BIOINFORMATICS

Vol. 26 ECCB 2010, pages i433–i439
doi:10.1093/bioinformatics/btq366

Integrating genome assemblies with MAIA

Jurgen Nijkamp[1,2,3,*], Wynand Winterbach[1,4], Marcel van den Broek[2,3], Jean-Marc Daran[2,3], Marcel Reinders[1,3,5] and Dick de Ridder[1,3,5]

[1]The Delft Bioinformatics Lab, Department of Mediamatics, Delft University of Technology, Mekelweg 4, 2628 CD Delft, [2]Industrial Microbiology Group, Department of Biotechnology, Delft University of Technology, Julianalaan 67, 2628 BC Delft, [3]Kluyver Centre for Genomics of Industrial Fermentation, P.O. Box 5057, 2600 GA Delft, [4]Network Architectures and Services, Department of Telecommunications, Delft University of Technology, Mekelweg 4, 2628 CD Delft and [5]Netherlands Bioinformatics Center, 260 NBIC, P.O. Box 9101, 6500 HB Nijmegen, The Netherlands

ABSTRACT

Motivation: *De novo* assembly of a eukaryotic genome with next-generation sequencing data is still a challenging task. Over the past few years several assemblers have been developed, often suitable for one specific type of sequencing data. The number of known genomes is expanding rapidly, therefore it becomes possible to use multiple reference genomes for assembly projects. We introduce an assembly integrator that makes use of all available data, i.e. multiple *de novo* assemblies and mappings against multiple related genomes, by optimizing a weighted combination of criteria.

Results: The developed algorithm was applied on the *de novo* sequencing of the *Saccharomyces cerevisiae* CEN.PK 113-7D strain. Using Solexa and 454 read data, two *de novo* and three comparative assemblies were constructed and subsequently integrated, yielding 29 contigs, covering more than 12 Mbp; a drastic improvement compared with the single assemblies.

Availability: MAIA is available as a Matlab package and can be downloaded from http://bioinformatics.tudelft.nl

Contact: j.f.nijkamp@tudelft.nl

1 INTRODUCTION

Next-generation sequencing (NGS) platforms, such as 454 (Roche, Branford, CT), Solid (AB, Foster City, CA) and Solexa (Illumina, San Diego, CA) allow for gigabytes of data generation at an affordable cost. The third generation sequencing platforms (Helicos, Cambridge, MA; Pacific Biosciences, Menlo Park, CA) may even let the cost per megabase drop under $1 per megabase (Shendure and Ji, 2008). Considering the relatively low cost of these platforms, compared with classical Sanger sequencing, it becomes possible to use them for *de novo* sequencing projects. However, the millions of short DNA sequences generated by NGS platforms, called *reads*, are still relatively small. Given this limited read length and the many repetitive regions in a eukaryotic genome, *de novo* assembly is still a challenging task. To alleviate this problem it is essential to design algorithms that make full use of all available data.

Over the past few years, several assemblers have been developed for NGS data. Assemblers pull millions of reads together into larger contiguous sequences, called *contigs*. A typical assembly of a eukaryotic genome is a set with thousands of contigs. These contigs are unordered as well as unoriented, i.e. it is unknown whether they come from the forward or reverse strand. The process to determine orientation and relative ordering of contigs is called *scaffolding*. Some assemblers have built-in scaffolders; otherwise, an external scaffolder can be used, such as Bambus (Pop *et al.*, 2004b). An alternative to *de novo* assembly is *mapping* the reads against a finished or draft genome from a close relative (a *template*). From such a mapping a consensus can be called, generating a *comparative assembly* (Pop *et al.*, 2004a). As the number of known genomes is growing rapidly, in the future, it will be more often the case that multiple close relative genomes are available to create such assemblies. However, mapping against a closely related genome will only yield those parts that are identical in target and template genome. To get the unique components in the target genome, a *de novo* assembly will always be required.

Assemblers are often specialized for a specific type of reads. De Bruijn graph-based assemblers, such as Velvet (Zerbino and Birney, 2008), Abyss (Simpson *et al.*, 2009) and ALLPATHS (Maccallum *et al.*, 2009) are most suitable for short reads (Solid; Solexa), whereas overlap-layout-consensus algorithms, such as Newbler (Roche) and CABOG (Miller *et al.*, 2008), are more suitable for the longer 454 sequences. It is not trivial to deal efficiently with different read types simultaneously or to combine outputs of different assemblers. Hybrid strategies (using two types of sequencing data) mostly work by altering the output of a first assembler to make it suitable for application on a second. Reinhardt *et al.* (2009) generated contigs with VCAKE (Jeck *et al.*, 2007) using Solexa data, which were subsequently used as input to the Newbler assembler together with 454 data. Goldberg *et al.* (2006) simulated Sanger reads from a set of contigs assembled by Newbler with 454 data. These reads were subsequently used as input to the Celera assembler combined with true Sanger reads. We are aware of only one *de novo* assembler designed to integrate Sanger and NGS data, called Forge (Diguistini *et al.*, 2009). However, Forge does not allow for integration of comparative assemblies. Other hybrid strategies (Argueso *et al.*, 2009; Salzberg *et al.*, 2008) combine assemblies using Minimus (Sommer *et al.*, 2007). Minimus is restricted to only two assemblies, so to combine three or more assemblies it has to be applied iteratively. Minimus also does not allow for weighted combinations of contigs.

