the average identity among the true contigs and the target genome[10]. More details on these performance criteria are provided in the Performance Measurements section.

| Upstream assembler | Contig set | N Contigs[4] | N50[5] | Coverage[6] | Average length[7] | Maximum length[8] | MPMB[9] | Average identity[10] |
|---|---|---|---|---|---|---|---|---|
| | *(a) Contigs of A. thaliana genome* | | | | | | | |
| Velvet | All[1] | 30,037 | 3,536 | 82,399,610 | 2,817 | 27,792 | 383.1 | 95.4% |
| | Extendable[2] | 7,839 | 4,138 | 25,356,693 | 3,247 | 27,398 | 310.0 | 97.6% |
| | Extendable + AlignGraph[3] | 5,319 | 7,720 | 29,673,314 | 5,474 | 49,623 | 186.5 | 91.7% |
| | Extendable (*A.th.*) | 23,852 | 3,608 | 68,367,186 | 2,888 | 27,792 | 350.2 | 96.8% |
| | Extendable + AlignGraph (*A.th.*) | 15,227 | 9,820 | 88,043,266 | 5,869 | 89,056 | 177.8 | 91.5% |
| ABySS | All | 30,972 | 2,567 | 69,337,135 | 2,263 | 29,760 | 463.5 | 97.3% |
| | Extendable | 10,643 | 2,794 | 26,082,265 | 2,435 | 16,343 | 412.6 | 98.8% |
| | Extendable + AlignGraph | 7,823 | 5,369 | 32,905,260 | 4,094 | 25,353 | 247.6 | 92.2% |
| | Extendable (*A.th.*) | 24,410 | 2,608 | 56,371,241 | 2,299 | 29,760 | 438.1 | 98.4% |
| | Extendable + AlignGraph (*A.th.*) | 18,559 | 6,466 | 81,540,651 | 4,389 | 77,823 | 236.5 | 91.7% |
| | *(b) Contigs of Human chromosome 14* | | | | | | | |
| ALLPATHS-LG | All | 4,383 | 38,590 | 83,847,514 | 19,249 | 240,764 | 53.1 | 98.9% |
| | Extendable | 1,636 | 38,699 | 33,993,990 | 20,216 | 200,495 | 50.3 | 98.8% |
| | Extendable + AlignGraph | 778 | 73,858 | 34,604,800 | 43,510 | 304,548 | 23.6 | 96.6% |
| ALLPATHS-LGc | All | 3,856 | 43,856 | 83,858,469 | 21,857 | 275,446 | 46.5 | 99.3% |
| | Extendable | 1,318 | 45,288 | 30,605,770 | 23,319 | 275,446 | 43.4 | 99.5% |
| | Extendable + AlignGraph | 640 | 82,046 | 33,514,214 | 50,484 | 391,100 | 20.2 | 96.2% |
| SOAPdenovo | All | 10,865 | 16,867 | 80,114,725 | 7,823 | 147,494 | 135.1 | 94.9% |
| | Extendable | 5,592 | 17,581 | 45,725,655 | 8,263 | 141,981 | 124.3 | 96.3% |
| | Extendable + AlignGraph | 3,442 | 33,311 | 53,069,539 | 15,415 | 221,608 | 66.8 | 93.9% |
| MaSuRCA | All | 19,034 | 5,768 | 75,491,835 | 3,869 | 53,837 | 280.7 | 98.9% |
| | Extendable | 9,479 | 6,096 | 39,836,321 | 4,171 | 51,249 | 255.4 | 99.2% |
| | Extendable + AlignGraph | 5,661 | 11,704 | 44,821,180 | 7,787 | 69,327 | 130.8 | 96.7% |
| CABOG | All | 3,118 | 46,523 | 84,988,860 | 27,472 | 296,888 | 37.1 | 97.3% |
| | Extendable | 1,665 | 45,669 | 46,089,218 | 27,301 | 296,888 | 36.9 | 98.7% |
| | Extendable + AlignGraph | 684 | 104,171 | 49,215,105 | 71,466 | 344,910 | 14.5 | 96.6% |
| | *(c) Scaffolds of Human chromosome 14* | | | | | | | |
| SOAPdenovo | All | 3,902 | 391,693 | 85,414,648 | 24,839 | 1,852,152 | 40.8 | 82.9% |
| | Extendable | 881 | 379,370 | 32,599,249 | 38,800 | 1,019,659 | 26.0 | 84.2% |
| | Extendable + AlignGraph | 748 | 528,039 | 39,195,931 | 53,744 | 1,907,306 | 19.2 | 80.6% |
| MaSuRCA | All | 721 | 584,807 | 65,429,147 | 68,083 | 2,943,966 | 11.1 | 57.2% |
| | Extendable | 122 | 255,926 | 4,865,797 | 40,830 | 520,370 | 27.5 | 89.4% |
| | Extendable + AlignGraph | 90 | 303,139 | 7,060,400 | 77,917 | 1,263,099 | 14.0 | 87.5% |
| CABOG | All | 471 | 387,876 | 81,163,384 | 176,980 | 1,944,475 | 5.8 | 91.9% |
| | Extendable | 131 | 327,656 | 24,909,270 | 189,898 | 1,905,529 | 5.3 | 98.0% |
| | Extendable + AlignGraph | 62 | 840,450 | 30,407,969 | 464,475 | 2,043,930 | 2.3 | 90.7% |

