Vol. 30 no. 1 2014, pages 40–49
doi:10.1093/bioinformatics/btt590

Genome analysis

Advance Access publication October 15, 2013

De novo finished 2.8 Mbp *Staphylococcus aureus* genome assembly from 100 bp short and long range paired-end reads

David Hernandez[1,*], Ryan Tewhey[2], Jean-Baptiste Veyrieras[3], Laurent Farinelli[4], Magne Østerås[4], Patrice François[1] and Jacques Schrenzel[1]

[1]Genomic Research Laboratory, Infectious Diseases Service, Geneva University Hospitals, 1211 Geneva 4, Switzerland, [2]Scripps Translational Science Institute, The Scripps Research Institute, La Jolla, CA 92037, USA, [3]BioMérieux, Data and Knowledge Laboratory, 69280 Marcy l'Etoile, France and [4]Fasteris SA, PO Box 28, 1228 Plan-les-Ouates, Switzerland

Associate Editor: Michael Brudno

ABSTRACT

Motivation: Paired-end sequencing allows circumventing the shortness of the reads produced by second generation sequencers and is essential for *de novo* assembly of genomes. However, obtaining a finished genome from short reads is still an open challenge. We present an algorithm that exploits the pairing information issued from inserts of potentially any length. The method determines paths through an overlaps graph by using a constrained search tree. We also present a method that automatically determines suited overlaps cutoffs according to the contextual coverage, reducing thus the need for manual parameterization. Finally, we introduce an interactive mode that allows querying an assembly at targeted regions.

Results: We assess our methods by assembling two *Staphylococcus aureus* strains that were sequenced on the Illumina platform. Using 100 bp paired-end reads and minimal manual curation, we produce a finished genome sequence for the previously undescribed isolate SGH-10-168.

Availability and implementation: The presented algorithms are implemented in the standalone Edena software, freely available under the General Public License (GPLv3) at www.genomic.ch/edena.php.

Contact: david.hernandez@genomic.ch

Supplementary Information: Supplementary data are available at *Bioinformatics* online.

Received on January 9, 2013; revised on September 20, 2013; accepted on October 8, 2013

1 INTRODUCTION

The past 5 years have witnessed a strong interest in the problem of *de novo* assembly from sequence reads of short and moderate length due to the rapid adoption of high-throughput sequencing technologies. One critical application is the sequencing and assembly of complete bacterial genomes that has become a routine task for many research laboratories. Although, with these technologies comes the need for novel computational and technological approaches to complement the specificities of each platform (Pop, 2009). One such example is the use of paired-end sequence reads for *de novo* assembly that is essential to overcome the shortness of the reads. By allowing the resolution of repeats larger than the read length, paired-end reads greatly improve *de novo* assembly contiguity (Wetzel *et al*., 2011). However, current assembly strategies still result in fragmented draft genomes complicating downstream analysis. Thus, the ability to produce a finished genome is more than a psychological threshold: it is an invaluable resource for the community as it is the foundation for comparative and evolutionary studies (Tsai *et al*., 2010). Closing bacterial genomes in a fast semi-automatic way was reported (Ribeiro *et al*., 2012). This was done using a combination of Illumina generated short reads, with long reads obtained using Pacific Biosciences' PACBIO RS sequencer. Illumina paired-end reads obtained from short and long inserts distributions are first used to produce an accurate scaffold. Remaining gaps are then filled in with long reads.

Here, we focus on obtaining finished genomes using reads from only the broadly adopted Illumina platform, on which paired-end 100 bp reads are routinely obtained. There exist several protocols for producing whole-genome libraries for Illumina sequencing with the most common resulting in either short paired-end libraries generally <600 bases or long paired-end libraries ranging from 3 to 10 kb. Both such libraries are mandatory for assembly, as bacterial genomes usually contain repeats of several kilobases. Short paired-end reads are generally accurate in length with a standard deviation ranging between 20–50 bp. However, the size of long paired-end is less accurate and may vary significantly.

Using the information of short and long paired-end reads for *de novo* assembly is not straightforward with current approaches not fully exploiting their utility. As most assembler programs make use of short paired-end, only a few of them are able to properly handle long inserts. These approaches have mainly been developed for the purpose of large eukaryotic genomes assembly, though they can also cope with smaller bacterial projects. To date, several strategies exist with the majority using short and long inserts at separate stages of the process.

Long paired-end reads are generally used at a later stage to produce scaffolds. ALLPATHS-LG (Gnerre *et al*., 2011) requires a minimum of two libraries, yielding linking information of short (200 bp) and long (4–5 kb) paired-end. It constructs scaffolds in an iterative manner by successively merging scaffolds obtained during the previous step. In SOAPdenovo (Li *et al*., 2010), paired-end is explicitly used at two stages: scaffolding

*To whom correspondence should be addressed.

and gap closure. During scaffolding, paired-end libraries are used in turn, starting from with the smallest and moving to long insert sizes. Only pairs displaying a unique unambiguous connection are used to join contigs into scaffold. During the second stage, gaps in the scaffold are filled by locally assembling the reads that may be located in the gap region. A related gap closure strategy has been reported in Boetzer and Pirovano (2012). ABYSS (Simpson *et al*., 2009) looks for unambiguous pair connection in the graph using the long inserts at a post-processing scaffolding step.