In this article, we describe MAIA (Multiple Assembly IntegrAtor), a graph-based algorithm for integration of several *de novo* and comparative assemblies. Assembly integration is related to both *de novo* assembly and scaffolding, but differs in its input. An assembler deals with short sequences (reads) and high genome coverage to account for read errors and repeats in the genome. A scaffolder tries to determine the relative ordering and orientation of large sequences (contigs) of a single *de novo* assembly, assuming the target genome is covered once. An integrator is a hybrid of

*To whom correspondence should be addressed.

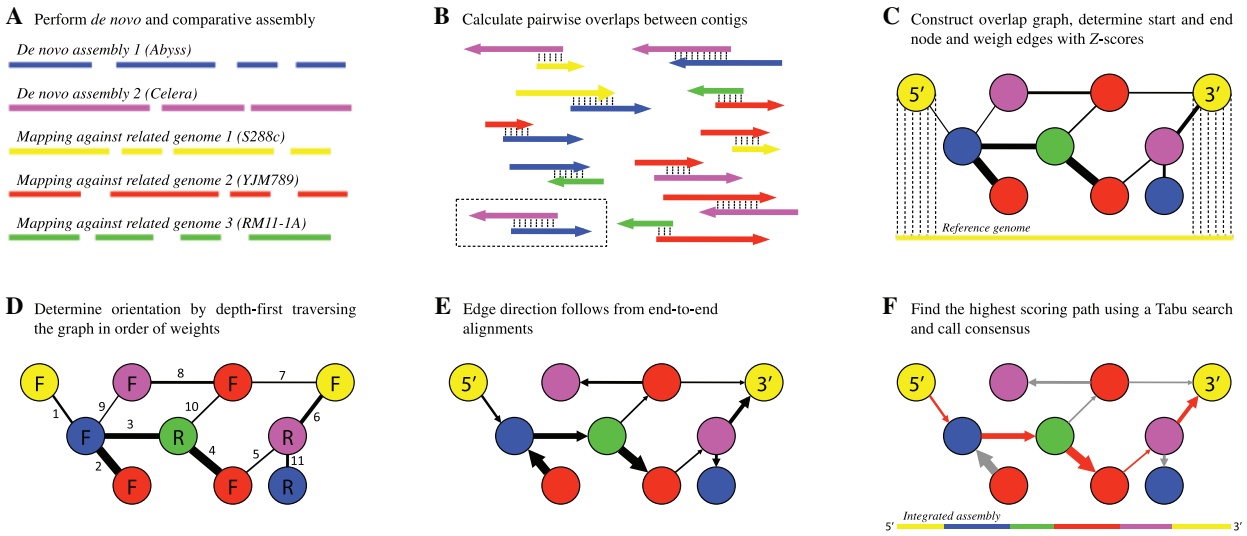

Fig. 1. An overview of the process of integrating several assemblies with MAIA. (**A**) Multiple *de novo* and comparative assemblies are created using specialized assemblers. (**B**) The resulting contigs are pairwise aligned to each other to find end-to-end overlaps. (**C**) An overlap graph is constructed, in which nodes represent contigs and edges represent overlaps. A forward and a reverse edge is added between the pairs of nodes, but these are indicated by an undirected edge for simplicity. A start node and an end node is determined using a reference genome. Edges are assigned weights based on several properties of the alignments and contigs, combined using weighted *Z*-scores. (**D**) An orientation is assigned to the contigs by traversing the graph depth-first in order of weight (indicated by the numbers). Edge 9 [dashed box in (B)] assigns reverse orientation to the blue node, while a forward orientation has already been assigned via edge 1, therefore it is recognized as conflicting and it is removed. (**E**) Oriented contigs and end-to-end overlaps form a directed graph. (**F**) The highest scoring path is found using a Tabu search procedure, which leads to the assembly of a chromosome.

these, dealing both with contigs and manifold genome coverage, allowing a number of assemblies to be considered simultaneously. MAIA is not restricted in the number of assemblies and uses the full contigs produced, not requiring these to be broken into reads or *k*-mers of any type. Pairwise alignments of contigs are calculated to generate an overlap graph. In this graph nodes represent contigs and edges represent alignments. These edges are weighted with several properties of the contigs and alignments, which are combined using weighted *Z*-scores. Assemblies are integrated at chromosome level by finding the combination of contigs which yields the highest score. This is achieved by optimizing a path in the overlap graph between the contigs that align closest to the 5′ and 3′ ends of a reference genome. The assembled chromosome directly follows from this path.

The MAIA approach has two main advantages. First, multiple known related genomes can be used simultaneously in the assembly process. Second, different NGS sources can be assembled with specific *de novo* assemblers, to be integrated afterwards with MAIA. As a demonstration of the algorithm, MAIA is applied to the *Saccharomyces cerevisiae* strain CEN.PK 113-7D, a strain widely used for systems biology research and metabolic engineering (Knijnenburg *et al.*, 2008; Medina *et al.*, 2010; Wisselink *et al.*, 2009). Its genome is assembled using Solexa reads, 454 reads and three genomes of previously sequenced, closely related *S.cerevisiae* strains. The method is compared with two other hybrid approaches, using Minimus and Velvet.