number of splits necessary to obtain the true contigs. Unjoin errors result in incomplete contigs that can be approximated by the number of contigs before the splits. Thus, MPMB $= \frac{e_m+e_u}{L} \times 10^6$, where $e_m$ and $e_u$ are the numbers of misjoin errros and unjoin errors, respectively, and $L$ is the accumulative length of the contigs. The average identity between true contigs and the target genome is calculated as $\frac{\sum_n t_i \times l_i}{\sum_n l_i}$ where $t_i$ is the identity for contig $i$ and $l_i$ is the length of contig $i$ ($0 < i \leq n$). In this formula, the identity $t_i$ of the true contigs $i$ is calculated as the number of aligned bases over the length of the alignment. The sensitivity measures include the N50 value and the coverage of the true contigs. The former is the contig size at 50% of the total number of contig bases, and the latter is the total number of genome bases covered by the contigs. Two additional measures are the average length and maximum length of the true contigs.

## 3.2 Results

*3.2.1 Extension of A. thaliana contigs.* The performance test results for the *A. thaliana* data set are given in Table 2a. In comparison to the initial contig sets assembled by Velvet or ABySS, AlignGraph extends 26.1-34.4% of them when it is guided with the *A. lyrata* genome as reference. The resulting set of extended contigs contains 26.5-32.1% less sequences, because AlignGraph has joined many of the initial contigs. This leads to improvements of the N50, coverage, average contig length, maximum contig length and MPMB values for the extendable contig set by 86.6-92.2%, 17.0-26.2%, 68.1-68.6%, 55.1-81.1% and 39.8-40.0%, respectively. As expected the average identity drops slightly (5.9-6.6%), because with increased length of the assembled sequences, internal sequence variations accumulate and complicate the alignment of the extended contigs against the target genome. A similar trend can be seen in the below results for the much longer scaffolds where the average identity is always lower for all of the tested assemblers (Table 2c). Overall the assembly results of AlignGraph contain for all three sample sets (3.2.1-3.2.3) a comparable number of sequence variations to the target genomes as the results of *de novo* assemblers (data not shown). This indicates a high sequence quality of the reassembled contigs. When guiding AlignGraph with the ideal reference genome (*A. thaliana*), 78.8-79.4% of the initial contigs can be extended and the extension results contain 24.0-36.2% less contigs. The corresponding improvements of the N50, coverage, average contig length, maximum contig length and MPMB values are 147.9-172.2%, 28.8-44.6%, 90.9-103.2%, 161.5-220.4% and 46.0-49.2%, respectively. These results indicate that AlignGraph is able to substantially improve the assemblies for the *de novo* assemblers Velvet and ABySS. The usage of a perfect reference genome improves the results in a more pronounced manner. However, even when a suboptimal and evolutionary distinct reference genome is used, as the one from *A. lyrata* (see 3.1.1), AlignGraph can lead to considerable improvements.