Another strategy consists in extending seeds by simultaneously using short and long inserts. Ray (Boisvert *et al*., 2010) exploits short and long paired-end reads to iteratively elongating paths through an assembly graph. Telescoper (Bresler *et al*., 2012) leverages an alternate seed extension strategy, which works by only building a local graph at the extension region and assessing a statistical score for possible extension candidates. It was specifically developed for assembling telomeric regions and requires to be provided with set of strings to start from.

The seed extension strategy has a significant advantage over other approaches in that it avoids the computation and storage of all possible routes between paired-reads, which rapidly become intractable as the distance between paired reads increases.

We present a novel seed extension strategy that discovers paths through a condensed overlaps graph. The graph structure itself is left untouched while the paths are determined. Possible paths extensions are represented by a tree structure. As soon as an extension is confirmed, it is fixed and other paths are pruned from the tree. The approach simultaneously makes use of the information issued from both types of inserts, short and long to resolve complex tangles.

The minimum overlaps size (similar to the *k*-mer size for DeBruijn graph based approaches) is the key parameter to optimize for a successful assembly. It is common to try a range of values and to choose the one that maximizes the assembly contiguity. The optimal setting is a trade-off between resolving short repeats and conserving the graph connectivity in weakly covered regions. We introduce a way to automatically determine suitable minimum overlap sizes according to the contextual sequencing coverage. Our method identifies and removes overlaps whose sizes are significantly smaller than what could be expected according to the local overlaps sizes distribution. This makes the assembly significantly less sensitive to the minimum overlaps parameter, which in turn simplify the parameterization task for the end users.

The described approaches are implemented in the new version of our overlaps–graph-based *de novo* assembler Edena (Hernandez *et al*., 2008). In addition, the program features an interactive shell that allows investigating an assembly at targeted regions. This mode provides a valuable assistance for many tasks such as resolving ambiguities, gap filling, assessing suspect assembly or designing primers.

We demonstrate the efficiency of Edena by assembling two *Staphylococcus aureus* strains from experimental Illumina GA sequencing data. Using paired-end reads sequenced from short and long insert libraries, we produce a finished 2.82 Mbp circular genome of the strain SGH-10-168. The sequence is assessed against a phylogenetically related parent and by polymerase chain reaction (PCR) at targeted regions. We also resequence the strain MW2 with paired-end reads obtained from short and long insert libraries. This assembly yields nine contigs that fully covers the genome with perfect accuracy.

2 METHODS

2.1 Loading the reads sequence files

Reads are first loaded into memory. However, the program only stores a single instance of equivalent reads. We say that two reads r_1, r_2 are equivalent if either r_2 or the reverse complement of r_2 is identical to r_1. Equivalent reads are merged into a single entry, though an individual numerical ID is kept for every single read in the input dataset. This allows conserving the read pairing information. Identifying equivalent reads is performed on the fly during the loading procedure. A binary search tree is used for that purpose. Such structure allows looking up for a read in $O(\log_2 n)$ time, with n being the number of elements already stored in the binary tree. Each newly loaded read is first searched for a possible equivalent read in the binary tree. If an equivalent one is found, we only store a unique identifier as well as the orientation information. Otherwise, a new read instance is added to the tree. This preprocessing step keeps track of every individual read as well as its orientation. The new set of reads defined by the binary search tree is thus non-redundant and is generally significantly smaller than the actual read set.

2.2 Building the overlaps graph

The raw overlaps graph consists of one node for every read from the non-redundant set. Reads that display a perfect overlap of a minimum size have their corresponding nodes linked by an edge labeled by the size of the overlap. The overlaps are found using a prefix table index, that is, the reads sorted according to the lexicographical order. This sorted table is directly obtained from the binary tree in $O(N)$ time, with N for the number of reads in the non-redundant set. Two prefix tables are built, one for each of the direct and the reverse complement sequences of the non-redundant reads set. These two prefix tables allow revealing all perfect overlaps by binary search in $O[(L\text{-}m)N \log_2 N]$ time, with L for the reads length and m for the minimum overlap size. This step is straightforwardly parallelized. To identify and remove transitive edges, we implemented the transitive reduction algorithm described in Myers (2005), which efficiently performs the task in linear time. Note that the overlapping step and read loading procedure require all reads to be the same length (as Illumina reads are). This requirement greatly simplifies these processes, as it avoids the need to consider included reads.

2.3 The overlaps string graph structure

A transitively reduced overlaps graph is a complex structure with a node for every single read from the non-redundant reads set. This structure is first simplified by removing short dead-ends and bubble that are caused by polymorphisms and sequencing errors. Then, the graph is simplified, without loss of information, by condensing unambiguous paths into single nodes. The obtained structure corresponds to the overlaps graph analog version of the repeat graph that was first described in Pevzner and Tang (2001). Other structures that follow the same principle are the string graph (Myers, 2005) and the unipath graph (Butler *et al*., 2008). Such condensed graph structures represent sequences that can be unambiguously assembled from the reads given a minimum overlap size or *k*-mer size. More importantly, it defines the building blocks from which repeats are made (Pevzner *et al*., 2004).