2 METHODS

MAIA is an assembly integrator using the overlap-layout-consensus paradigm, known from genome assembly algorithms, to combine several assemblies into a single integrated assembly. The algorithm takes as input sets of contigs, each set originating from either a *de novo* or a comparative assembly, i.e. from mapping against a related genome (Fig. 1A). Overlap between contigs is detected by pairwise aligning the contigs in an all-vs-all fashion among the sets (Fig. 1B). An undirected overlap graph is then constructed, with nodes representing contigs and edges representing overlaps. Using a reference genome, i.e. the evolutionary closest of the related genomes available that is of high quality, a 5′ start and 3′ end node is determined to guide the integration (Fig. 1C). The edges are weighted, reflecting the likelihood that the alignment represents an actual overlap in the genome (Fig. 1D). The graph is then directed by assigning orientation to the contigs, i.e. forward or reverse (Fig. 1E). Assembly integration is finally achieved by finding a highest scoring path between the start and end nodes in the overlap graph and calling the consensus (Fig. 1F). These steps are described in detail below.

2.1 Constructing an overlap graph from pairwise alignments of contigs

A graph $G=(V,E)$ is created in which $V=\{v_1,v_2,...,v_n\}$ is the set of nodes and $E=\{\{v_i,v_j\}|v_i,v_j\in V\}$ is the set of edges. Each contig $c\in C$ is assigned to a node. Overlapping regions in contigs are detected by pairwise aligning all contig pairs in different sets. For every aligned pair of contigs, two filters are applied. First, only the longest mutually consistent set of alignments is selected. Second, if there still is more than one match between two contigs, only the longest is retained. For these steps we used Nucmer and Delta-filter, both part of the Mummer package (Delcher *et al.*, 2002), although other tools could be used. The resulting alignments $a\in A$ are used to generate the edges in G. Contigs v_i and v_j that overlap end-to-end with a minimum alignment length $l^{A,\min}$ and maximum length $l^{O,\max}$ of non-aligned overhang (Fig. 2), i.e. the part of the contig that will be clipped when merging the two, are then joined by a forward and a reverse edge, (v_i,v_j) and (v_j,v_i).

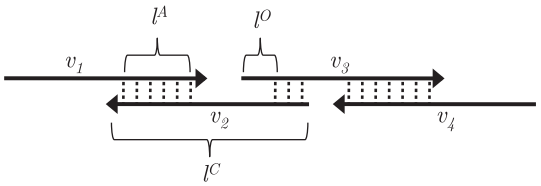

Fig. 2. Four pairwise aligned contigs. The alignment length l^A, the contig length l^C and the length of the non-aligned overhang l^O are three of the four properties used for edge weighting, the fourth is an assembly quality score.

2.2 Weighting the edges in the overlap graph

A score $Z(e)$ is assigned to each edge $e \in E$ to reflect the quality of the alignment and the quality of the contig to which the edge leads. Edge weights differ between forward and reverse edges. For three properties of contigs and alignments (Fig. 2), a *P*-value is calculated. Null distributions for these properties are inferred using all contigs and possible contig pairs. These distributions reflect the probability that the property occurs by chance in a pair of contigs, which do not overlap in the target genome. The *P*-values are transformed into *Z*-scores using the inverse of the cumulative density function N^{-1}, and combined into Liptak–Stouffer's weighted *Z*-score, where the weights w_i are user-specified per property (Hwang *et al.*, 2005):

$$Z(e) = \frac{1}{\sqrt{\sum w_i^2}} \left[\sum_{i=1}^{3} w_i N^{-1}(1 - p_i(e)) + w_4 Z_4(e) \right] \quad (1)$$

For each edge $e \in E$ the following four properties (illustrated in Fig. 2) are calculated:

(1) The length of the contig; longer contigs are preferred over smaller contigs. A *P*-value for a particular contig c is estimated as $p(l^C \geq l_c^C) = \frac{|\{c' \in C | l_{c'}^C \geq l_c^C\}|}{|C|}$, where l_c^C is the length of the contig c to which edge e points and $|C|$ is the total number of contigs.

(2) The length of the alignment; longer overlap between contigs is preferred. For the calculation of the *P*-value, only the number of correctly aligned nucleotides $l^{A*} = l^A \cdot f^A$ are considered, where l^A is the full length of the alignment and f^A is the fraction of aligned nucleotides that are identical. The *P*-value for a particular alignment a is estimated as $p(l^{A*} \geq l_a^{A*}) = \frac{|\{a' \in A | l_{a'}^{A*} \geq l_a^{A*}\}|}{|A|}$ where $|A| = \prod_{x=1}^{n} |C_x|$ is the total number of possible contig pairs, x is the assembly number and n is the total number of assemblies.

(3) The percentage of non-aligned overhang; the length of the non-aligned overhang l^O should ideally be zero. For a particular alignment a a *P*-value is calculated as $p(l^O \leq l_a^O) = \frac{|\{a' \in A | q_{a'} \leq q_a\}|}{|A|}$, where $q_a = \frac{l_a^O}{l_a + l_a^O}$ is the fraction of non-aligned overhang and l_a^O is the number of nucleotides that have to be clipped if the two contigs would be merged. We consider the number of nucleotides overhang relative to the contig length to avoid connecting small contigs with large overhangs.

Finally, a manually assigned score Z_4 is added for the quality of the assembly, which can differ per assembly source. This score attribute is used to have MAIA prefer high-quality assemblies.

2.3 Directing the overlap graph

All contig alignments are end-to-end and can be represented as directed edges in the overlap graph. The direction of each edge depends on the orientation of the contigs it connects. If the upstream end of node v_i aligns to the downstream end of node v_j, the edge in the graph would be $e = (v_i, v_j)$. Since the orientation of the contigs is unknown, taking the reverse complement of the two contigs flips the edge to $e = (v_j, v_i)$. These two edges represent the forward and reverse strands of the DNA. Since only one strand needs to be assembled, the orientation of the contigs is fixed.