*3.2.2 Extension of human contigs and scaffolds from GAGE.* The test results for the human chromosome 14 contigs are given in Table 2b. Of the contigs assembled by ALLPATHS-LG, 37.3% of them can be extended and the extension result contains 52.4% less contigs due to the joins generated by AlignGraph. These improvements are more pronounced than in the above experiment with *A. lyrata* as reference, because the genomes of human and chimpanzee share a much higher sequence similarity than the genomes of *A. thaliana* and *A. lyrata*. The N50, coverage, average contig length, maximum contig length and MPMB values for the extendable contig set consistently improve by 90.9%, 1.8%, 115.2%, 51.9% and 53.1%, respectively. Similar results could be obtained with the other *de novo* assemblers SOAPdenovo, MaSuRCA and CABOG. After AlignGraph processing their extendable contigs improved for the same five evaluation metrics by 89.5-128.1%, 6.8-16.1%, 86.6-161.8%, 16.2-56.1% and 46.3-60.7%, respectively. If the ALLPATHS-LG is run in its cheat mode by guiding it with the same reference genome as AlignGraph, then both the sensitivity and precision measures of the ALLPATHS-LGc contigs improve compared to the assembly without a reference. Nevertheless, AlignGraph is still able to extend 34.2% of the ALLPATHS-LGc contigs and the extension results contain 51.4% less contigs, while the five evaluation metrics improve by 81.2%, 9.5%, 116.5%, 42.0% and 53.5%, respectively. These improvements indicate that the reference-assisted improvements provided by AlignGraph are more efficient than the ones used by ALLPATHS-LG's in its cheat mode at the contig assembly stage.

AlignGraph's performance results on the scaffolds from the same human chromosome 14 dataset are given in Table 2c. The scaffold sets from SOAPdenovo, MaSuRCA and CABOG contain much smaller numbers of sequences than their corresponding contig sets. Nevertheless, AlignGraph is able to extend 16.9-27.8% of them and the extended sets contain 15.1-52.7% fewer but longer and more complete scaffolds. This results in improvements of the above five evaluation metrics by 18.4-156.5%, 20.2-45.1%, 38.5-144.6%, 7.3-142.7% and 26.2-49.1%, respectively. The MPMB values for the scaffolds are generally smaller than for the contigs, because the scaffolds are much longer and have much fewer unjoin errors than the corresponding contigs. The scaffolds results of ALLPATHS-LG (not shown in Table 2) are more complete than those of the other *de novo* assemblers, and AlignGraph was not able to further improve them.

**Table 3.** *Improvements to Published Genome.* The published scaffolds from Landsberg *erecta* were extended with AlignGraph using the *A. thaliana* genome as reference. The rows and columns are arranged the same way as in Table 2, but several columns are missing here, because it is not possible to compute the corresponding performance measures in a meaningful manner without having access to a 'true' target genome sequence. In addition, we report here the total number of bases in the contigs[1].

| Contig set | N Contigs | N50 | N total bases[1] | Average length | Maximum length |
|---|---|---|---|---|---|
| All | 1676 | 341625 | 112,581,547 | 67,172 | 2,930,102 |
| Extendable | 549 | 388,367 | 55,611,539 | 101,296 | 1,716,958 |
| Extendable+AlignGraph | 552 | 793,894 | 64,015,527 | 115,970 | 2,103,349 |