This condensing operation is achieved while conserving the read layout information. Every node stores a layout of reads, which can be seen as a multiple un-gapped alignment. This layout is encoded as a linked list, where each element of the list contains the read identifier, its direction as well as its offset distance in the node. This structure tracks every single read in the graph, which in turn provides support for the pairing information as well as the sequencing coverage. Moreover, it has the potential to directly represent symbol ambiguities. We call the obtained condensed and transitively reduced graph an overlaps string graph (OSG). This structure encodes sequences as nodes, whereas edges represent overlaps between sequences. A particularity of the OSG over the other mentioned structures is that it preserves the overlaps information between the sequences, as well as between individual reads, which is required for the contextual cleaning procedure described in next section.

Actually, the OSG is constructed as a bi-directed graph, that is, edges have an arrowhead at each end that can be either oriented in or out the node. To be valid, a path must go through nodes either from an in-arrow to an out-arrow or from an out-arrow to an in-arrow. The first case means that the sequence in the node is considered in its direct orientation, whereas the second in its reverse complement orientation.

2.4 Genome-incoherent edges removal using contextual coverage information

Short genomic repeats whose length is smaller than the reads length and larger than the minimum overlap size *m* may cause genome-incoherent edges (i-edges). Such edges are misleading because the sequence they represent has no occurrence in the target genome. They act as a shortcut that connects the instances of the repeat (Fig. 1).

We present a statistical method to automatically identify most of the i-edges in an OSG. This operation allows the reduction of the graph complexity, which further facilitates the assembly process and analysis. The method is based on two specificities of the i-edges. The first and more important one is statistical; we exploit the fact that the size of the i-edges is bounded by the size of the repeat, whereas the size of relevant edges directly depends on the achieved sequencing coverage (Fig. 2). This means that in sufficiently covered region, i-edges size may be significantly shorted than expected for relevant edges. The second specificity is topological; because i-edges shortcut two nodes, there should exist for each of the i-edge ends, a sibling edge that corresponds to the correct assembly.

To address the statistical criteria, we use the fact that the overhanging length between adjacent reads follows a geometric distribution. The overhanging length *oh* corresponds to the distance between the starting points of two adjacent reads in the assembly. In our context for which all reads lengths are equal, $oh = L - m$. To assess the statistical significance of a particular edge of size *ov*, we estimate a *P*-value $\mathrm{P}(X \geq L - ov)$ as follows: reads overhanging lengths *X* are sampled in the region two hundred bases around the edge. The geometric distribution parameter is given by $g = (1 + \mathrm{E}(X))^{-1}$, with E(*X*) for the mean of *X*. The *P*-value for an edge of length *ov* corresponds to $(1-g)^{L-ov}$, which is the probability of observing an edge whose size is smaller or equal to *ov* given the contextual sequencing coverage.

The contextual cleaning procedure is then performed as follows: we first compute the *P*-value for each edge in the OSG. Edges displaying a probability below a given cutoff (we use 10^{-6}) are further checked for the presence of the two correct sibling edges. If this presence is confirmed, the edge is disrupted. Once all edges have been treated, the graph can be further condensed, which results in a significantly smaller graph far more suited to subsequent analyses.

This graph cleaning procedure is specific to overlaps graphs and is not usable by assemblers that are based on the *k*-mer graph framework. These assemblers use an alternative procedure called 'read threading' that follows from the Eulerian Superpaths concept (Pevzner *et al.*, 2001). Though not directly comparable, these two cleaning procedures can, in certain cases, end up with the same results. However, our procedure is much simpler as it only involves edge removal as a graph operation, whereas read threading requires more complex graph operations (Pop, 2009). Moreover, our procedure relies on a statistical assessment, which minimizes the risk of performing a misleading operation.

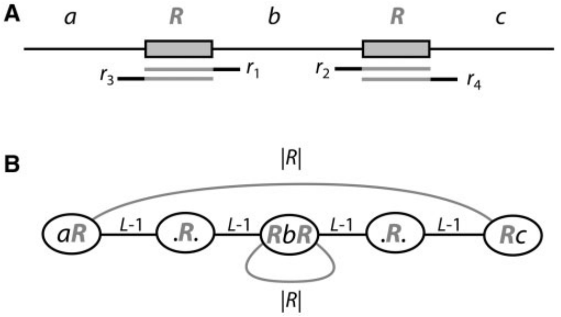

Fig. 1. This example illustrates the cause of i-edges in OSG. (**A**) A representation of a simple repeat *R* having two occurrences in a sequence *S*. $|R|$ denotes the length of *R*. The three unique segments, which can be of any length, are labeled *a*, *b* and *c*. Four reads, r_1, r_2, r_3, r_4 that partially match *R* without spanning it are displayed. In this example, we consider *L*, the reads length, to be at least $|R| + 2$. This means that the whole segment *S* could be resolved by using a minimum overlap size $m = |R| + 1$. However, if $m \leq |R|$, and provided a fully achieved coverage, the OSG depicted in (**B**) results. Two misleading edges are caused by the overlaps between reads r_1 and r_2, as well as between reads r_3 and r_4. The correct assembly goes through path $aR \rightarrow \cdot R \cdot \rightarrow RbR \rightarrow \cdot R \cdot \rightarrow Rc$. The two additional edges $aR \rightarrow Rc$ and $RbR \rightarrow RbR$ (displayed in gray) are said genome-incoherent because the resulting sequence has no occurrence in *S*. Note that the size of these edges is bounded by $|R|$. Removing these two i-edges would allow the OSG to be condensed into a single node resulting thus in a single contig for *S*