Assigning an orientation to the contigs can cause problems, by introducing cycles in the graph that disagree on orientation. These cycles are caused by alignments of contigs that are not actually overlapping in the genome. Edges causing these conflicts have to be removed. An optimal solution would be to assign an orientation to the contigs which minimizes the number of conflicting edges. Since this problem is non-deterministic polynomial-time hard (NP-hard), a greedy approach is used, similar to the contig orientation method in Bambus (Pop *et al.*, 2004b). This approach starts by fixing the orientation of the start node to *forward*. Next, the graph is traversed depth-first in order of descending weights. For every node an orientation is assigned based on the alignment and orientation of the previously visited node.

The contig orientation is illustrated by the example in Figure 2, in which arrows represents contigs and dashes their alignments. Node v_1 is the start node and has a fixed *forward* orientation. The graph is traversed to v_2. Since the reverse complement of v_2 aligns to v_1 (opposing arrows), a *reverse* orientation is assigned to v_2. Subsequently, using the same reasoning, a *forward* and *reverse* orientation will be assigned to v_3 and v_4, respectively. If v_4 had already been visited and was assigned a *forward* orientation, the edge between v_3 and v_4 conflicts with the previously assigned orientation and will be removed from the graph. After all nodes have been oriented it is known for each end of a contig whether it is the up- or downstream end. The end-to-end alignments can now be used to direct the graph, e.g. as node v_2 aligns to the downstream end of node v_1, the directed edge will be $e = (v_1, v_2)$.

2.4 Finding the highest scoring path

A chromosome can be assembled by finding a simple path $P = v_0 v_1 v_2 ... v_k$ connected by edges $e_1 e_2 ... e_k$ in the overlap graph, visiting no node more than once (Fig. 1F). A start node v_0 and end node v_k are determined to avoid having to evaluate paths between all possible node pairs in the graph. These nodes are set to be those contigs that originate from the 5′ and 3′ ends of comparative assembly against the reference genome (Fig. 1C). The combination of contigs connecting v_0 and v_k is optimized by maximizing the sum of the edge scores $S(P) = \max(\sum_{e \in P} Z(e))$. This optimization can be shown to be NP-complete by taking an instance of G with only positively weighted edges, thereby reducing the maximization to a search for a Hamiltonian path, which is known to be NP-complete. This makes finding the global optimum computationally expensive; therefore, we search for the highest scoring path using a Tabu procedure (Glover, 1986).

The Tabu search starts by finding an initial solution for P by performing a Dijkstra shortest path search on the graph with inverted edge weights $\hat{Z}(e)$. These inverted weights are calculated for each edge e as $\hat{Z}(e) = \max_e(Z(e)) - Z(e) + 1$. The Tabu search proceeds by systematically applying the change to the path that yields the most improvement in terms of $S(P)$. All pairs of adjacent edges in P are considered for modification (2-opt). Four modifications are possible to a set of two edges. Figure 3 shows an example for a set of three nodes v_i, v_j and v_k, connected by the edges (v_i, v_j) and (v_j, v_k). The possible modifications are: (i) v_j is bypassed by directly connecting v_i and v_k with edge (v_j, v_k); (ii) v_i and v_k are connected via a fourth node v_l; (iii) v_i or v_j are connected via v_l; and (iv) v_i and v_j are connected via v_l. After the change has been applied, the inverse of the change (the 'undo') is stored in the Tabu list. Changes in the Tabu list are not allowed to be applied to avoid entrapment in cycles of repeated identical changes. After a certain number of cycles (here, 3) the change is removed from the Tabu list. The algorithm proceeds until for a certain number of changes (15) no improvement is seen compared with the best path found so far. As the initial solution is often close to the final one, convergence is often fast. In practice, the algorithm is limited by memory size (to hold the overlap graph) rather than computational complexity.

2.5 Connecting unconnected subgraphs

If no path exists between start and end node, contigs from one or more assemblies are aligned to the reference. If a region of the reference genome

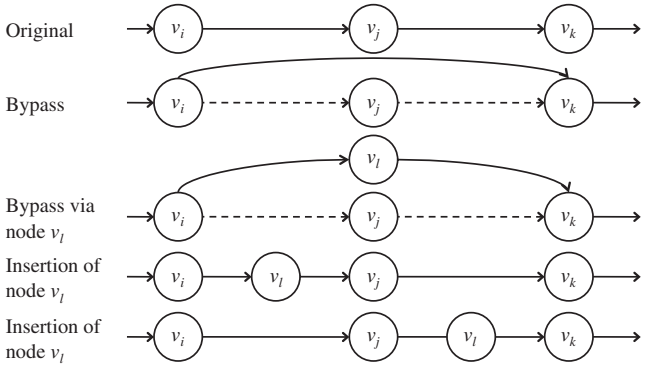

Fig. 3. Four types of modifications that are applied iteratively to the path by the Tabu search procedure.