*3.2.3 Improvements to published genome.* The test results for the published Landsberg *erecta* genome are shown in Table 3. The initial scaffold set used in this test consisted of 1,676 sequences. AlignGraph extended 32.8% of these scaffolds. The extension results contain in this case a slightly larger number of sequences, because the reference-assisted assembly pipeline, used by Schneeberger *et al.* (2011) to generated the initial genome sequence, has already established most of the possible joins. Thus, the remaining room for improvements is restricted to scaffold extensions where AlignGraph improves the N50, number of total bases, average contig length and maximum length values for the extendable scaffolds by 104.4%, 15.1%, 14.5% and 22.5%, respectively. These improvements demonstrate AlignGraph's usefulness in improving published genome sequences, even for those that have been carefully curated by their authors.

In summary, the above performance tests demonstrate AlignGraph's efficiency in improving the results of a variety of *de novo* assemblers and species with variable genome complexity by taking advantage of closely related reference genomes.

## 4 CONCLUSIONS AND FUTURE WORK

This study introduces a novel de Bruijn graph-based algorithm for improving *de novo* genome assemblies guided by sequence information from a closely related species. The chosen PE multi-positional de Bruijn graph approach provides an elegant and efficient solution to this problem. Our performance results demonstrate that the implemented AlignGraph software is able to improve the results of a wide range of *de novo* assemblers for complex genomes even with relatively diverse and suboptimal guide sequences as reference. Moreover, our results demonstrate AlignGraph's usefulness for improving unfinished genome assemblies. Another advantage is that AlignGraph can be used in combination with most existing *de novo* assemblers. In the future, we will expand AlignGraph in the following areas: (i) we will provide support for additional variant-aware alignment tools for both PE read and contig data, such as GSNAP and GMAP, respectively; (ii) *de novo* assembly functionality will be added to

AlignGraph to further optimize assemblies at many stages of the reference-assisted workflow; (iii) utilities will be incorporated for detecting and resolving misassemblies, and (iv) the processing of scaffolds with very large gaps will be improved.

## ACKNOWLEDGEMENT

We acknowledge the support of the core facilities at the Institute for Integrative Genome Biology (IIGB) at UC Riverside.
*Funding:* This work was supported by grants from the USDA National Institute for Food and Agriculture [NIFA-2010-65106-20675 to T.G.] and the National Science Foundation [ABI-0957099 to T.G., IOB-0420152 to T.G., MCB-1021969 to T.G., IIS-0711129 to T.J.].

## REFERENCES

Boetzer, M. and Pirovano, W. (2012). Toward almost closed genomes with gapfiller. *Genome biology*, **13**(6), R56.

Boetzer, M., Henkel, C. V., Jansen, H. J., Butler, D., and Pirovano, W. (2011). Scaffolding pre-assembled contigs using sspace. *Bioinformatics*, **27**(4), 578–579.

Chaisson, M. J. and Pevzner, P. A. (2008). Short read fragment assembly of bacterial genomes. *Genome research*, **18**(2), 324–330.

Chaisson, M. J., Brinza, D., and Pevzner, P. A. (2009). De novo fragment assembly with short mate-paired reads: Does the read length matter? *Genome research*, **19**(2), 336–346.

Dayarian, A., Michael, T. P., and Sengupta, A. M. (2010). Sopra: Scaffolding algorithm for paired reads via statistical optimization. *BMC bioinformatics*, **11**(1), 345.

Dohm, J. C., Lottaz, C., Borodina, T., and Himmelbauer, H. (2007). Sharcgs, a fast and highly accurate short-read assembly algorithm for de novo genomic sequencing. *Genome research*, **17**(11), 1697–1706.

Gao, S., Sung, W.-K., and Nagarajan, N. (2011). Opera: reconstructing optimal genomic scaffolds with high-throughput paired-end sequences. *Journal of Computational Biology*, **18**(11), 1681–1691.