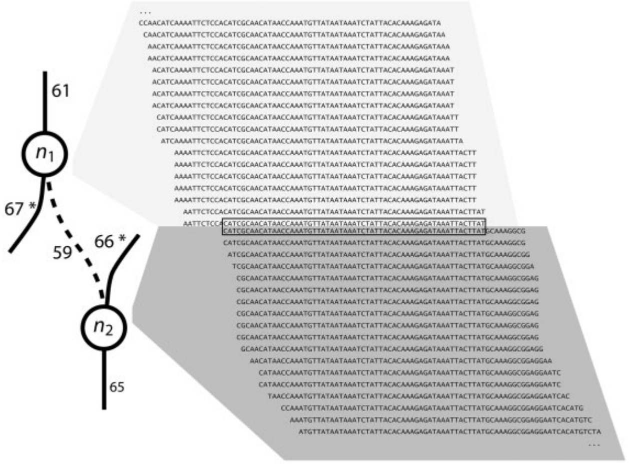

Fig. 2. A genome incoherent edge is represented within its context. This example is taken from a real dataset. Two nodes, n1 and n2, are linked by an incoherent edge (dashed). The reads layouts inside each of the nodes are represented in the gray areas. The size of the misleading overlap (boxed) is 59, which is significantly smaller than expected according to the contextual coverage. The actual probability computed for this edge is $1.3{\cdot}10^{-5}$. The two edges labeled with a star are the sibling edges that belong to the correct assembly. Note that their corresponding overlaps are not displayed on the multiple alignment. The OSG context of this example is provided in Supplementary Figure S1

2.5 Inserts lengths distribution and expected number of paired-end matches

The user typically knows the sizes of the targeted libraries. However, the mean, standard deviation and shape of the actual insert size distribution may significantly differ from what is expected. To obtain reliable values, the insert length distributions are directly obtained from the data, which exempt the user from providing this information. Inserts lengths are sampled as path distances through the OSG. Let n be the total number of pairs loaded for a given library. Each pairs is examined for possible connections in the graph within a maximum search distance h (horizon), the unit for h being the nucleotide. Because the number of explored nodes grows exponentially with h, the search is cancelled if the number of explored nodes becomes too large. Each read pair can be either: unambiguously connected, ambiguously connected, unconnected or cancelled. The inserts lengths distribution I is estimated only from unambiguously connected pairs. We define then the allowed distance range between two pairs as $\mathrm{E}(I) \pm 2 \cdot \sigma$. We say for a read pair to be matched when the two instances of the pair are connected within the allowed distance range.

For each library, we estimate the number of usable pairs u as follows: let t be the number of matched pairs that have been successfully connected and c the number of pairs for which the connection attempt has been cancelled. The number of usable pairs u is extrapolated by $u = t + c \cdot (t/n)$. From this follows the usable paired-end coverage $C^{PE} = u/T$, with T for the target genome size.

From the distribution I, the probability for a matched read pair to be separated by a distance ranging from a to b is given by $M(a,b) = \mathrm{P}(I \leq b)$ - $\mathrm{P}(I \leq a)$. Given that I may correspond to a mixture of distribution, it is not trivial here to provide an analytical solution to this probability. Instead, we performed the easy way out of this by estimating this value directly from the observed frequencies. Practically, the sampled insert lengths are stored as a cumulative table $D = [d_1, d_2, \ldots, d_n]$, for which d_i is the number of time an insert length greater or equal to i has been observed.

Let G be an OSG and let $P = (v_1, v_2, \ldots, v_{|P|})$ be a path through G made up of $|P|$ nodes. Each p_i corresponds to a node in G and some nodes can have >1 occurrence in P. For sake of simplicity and without loss of generality, we omit the nodes orientation issue and assume such paths to be valid routes in G. Let $s(P)$ be the nucleotides sequence spelled by P and $l(P)$ its length.

Suppose now a path made up by the concatenation of three sub-paths $PP = P1 \rightarrow P2 \rightarrow P3$ with $P2$ being possibly empty and $s(PP)$ having a single occurrence in the target genome. Similarly to Bresler *et al.* (2012), we estimate the expected number of paired-end matches between $P1$ and $P3$ and that spans $P2$ by summing the probability M over the possible positions in s($P1$), which gives:

$$E^{PE} = C^{PE} \sum\nolimits_{i=0}^{i=l(P1)-1} M(a+i, b+i) \qquad (1)$$

with $\mathrm{a} = l(P2)$ and $\mathrm{b} = l(P2 \rightarrow P3)$. Of course, these statistics assume that the paired-end coverage is uniform.

2.6 Layout step: using short and long paired-end to reveal paths in the OSG

Without additional information, G represents all that can be obtained from unpaired sequence reads. Each node, or pair of adjacent nodes in certain cases, will result in a contig. The paired-end assisted layout process consists in iteratively elongating a path P on top of G. P is initialized with a seed node and each iteration of the process elongates P by one node. Nodes that correspond to repeats may (and should) occur more than once in the discovered path(s).The ultimate goal is to traverse G from a single seed, which would result in a finished assembly.

The structure supporting this process is a breadth-first-like search tree T rooted at $p_{|P|}$ (Fig. 3). Such a tree represents the set of elongation candidates for P. The general principle consists of mapping the reads that are paired between P and T. The success of this approach rely on that P acts as a constraint for connecting paired reads. A long insert connection can be efficiently established to a node laying several kilobases backwards in P, whereas being computationally intractable to be searched directly through G. Because T is not required to be deeply explored, its size stays negligible. When a candidate is chosen, it is appended to P and others are pruned. The process is then iterated by building a new tree from the updated $p_{|P|}$.