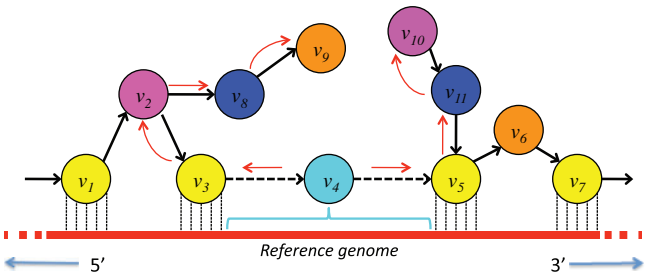

Fig. 4. A pseudo node $\tilde{v}_4$ is inserted in the graph if no path exists between start and end node. An initial path $P = v_1v_2v_3\tilde{v}_4v_5v_6v_7$ has been found. If $\tilde{v}_4$ exceeds the maximum size $l_{\tilde{c}}^{\max}$, the path is subsequently split into $P_1 = v_1v_2v_3$ and $P_2 = v_5v_6v_7$. P_1 and P_2 are backtracked (red arrows) to the nearest branchpoints (v_2 and v_5, respectively) and greedily extended to v_9 and v_{10} yielding $P_3 = v_1v_2v_8v_9$ and $P_4 = v_{10}v_{11}v_5v_6v_7$. Finally, the highest scoring paths are chosen from $\{P_1, P_3\}$ and $\{P_2, P_4\}$.

is not covered by an aligned contig, a pseudo node $\tilde{v}$ is created, containing the DNA sequence of this non-covered region (Fig. 4). Edges with a low score (i.e. a penalty) are inserted between $\tilde{v}$ and the pair of nodes that align on both sides of $\tilde{v}$. The number of pseudo nodes in the graph is kept minimal by gradually increasing the allowed pseudo node size until a path between start and end node is found.

2.6 Post-processing of the path

The start and end node determined using the reference genome are not necessarily the ends of the target chromosome. Therefore, the path P is greedily extended toward the 5′ and 3′ extremes of the target genome. P is iteratively extended from the current last node to the node connected with the highest edge weight, provided that it has a specified minimum alignment length $l^{A,\min}$ and minimum percentage alignment identity $f^{A,\min}$.

The maximum size of a pseudo node is set to $l_{\tilde{c}}^{C,\max}$. P is split at pseudo nodes exceeding $l_{\tilde{c}}^{C,\max}$ and backtracked to the nearest branchpoint on both sides of the pseudo node. From there on the paths are greedily extended until no extension is possible, similar to the end extension described above, resulting in multiple contigs per chromosome. Figure 4 gives an example of splitting P at a large pseudo node.

2.7 Calling the consensus

Finally, the integrated contigs follow from the path found by the Tabu search. The contigs and their associated pairwise alignments are transformed into an alignment matrix with one row for every source assembly. The consensus is called by taking for each column the base or gap (arising from the nucmer gapped alignment) of the highest quality assembly present in that column. In the resulting consensus, gaps are removed and the bases are tagged with the assembly from which they originate. This information can be used in further analyses of the assembly.

2.8 Assembly validation

To assess the quality of both the individual assemblies and the MAIA integrated assembly, the paired-end Solexa reads were mapped onto the assemblies using the Burrows-Wheeler Alignment tool BWA (Li and Durbin, 2009). Two statistics were extracted from the mappings using Samtools (Li *et al.*, 2009). First, to assess the completeness of an assembly, the percentage of reads that mapped on the assembly was calculated. Second, to assess the accuracy of an assembly, the percentage supporting read pairs was calculated. This is calculated as the percentage of the total number of mapped pairs that map at a proper distance from each other on a contig. The insert size distribution $N(208, 13)$ and the maximum allowed insert size ($\sim 6\sigma$) were estimated by BWA.

2.9 Experimental setup

DNA of the *S.cerevisiae* lab strain CEN.PK 113-7D (*MATa MAL2-8c SUC2*) was prepared (Burke *et al.*, 2000). A library of 200 bp fragments was created and sequenced paired-end using the Illumina Solexa system, generating $\sim$56 million paired reads. A second library with mate-pair reads with an insert size of 8 kb was prepared sequenced on the Roche 454 Titanium. Both libraries were prepared according to manufacturer recommendations (Illumina and Roche). The pairing rate for the 454 mate pair library was 19%, yielding 149 900 paired reads.

De novo unscaffolded assemblies were performed with Abyss (Simpson *et al.*, 2009) and the Celera assembler (Miller *et al.*, 2008) on the Solexa and 454 reads, respectively. Abyss was tested for all combinations of k-mer size $\in \{23, \ldots, 33\}$ and coverage cut-off $\in \{0, \ldots, 12\}$; the combination yielding the best N50 was chosen. The Celera assembler was used with standard settings as described in Lee (2007). Comparative assemblies were made by mapping the Solexa reads to the (draft) genomes of the *S.cerevisiae* strains S288c, YJM789 and RM11-1A using MAQ (Li *et al.*, 2008). These genomes are 99.3, 98.4 and 98.0% identical to CEN.PK, calculated by dividing the number of identical bases by the length of the genome. The consensus sequences were split into contigs at every occurrence of an 'N'. Contigs <200 bp have been discarded.

Integration of the assemblies with MAIA has been performed per chromosome. From the S288c comparative assembly, only contigs originating from the chromosome being assembled were used. A minimum alignment length $l^{A,\min}$ of 20 nt is used for finding pairwise alignments. MAIA finds all contigs that align end-to-end. The maximum allowed non-aligned overhang $l^{O,\max}$ was set to 10 nt. Scores for the assembly qualities were set to $Z = 3, 2.5, 2, 1, 0.5$ for the Abyss, Celera, S288c, YJM789 and RM11-1A assembly, respectively, reflecting our beliefs concerning the relative quality of the assemblies. *De novo* assemblies received the highest Z-scores, since these may contain structural variants unique to the target genome. The weights in the combined Z-score for the contig length, alignment length, non-aligned overlap and assembly quality were rather arbitrarily set to be 0.35, 0.25, 0.15 and 0.25, respectively, corresponding to the relative importance of the forms of evidence for merging contigs. Pseudo nodes are iteratively added with increasing sizes until a path from start to end node is found. Only contigs from the S288c assembly were used to create pseudo nodes. The edge weight of a pseudo node is set to -10 and maximum pseudo node size $l_{\tilde{c}}^{C,\max}$ was set to 250.