Gnerre, S., Lander, E. S., Lindblad-Toh, K., Jaffe, D. B., *et al.* (2009). Assisted assembly: how to improve a de novo genome assembly by using related species. *Genome Biol*, **10**(8), R88.

Gnerre, S., MacCallum, I., Przybylski, D., Ribeiro, F. J., Burton, J. N., Walker, B. J., Sharpe, T., Hall, G., Shea, T. P., Sykes, S., *et al.* (2011). High-quality draft assemblies of mammalian genomes from massively parallel sequence data. *Proceedings of the National Academy of Sciences*, **108**(4), 1513–1518.

Gritsenko, A. A., Nijkamp, J. F., Reinders, M. J., and de Ridder, D. (2012). Grass: a generic algorithm for scaffolding next-generation sequencing assemblies. *Bioinformatics*, **28**(11), 1429–1437.

Hernandez, D., François, P., Farinelli, L., Østerås, M., and Schrenzel, J. (2008). De novo bacterial genome sequencing: millions of very short reads assembled on a desktop computer. *Genome research*, **18**(5), 802–809.

Hu, T. T., Pattyn, P., Bakker, E. G., Cao, J., Cheng, J. F., Clark, R. M., Fahlgren, N., Fawcett, J. A., Grimwood, J., Gundlach, H., Haberer, G., Hollister, J. D., Ossowski, S., Ottilar, R. P., Salamov, A. A., Schneeberger, K., Spannagl, M., Wang, X., Yang, L., Nasrallah, M. E., Bergelson, J., Carrington, J. C., Gaut, B. S., Schmutz, J., Mayer, K. F., Van de Peer, Y., Grigoriev, I. V., Nordborg, M., Weigel, D., and Guo, Y. L. (2011). The arabidopsis lyrata genome sequence and the basis of rapid genome size change. *Nat Genet*, **43**(5), 476–481.

Jeck, W. R., Reinhardt, J. A., Baltrus, D. A., Hickenbotham, M. T., Magrini, V., Mardis, E. R., Dangl, J. L., and Jones, C. D. (2007). Extending assembly of short dna sequences to handle error. *Bioinformatics*, **23**(21), 2942–2944.

Kent, W. (2002). Blatthe blast-like alignment tool. *Genome research*, **12**(4), 656–664.

Kim, J., Larkin, D. M., Cai, Q., Zhang, Y., Ge, R.-L., Auvil, L., Capitanu, B., Zhang, G., Lewin, H. A., Ma, J., *et al.* (2013). Reference-assisted chromosome assembly. *Proceedings of the National Academy of Sciences*, **110**(5), 1785–1790.

Langmead, B. and Salzberg, S. L. (2012). Fast gapped-read alignment with bowtie 2. *Nature methods*, **9**(4), 357–359.

Li, H. and Homer, N. (2010). A survey of sequence alignment algorithms for next-generation sequencing. *Briefings in bioinformatics*, **11**(5), 473–483.

Li, R., Zhu, H., Ruan, J., Qian, W., Fang, X., Shi, Z., Li, Y., Li, S., Shan, G., Kristiansen, K., *et al.* (2010). De novo assembly of human genomes with massively parallel short read sequencing. *Genome research*, **20**(2), 265–272.

Lin, Y., Li, J., Shen, H., Zhang, L., Papasian, C. J., *et al.* (2011). Comparative studies of de novo assembly tools for next-generation sequencing technologies. *Bioinformatics*, **27**(15), 2031–2037.

Luo, R., Liu, B., Xie, Y., Li, Z., Huang, W., Yuan, J., He, G., Chen, Y., Pan, Q., Liu, Y., *et al.* (2012). Soapdenovo2: an empirically improved memory-efficient short-read de novo assembler. *GigaScience*, **1**(1), 18.