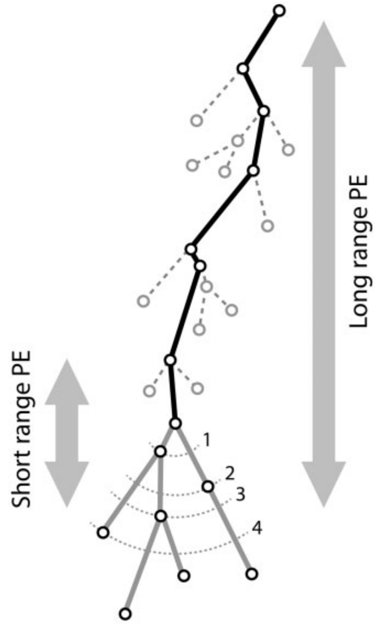

Fig. 3. Schematic representation of the search tree structure. Note that we use here a 'sequence as edge' representation, which allows displaying the length of individual sequences in a convenient way. In this representation, the length of each edge is proportional to the corresponding sequence length. The black edges define P, the path determined so far. The dotted edges are the ones that correspond to candidates that were rejected during previous iterations. The tree T rooted at $p_{|P|}$ is displayed with gray edges. T is increased by steps as represented by the four numbered dotted arcs. The program examines the paired-end matches between T and individual segments in P. This structure allows connecting paired end issued from inserts that are as long as the length of P

The tree is built by stages, which consists of adding the nearest node(s) as depicted by the numbered dashed arcs in Figure 3. This breadth-first-like building approach ensures T to be balanced in terms of paths length, which is required to avoid paired-end connections counting bias. To build the smallest tree as possible, T is progressively increased until a decision can be made. This is generally achieved during the first two stages. However, to prevent the process getting stuck in intractable computations, the size of T is limited to 500 leafs at most.

The evaluation procedure either concludes with the choice of an elongation candidate, or with an unresolvable ambiguity, in which case the elongation process is aborted. This procedure considers the paired-end connections pattern between individual nodes in P and the tree rooted at $p_{|P|}$. This pattern depends on the local repeats structure (Fig. 4). Some p_i represents repeats whose instances are associated with >1 candidate. They contain a mixture of reads issued from each instance of the repeat. In the context of resolving this repeat, such paired-end connections are said non-informative as they support >1 candidate. A given p_i is said informative if at least 95% of its paired-end matches are connected to the same candidate. This 5% tolerance is required to account for false-positive matches, mainly caused by paired reads that can be matched through more than one path.

Heavily repeated paths display an overrepresentation of paired-end matches that can mislead the elongation process and cause a misassembly. A clean normalization approach seems tricky to perform, as it would require wider contextual information. Instead, we use a safeguard that

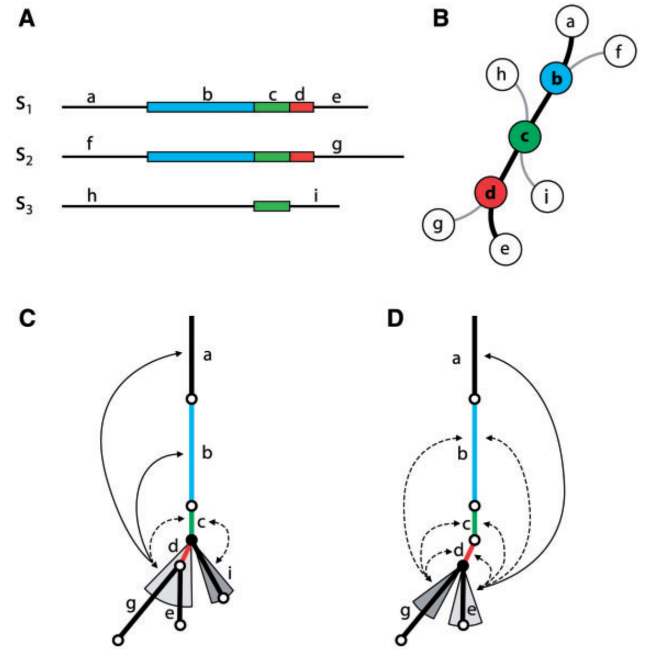

Fig. 4. A) S_1, S_2 and S_3 are three sequences that share some repeats. (**B**) The OSG resulting from the three sequences. The path corresponding to s_1 is bolded. (**C and D**) display two successive iterations for an elongation process that was initialized with node a, untangling thus the sequence S_1. Note that unlike (B, C and D) use a 'sequences as edges' representation. The plain black dot is $p_{|\mathbf{P}|}$, the end of the determined path. The paired-end connections between the elongation candidates and the determined path are represented with arrows. The gray slices show, for each candidate, the region that is mapped, which is ensured to be the same depth for all candidates. Plain arrows represent mapping to informative nodes, which map a single candidate by definition. Dotted arrow correspond to the paired-end mapping that is misleading for designating the correct candidate, as they involve repeats occurring both in the correct and incorrect elongation. The informative status of a node depends on the actual elongation context. Note that all the reads in a slice are considered together by the mapping procedure

consists in bounding the number of observed paired-end matches by the number of expected counts as given by the E^{PE} statistics Equation (1).