Two other hybrid methods were applied as a comparison. First, a *de novo* assembly with Velvet was performed on a combination of the Solexa reads and 454 data-based contigs pre-assembled by the Celera assembler. The parameters (k-mer size and coverage cut-off) were optimized w.r.t. the N50. The paired-end information of the Solexa reads was then used for scaffolding.

Table 1. CEN.PK assembly statistics of single input and hybrid assemblies

Strategy	Assembly	Package	# contigs	Total size (Mb)	N50 (kb)	Mapped reads (%)	Supporting pairs (%)
Single input	*De novo*	Abyss	1223	11.64	20	84.8	97.6
	De novo	Celera	4148	9.03	3	62.8	98.5
	Comparative (S288c)	Maq	375	12.06	162	96.9	99.0
	Comparative (YJM789)	Maq	907	11.77	44	90.8	98.3
	Comparative (RM11-1A)	Maq	795	11.54	41	78.2	98.5
Hybrid	*De novo*	Velvet	654	11.40	72	75.5	97.7
	De novo + comparative	Minimus	71	12.21	290	92.1	99.3
	De novo + comparative	MAIA	29	12.01	918	96.5	99.3

Only contigs ≥200 bp were used to generate statistics (two rightmost columns).

Second, an assembly integration with Minimus was performed by merging two assemblies and iteratively applying Minimus to the merged result and a next assembly, whereby the singletons were discarded in every step. The order of combination was: S288c + Abyss, + Celera, + YJM789, + RM11-1A.

3 RESULTS AND DISCUSSION

MAIA has been developed to integrate multiple assemblies. An integrated assembly of the *S.cerevisiae* lab strain CEN.PK 113-7D, from hereon called CEN.PK, has been constructed to demonstrate the algorithm.

3.1 Individual assemblies are fragmented and vary in error rates

Two *de novo* and three comparative assemblies have been made for CEN.PK. The results for the individual assemblies are shown in Table 1. Despite the high genome coverage (∼160*X* for the Solexa and ∼20*X* for the 454 data), the Abyss and Celera *de novo* assemblers generated fragmented assemblies, with an *N*50 of 20.3 and 2.7 kb, respectively. The N50 is the smallest possible contig length, such that the sum of lengths of all contigs $c' \in C$ with $l_{c'}^{C} \geq N50$ is at least 50% of the total assembly size.

The level of fragmentation of the comparative assemblies depends on evolutionary closeness and quality of the genome. The comparative assembly using S288c as template yields the best individual assembly, covering 12.06 million nucleotides with only 375 contigs. The available S288c genome is of high quality and evolutionary closer to CEN.PK than the other strains (Schacherer *et al.*, 2009). Most reads could be mapped to the S288c comparative assembly, which is therefore the most complete; only 3.1% of the reads could not be mapped. The least number of reads mapped to the Celera and Velvet hybrid assemblies. Running Velvet to assemble only Solexa reads (results not shown) allowed 10% more reads to be mapped. That the use of more 454 reads lowers the percentage of mapped Solexa reads, hints at 454 data quality problems.

3.2 MAIA drastically lowers the number of contigs

The number of contigs >200 bp in the individual source assemblies range from 375 to 1,223. MAIA reduces this to 29. Most chromosomes have been assembled in a single contig, except for chromosomes 1, 3, 8, 10, 12 and the mitochondrial DNA, which consist of 5, 4, 3, 2, 2 and 2 contigs, respectively. These chromosomes are known to be relatively divergent among

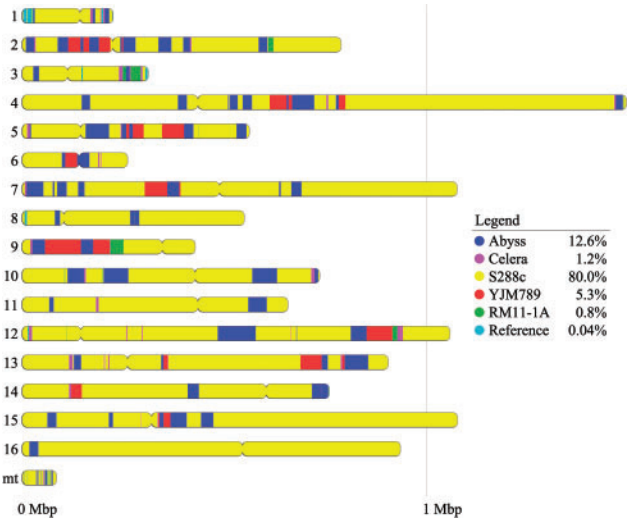

Fig. 5. Usage of the different assemblies in the input per chromosome.

S.cerevisiae strains. Schacherer *et al.* (2009) showed deleted regions in every one of these chomosomes using a whole-genome tiling array. The most apparent of these deletions is the 10 kb deleted region at the extreme of the left arm of chromosome 1, which is also seen in the MAIA assembly (Fig. 5). The splits in the chromosomes assembled by MAIA are generally observed near their ends, which are known to be divergent regions in yeast (Argueso *et al.*, 2009). Divergent regions can benefit less from comparative assemblies and therefore MAIA cannot fully close the genome.