Medvedev, P., Pham, S., Chaisson, M., Tesler, G., and Pevzner, P. (2011). Paired de bruijn graphs: a novel approach for incorporating mate pair information into genome assemblers. *Journal of Computational Biology*, **18**(11), 1625–1634.

Miller, J. R., Delcher, A. L., Koren, S., Venter, E., Walenz, B. P., Brownley, A., Johnson, J., Li, K., Mobarry, C., and Sutton, G. (2008). Aggressive assembly of pyrosequencing reads with mates. *Bioinformatics*, **24**(24), 2818–2824.

Ossowski, S., Schneeberger, K., Clark, R. M., Lanz, C., Warthmann, N., and Weigel, D. (2008). Sequencing of natural strains of Arabidopsis thaliana with short reads. *Genome Res*, **18**(12), 2024–2033.

Peng, Y., Leung, H., Yiu, S., and Chin, F. (2010). Idba–a practical iterative de bruijn graph de novo assembler. In *Research in Computational Molecular Biology*, pages 426–440. Springer.

Pevzner, P., Tang, H., and Waterman, M. (2001). An eulerian path approach to dna fragment assembly. *Proceedings of the National Academy of Sciences*, **98**(17), 9748.

Phillippy, A. M., Schatz, M. C., and Pop, M. (2008). Genome assembly forensics: finding the elusive mis-assembly. *Genome Biol*, **9**(3).

Pop, M., Phillippy, A., Delcher, A. L., and Salzberg, S. L. (2004a). Comparative genome assembly. *Briefings in bioinformatics*, **5**(3), 237–248.

Pop, M., Kosack, D. S., and Salzberg, S. L. (2004b). Hierarchical scaffolding with bambus. *Genome research*, **14**(1), 149–159.

Ronen, R., Boucher, C., Chitsaz, H., and Pevzner, P. (2012). Sequel: improving the accuracy of genome assemblies. *Bioinformatics*, **28**(12), i188–i196.

Salmela, L., Mäkinen, V., Välimäki, N., Ylinen, J., and Ukkonen, E. (2011). Fast scaffolding with small independent mixed integer programs. *Bioinformatics*, **27**(23), 3259–3265.

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., *et al.* (2012). Gage: A critical evaluation of genome assemblies and assembly algorithms. *Genome Research*, **22**(3), 557–567.

Schatz, M. C., Phillippy, A. M., Sommer, D. D., Delcher, A. L., Puiu, D., Narzisi, G., Salzberg, S. L., and Pop, M. (2013). Hawkeye and AMOS: visualizing and assessing the quality of genome assemblies. *Brief Bioinform*, **14**(2), 213–224.

Schneeberger, K., Ossowski, S., Ott, F., Klein, J. D., Wang, X., Lanz, C., Smith, L. M., Cao, J., Fitz, J., Warthmann, N., *et al.* (2011). Reference-guided assembly of four diverse arabidopsis thaliana genomes. *Proceedings of the National Academy of Sciences*, **108**(25), 10249–10254.

Simpson, J., Wong, K., Jackman, S., Schein, J., Jones, S., and Birol, İ. (2009). Abyss: a parallel assembler for short read sequence data. *Genome research*, **19**(6), 1117–1123.

Tsai, I. J., Otto, T. D., and Berriman, M. (2010). Method improving draft assemblies by iterative mapping and assembly of short reads to eliminate gaps.

Warren, R. L., Sutton, G. G., Jones, S. J., and Holt, R. A. (2007). Assembling millions of short dna sequences using ssake. *Bioinformatics*, **23**(4), 500–501.

Zerbino, D. and Birney, E. (2008). Velvet: algorithms for de novo short read assembly using de bruijn graphs. *Genome research*, **18**(5), 821–829.

Zimin, A. V., Marçais, G., Puiu, D., Roberts, M., Salzberg, S. L., and Yorke, J. A. (2013). The masurca genome assembler. *Bioinformatics*, **29**(21), 2669–2677.