A candidate is designated only if all insert libraries agree and this is what happens in the large majority of the cases. The contrary would indicate some inconsistencies between individual insert libraries. Some disagreements can, however, occur when the numbers of observations are near cutoff values. In such cases, the conflict is usually resolved by extending the search tree a few stages more, which allows considering a larger sample. However, if the conflict remains, the elongation is stopped to prevent a possible misassembly.

The whole process is iterated until one of the three following stop conditions is triggered: (i) detection of a redundant path (see next section), (ii) reaching the maximum allowed size for T without being able to determine the correct candidate and (iii) reaching the bounds of the graph.

2.7 Detecting redundant paths and seeds choice

Any node could act as a seed but it is far more efficient to use nodes that encode a long sequence. Such nodes are likely to have a single occurrence in the genome. Moreover, initializing the search tree structure form a long seed directly allows the mapping of both short and long range paired end. The program chooses seeds in order of decreasing node length. However, nodes that have already been included in a path are not allowed anymore to act as a seed.

It is naturally expected for some nodes to be included more than once in the final assembly. To avoid assembling overlapping path, which would produce an increased genome, the program continuously checks for paths redundancies by using a node marking strategy. Let *maxd* be the maximum length allowed for a repeat. By default, this parameter is set to the maximum allowed distance for the larger inserts library. This value actually corresponds to the maximum length of a resolvable repeat. Let *d* be a nucleotide distance counter that is updated as the path is elongated. During elongation, each encountered node is flagged. The distance counter *d* is initialized to zero each time a non-flagged node is encountered. If *d* reaches a value greater that *maxd*, the elongation is interrupted. Such a case means that the elongation process is re-assembling a repeat that is at least *maxd* in length, and thus for which a resolution is hopeless given the length of the larger inserts library. By default, the elongation is interrupted just before the repeat, such that it is included only once in the final contigs set. In this way, the resulting contigs can display overlaps of at much *L-1* bases. The user can, however, override this setting to control the maximum length of ending repeats. This allows obtaining contigs that display large overlaps, which may facilitate further manual assembly operations.

This flagging strategy is a heuristic, as it does not take into account the fact that nodes can be visited in any of the two directions, depending on the elongation context. Therefore, it cannot be excluded for this procedure to prematurely interrupt a path elongation, though we never observed such a case.

2.8 An interactive mode

The Edena implementation includes an interactive shell that was initially implemented for development purpose. This shell allows investigating the OSG at targeted nodes and to possibly manually resolve some ambiguities. Among other possibilities, this shell allows visualizing some regions of the OSG, visualizing the search tree, initializing an elongation process from a provided path, while displaying the number of matching pairs for each of the candidates. It also features some querying operations such as getting the nucleotide sequence corresponding to a path, or getting sequence with marks defining target PCR amplification as suited to the Primer3 program (Untergasser *et al.*, 2012). Though not originally developed to be user-friendly, this shell provides a valuable assistance in checking and solving particular assembly issues, or to manually improve the assembly.

3 RESULTS

3.1 Assembly of the *S.aureus* strain SGH-10-168 genome

The bacterium *S.aureus* strain SGH-10-168 is a new isolate that was acquired as part of an ongoing retrospective methicillin resistant *S.aureus* surveillance program at Scripps/Green Hospital of Scripps Health and Scripps Translational Science Institute. The isolate was collected from an asymptomatic nasal carrier collected during 2010 in La Jolla, CA, USA. The closest phylogenetically related isolate for which a complete genome sequence is available is the *S.aureus* strain N315 (Kuroda *et al.*, 2001) (Genbank: BA000018.3).

Genomic DNA from strain SGH-10-168 was shotgun sequenced using an Illumina HiSeq (see supplementary information). Libraries of insert lengths ranging from 165 bp to 4 kb were produced and sequenced for a total of 71.4 million unambiguous reads (i.e. reads that do not contain any ambiguous symbol).

Details of the reads dataset and inserts length are shown in Table 1 and Supplementary Figure S1.

3.1.1 Assembly and manual finishing We launched Edena using all six datasets. The redundancy filter reduced to dataset to 26.0 million of unique reads. The minimum overlap size was set to 60. A total of nine contigs were produced. Among the contigs, five orphan contigs without any connection to other contigs were discarded. These contigs were confirmed to be artifacts using the Blast program (Altschul *et al.*, 1997) against the non-redundant database as all reported matches corresponding to eukaryotic sequences. Among the four remaining contigs, two were identified as circular and correspond to plasmids. The two remaining contigs display size of 1.167 and 1.649 Mbp, respectively. Using the interactive mode of Edena, we investigated the contigs end in the context of the OSG. We found that both ends of the two contigs are connected to a single node that is 5.3 kb in length (Supplementary. Fig. S3). This particular node corresponds to a repeat that occurs twice in the genome, each instance being reverse complemented. There are thus two possible layouts for joining the two remaining contigs into a single circular sequence. The first one would include the two contigs in the same orientation, whereas the second one would include one of the contig reverse complemented relatively to the other one. Unfortunately, because no read pair spans the 5.3 kb repeat, it is not possible to formally address this using the reads pairing information. It was, however, straightforward to determine the correct orientation of these two contigs by examining the GC skew resulting from the replication bias in bacteria (Grigoriev, 1998) (Supplementary Fig. S4). The two contigs were then assembled into a single circular sequence that is 2.82 Mbp in length.