The final integrated CEN.PK genome is compiled of five source assemblies; two *de novo* and three comparative assemblies. Four additional MAIA runs were performed where in each step one of the assemblies was incrementally added to its input, starting with only the S288c comparative assembly. Figure 6 shows that each individual input assembly positively contributes to the final result. Table 1 and Figure 5 show the assemblies and their use for integration. The usage differs from only 0.8% for the comparative assembly with RM11-1A as template to 80% with S288c as template. The S288c genome is fully finished and of high quality. S288c and CEN.PK are both laboratory strains, known to be evolutionary close (Schacherer *et al.*, 2009); therefore mapping yields large contigs. Both contig quality and contig length are reflected in the *Z*-scores

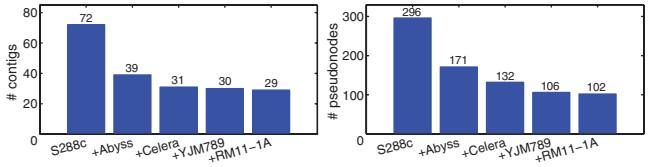

Fig. 6. MAIA results for the incremental addition of input assemblies.

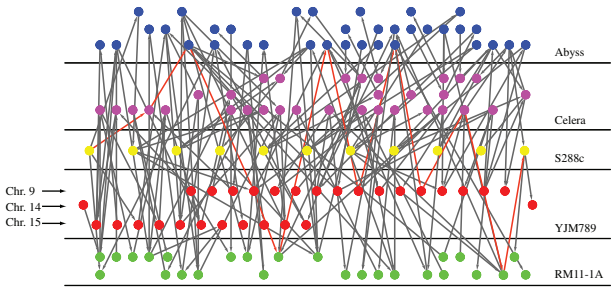

Fig. 7. The overlap graph for chromosome 9. The highest scoring path found by the Tabu search is indicated by the red arrows.

on the edges in the overlap graph. Therefore, MAIA has a preference for the S288c contigs and often selects them for integration. On the contrary, the RM11-1A genome is a draft genome composed of a set of supercontigs. RM11-1A is a phylogenetically more distant wine strain and therefore contributed to a much lower extend in the final assembly. The MAIA assembly contains 0.04% sequences from 102 pseudo nodes, which are 4340 nt in stretches individually not >250 bp. These sequences do no originate from read data, but from the reference genome.

As an illustration the overlap graph of chromosome 9 is shown in Figure 7. All five input assemblies are used to construct chromosome 9 of CEN.PK. The contigs of the YJM789 comparative assembly have been grouped by the chromosome from which they originate and are divided among three levels in the layout, indicated by the arrows in Figure 7. Contigs originating from YJM789's chromosomes 14 and 15 appear in this graph because of repeat sequences that are present in both these chromosomes and CEN.PK's chromosome 9. Although these repeat-induced connections are present, the Tabu search does not include them in the path. Only the contigs originating from YJM789's chromosome 9 are incorporated in the MAIA integrated chromosome 9 of CEN.PK.

3.3 MAIA integrates assemblies at low error rate

The quality of both the integrated and single assemblies has been assessed using the percentage of mapped pairs that map at a proper distance from each other (Table 1). These supporting pairs reflect the accuracy of the assembly algorithms. In both the MAIA and the Minimus assemblies, 99.3% of the mapped pairs can be mapped at their proper distance, showing that these assemblies are of the highest quality in the list. However, only 92.1% of the reads mapped on the Minimus assembly.

The S288c comparative assembly is 50 kb longer than the MAIA assembly. This is also reflected in the percentage of reads that map to the assemblies; 96.9% of the reads map to the S288c comparative assembly compared with 96.5% to the MAIA assembly (Table 1).

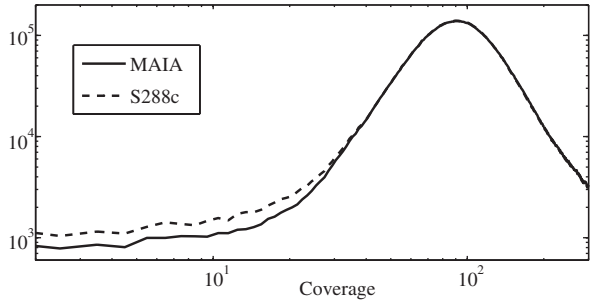

Fig. 8. Histograms of coverage of reads mapped on the MAIA integrated assembly and the S288c comparative assembly.

The length difference can be partially attributed to the relaxed comparative assembly settings that were used for Maq; no minimum read depth and mapping quality was used. Analysis of the reads mapped by BWA on both the MAIA integrated assembly and the S288c comparative assembly showed that a far larger part of the latter is covered by only few reads (Fig. 8). In particular, 15 kb of the nucleotides were covered by five reads or less, whereas for the MAIA assembly this was the case for only 3 kb.

3.4 Other hybrid strategies are less complete

The Minimus assembly contains >12 million base pairs, but only 92.1% of the reads mapped to it. This indicates the iterative approach taken with Minimus results in overlapping information within the integrated assembly. The hybrid *de novo* assembly generated with Velvet is less complete than the MAIA assembly; only 75.5% of the reads can be mapped to it.