To assess the accuracy of SGH-10-168, we choose the closest related isolate in which there was a publically available complete genome that was determined to be the *S.aureus* strain N315 (Kuroda *et al.*, 2001) (Genbank: BA000018.3). We compared the two genomes with the MUMmer software package (Kurtz *et al.*, 2004) together with the Circos visualization engine (Krzywinski *et al.*, 2009) and observed a quasi-perfect synteny between the two genomes (Fig. 5). A surprising exception is found in a 48 kb inversion (genome coordinates 42.5–87.2 kb) corresponding to the methicillin resistance cassette. This inversion causes a GC-skew jump (Supplementary Fig. S3) suggesting either a misassembly, or that this structural alteration occurred recently relative to the tip of the phylogenetic tree.

Using the interactive mode of Edena, we investigated the paired end spanning the inversion boundaries and found that thousands of pairs agree with this inversion (Supplementary Fig. S3). We performed some PCR experiments to assess the orientation of the 48 kb fragment in an independent way. The result confirmed the orientation of the 48 kb segment, as assembled by Edena (results available on request).

We also observed translocations for the 5.3 kb repeat that has two occurrences in SGH-10-168. N315 displays the same two occurrences, plus three more that are absent in SGH-10-168. AQ6/>This element corresponds to Tn544 transposon, as annotated in the N315 genome sequence. It is, therefore, not surprising to observe such a variation between the two strains. Though this comparison cannot provide a fine assessment, it demonstrates that the assembled genome is consistent with what could be expected.

Table 1. Reads dataset that were obtained for the strains SGH-10-168 and MW2

Library	PE ($\times 10^6$)	TIS	OIS: mean (sd)	UP ($\times 10^6$)
SGH_1	8.6	165 bp	158 (17)	7.5
SGH_2	4.9	275 bp	274 (34)	3.2
SGH_3	6.4	2.5 kb	2497 (344)	2.1
SGH_4	8.0	3.0 kb	3357 (716)	2.4
SGH_5	3.8	3.0 kb	3415 (579)	1.1
SGH_6	3.9	4.0 kb	3971 (942)	1.0
MW2_1	3.7	400 bp	427 (72)	2.9
MW2_2	1.8	5.0 kb	4171 (524)	0.5
MW2_3	1.6	10.0 kb	5550 (787)	0.4

Note: Column legend: 'PE' is the number of pairs (number of reads divided by two). TIS is the insert size that was targeted during libraries preparation and OIS is the observed insert size, as automatically sampled during the assembly process. 'UP' is an estimate of the number of usable pair. Note that the target and observed insert size for MW2_2 differs significantly, which is probably because of a technical bias during the library production.

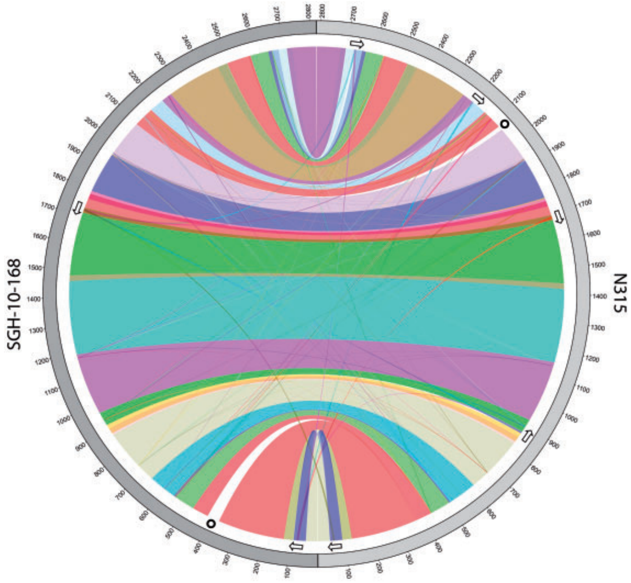

Fig. 5. Complete genomes mapping between strains SGH-10-168 and N315. Strain SGH-10-168 is displayed on the left side, whereas N315 is on the right. Colored ribbons indicate regions of high similarity as revealed by the NUCmer script from the MUMmer software package. The grey levels are randomly displayed and do not correspond to a particular scheme. A minimum seed match size of 1 kb has been used. The two black circles indicate two large insertions. The arrows indicate the occurrences and orientations of the Tn544 transposon, which has two occurrences in SGH-10-168 and five occurrences in N315. The 48 kb inversion contain Tn544 and is depicted by the blue ribbon located at the bottom of the figure (SGH-10-168 genome coordinates 41722:87958)