4 CONCLUSIONS

We developed MAIA, an integrator for assembly information. Our work extends previously developed algorithms for *de novo* and comparative assembly, enabling integration of multiple assemblies at once. MAIA makes it possible to use specific assemblers for different NGS data sources, to use multiple reference genomes for comparative assemblies or to combine outputs of different runs of an assembler. The number of known genomes is currently increasing rapidly. In the future it will be more common that multiple closely related genomes are available, as is the case already for *S.cerevisiae*. These genomes can be leveraged by using MAIA in combination with a comparative assembler.

MAIA improves genome assemblies by making use of all available information. The algorithm integrates single assemblies from different sources into longer contigs, up to chromosomal length as shown in the *S.cerevisiae* assembly integration. Five single assemblies were integrated into 29 contigs covering 12.01 Mb. In the MAIA integrated assembly, 99.3% of the mapped read pairs mapped at a correct distance from each other. This percentage is higher than for each of the single assemblies, indicating that the integrated assembly is of higher quality than the single assemblies.

The edge weighting system in MAIA can be extended. Integration is achieved by building an overlap graph from pairwise aligned contigs and subsequently finding the highest scoring path. Edges in this graph are weighted by properties of the involved contigs and alignments. Currently, four properties for the edge weighting

are implemented. These can be extended by calculating *P*-values for additional properties such as alignment-overspanning mate pair data, distances of contigs on the related genomes, or physical or genetic map information.

ACKNOWLEDGEMENTS

We would like to thank David Adams from the Sanger institute for his help with obtaining the 454 reads.

Funding: This project was carried out within the research programme of the Kluyver Centre for Genomics of Industrial Fermentation which is part of the Netherlands Genomics Initiative/Netherlands Organization for Scientific Research.

Conflict of Interest: none declared.

REFERENCES

Argueso,J.L. *et al*. (2009) Genome structure of a *Saccharomyces cerevisiae* strain widely used in bioethanol production. *Genome Res.*, **19**, 2258–2270.

Burke,D. *et al*. (2000) *Methods in Yeast Genetics: a Cold Spring Harbor Laboratory Course Manual*. 2000 edn. Cold Spring Harbor Laboratory Press, Plainview, N.Y.

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

Diguistini,S. *et al*. (2009) *De novo* genome sequence assembly of a filamentous fungus using sanger, 454 and illumina sequence data. *Genome Biol.*, **10**, R94.

Glover,F. (1986) Future paths for integer programming and links to artificial intelligence. *Comput. Oper. Res.*, **13**, 533–549.

Goldberg,S.M.D. *et al*. (2006) A sanger/pyrosequencing hybrid approach for the generation of high-quality draft assemblies of marine microbial genomes. *Proc. Natl Acad. Sci. USA*, **103**, 11240–11245.

Hwang,D. *et al*. (2005) A data integration methodology for systems biology. *Proc. Natl Acad. Sci. USA*, **102**, 17296–17301.

Jeck,W.R. *et al*. (2007) Extending assembly of short dna sequences to handle error. *Bioinformatics*, **23**, 2942–2944.

Knijnenburg,T.A. *et al*. (2008) Combinatorial influence of environmental parameters on transcription factor activity. *Bioinformatics*, **24**, i172–i181.

Lee,W.W. (2007) Using the Celera Assembler. Available at http://wgsassembler.sourceforge.net/UsingCeleraAssembler.pdf (last accessed date January 8, 2010).

Li,H. and Durbin,R. (2009) Fast and accurate short read alignment with burrows-wheeler transform. *Bioinformatics*, **25**, 1754–17560.

Li,H. *et al*. (2008) Mapping short dna sequencing reads and calling variants using mapping quality scores. *Genome Res.*, **18**, 1851–1858.

Li,H. *et al*.; 1000 Genome Project Data Processing Subgroup (2009) The sequence alignment/map format and samtools. *Bioinformatics*, **25**, 2078–2079.

Maccallum,I. *et al*. (2009) Allpaths 2: small genomes assembled accurately and with high continuity from short paired reads. *Genome Biol.*, **10**, R103.

Medina,V.G. *et al*. (2010) Elimination of glycerol production in anaerobic cultures of a *Saccharomyces cerevisiae* strain engineered to use acetic acid as an electron acceptor. *Appl. Environ. Microbiol.*, **76**, 190–195.

Miller,J.R. *et al*. (2008) Aggressive assembly of pyrosequencing reads with mates. *Bioinformatics*, **24**, 2818–2824.

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

Pop,M. *et al*. (2004b) Hierarchical scaffolding with bambus. *Genome Res.*, **14**, 149–159.

Reinhardt,J.A. *et al*. (2009) *De novo* assembly using low-coverage short read sequence data from the rice pathogen *Pseudomonas syringae pv. oryzae*. *Genome Res.*, **19**, 294–305.

Salzberg,S.L. *et al*. (2008) Gene-boosted assembly of a novel bacterial genome from very short reads. *PLoS Comput. Biol.*, **4**, e1000186.

Schacherer,J. *et al*. (2009) Comprehensive polymorphism survey elucidates population structure of *Saccharomyces cerevisiae*. *Nature*, **458**, 342–345.

Shendure,J. and Ji,H. (2008) Next-generation dna sequencing. *Nat. Biotechnol.*, **26**, 1135–1145.

Simpson,J.T. *et al*. (2009) Abyss: a parallel assembler for short read sequence data. *Genome Res.*, **19**, 1117–1123.

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

Wisselink,H.W. *et al*. (2009) Novel evolutionary engineering approach for accelerated utilization of glucose, xylose, and arabinose mixtures by engineered *Saccharomyces cerevisiae* strains. *Appl. Environ. Microbiol.*, **75**, 907–914.

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