3.1.2 Assemblies using datasets downsampled at various coverage The use of all six SGH-10-168 datasets results in an untypically high raw coverage of about 2500×. To assess Edena in more standard conditions, we performed multiple assembly combinations. More precisely, we included SHG_2 alone or in combination with any of the long insert SGH_3, SGH_4 and SGH _6. We also evaluated assemblies including SGH_2, 3, 4 and 6 together. Moreover, we performed each of these assemblies at 70, 140, 210 and 280× coverage values. This was done by randomly downsampling the SGH-10-168 datasets, such that the total raw coverage for a given assembly reaches the desired value and that each dataset included in the assembly contains the same number of reads. Assembly statistics are presented in Table 2 and Nx plots as Supplementary Figure S5. Results show that including long insert libraries clearly improves the assembly contiguity. There are, however, some differences between long insert libraries. The longest one, SGH_6 that is 4 kb long, does not perform as well as other large libraries. This is counter intuitive as one would expect the longest library to perform better. This lower performance is likely due to the fact that this library distribution displays the largest standard deviation (Supplementary Fig. S2), which increases the probability for wrong positive paths to be validly connected. Interestingly, the use of all four datasets almost always improves the assemblies. At 140, 210 and 280×, assemblies result in higher N50 and N75 values as compared with using only two datasets. This improvement can be explained by some complementarity between individual libraries. SGH_3 and 4 may be able to resolve some parts where the larger one fails because of its high standard deviation. On the other hand, SGH_6 may be able to resolve a few points requiring a 4 kb insert size, where other shorter libraries fail. At 70× coverage, the use of all four libraries is outperformed by other combinations. In this case, individual library coverage values drop to ~18×, which become borderline regarding the minimum required number of paired-end connections. This indicates that the coverage of individual libraries should not be <35×. However, even at 280× coverage, the contiguity is decreased as compared with the use of all six datasets with full coverage. Such a high coverage was probably required, on some particular parts of the assembly graph, to compensate for the weak support provided by the long insert libraries due to the number of usable pairs (see Table 1).

Table 2. Assembly statistics for various downsampled SGH-10-168 datasets

Cov	Downsampled PE datasets	Number of seq	% ref	N50	N75
70×	SGH_2	33	98.8	244	160
	SGH_2, SGH_3	25	99.1	433	190
	SGH_2, SGH_4	21	99.2	358	190
	SGH_2, SGH_6	22	99.1	296	171
	SGH_2, SGH_3, SGH_5, SGH_6	51	98.2	169	123
140×	SGH_2	33	98.8	244	160
	SGH_2, SGH_3	13	99.4	649	278
	SGH_2, SGH_4	13	99.4	650	279
	SGH_2, SGH_6	14	99.4	650	224
	SGH_2, SGH_3, SGH_5, SGH_6	13	99.3	846	224
210×	SGH_2	33	98.8	244	160
	SGH_2, SGH_3	12	99.4	650	277
	SGH_2, SGH_4	13	99.4	650	279
	SGH_2, SGH_6	12	99.5	650	230
	SGH_2, SGH_3, SGH_5, SGH_6	9	99.5	848	279
280×	SGH_2	33	98.8	244	160
	SGH_2, SGH_3	11	99.4	650	277
	SGH_2, SGH_4	11	99.4	650	279
	SGH_2, SGH_6	11	99.4	650	231
	SGH_2, SGH_3, SGH_5, SGH_6	11	99.4	651	279

Note: Contigs associated with non-bacterial sequences and contigs <200 pb have been removed before evaluation. 'cov' is the total reads coverage, 'Number of seq' is the number of contigs, '% ref' is the percentage of the complete SGH-10-168 genome that is covered by the contigs. N50 and N75 values are given in bp.

3.2 Assembly of *S.aureus* strain MW2 genome

To provide an additional assessment of the assembly method, we resequenced the *S.aureus* strain MW2 (Baba *et al.*, 2002) for which a high-quality reference sequence is available (GenBank: BA000033.2). Genomic DNA was shotgun-sequenced using an Illumina MiSeq with the Nextera protocol (see supplementary information). Three libraries of short and long inserts were produced for a total of 14.1 million of unambiguous reads. Details of the sequencing libraries and inserts length are shown in Table 1 and Supplementary. Figure S1.

We ran Edena using the three libraries with a minimum overlap size to 60 bp. The redundancy filter reduced to dataset to 5.96 million of unique reads. A total of nine contigs summing up to 2.84 Mbp were produced. Contig #6 was identified by Edena as circular and corresponds to the pMW2 plasmid (GenBank: AP004832.1).

We compared the assembled contigs with the MW2 reference genome with MUMmer and Exonerate (Slater and Birney, 2005) software packages. After allowing several small contigs that corresponded to repetitive regions to map to multiple locations, the assembly covered 100% of the MW2 reference. The comparison displays five discrepancies, which we refer to as D1 to D5. The first one, D1, is a small rearrangement within a 1 kb region located at coordinates 2799000:2799941 (collagen adhesin precursor). Discrepancies D2–D5 all occur in ribosomal RNA genes. D2 consists in two short segments (537415:537482 and 542609:542693) that are swapped. The three remaining are single nucleotide difference: D3 (496372 A→T), D4 (541998 C→T) and D5 (2253486 T→C). Using the interactive mode of Edena, we investigated the five discrepancies and found no sign of misassembly. We performed PCR and Sanger sequencing at region D1 and confirmed that the Edena assembly was correct (results available on request). Because the three remaining discrepancies are located within repeats several kilobases long, we were unable to verify using standard laboratory procedures. Instead, we assessed the assembled contigs as well as the MW2 reference sequence by examining the paired-end concordance at regions D2–D5. Using the Burrows-Wheeler Aligner (Li and Durbin, 2009) and Samtools (Li *et al.*, 2009), we mapped the long range paired-end reads onto the two genome sequences. For the four discrepancies, the mapping across the MW2 reference assembly caused read pair discordances, whereas the Edena assembly showed concordant mapping across the sites. Importantly, because D2–D5 occurs within large repeats, only the long range paired ends are able to reveal these assembly discordances in the MW2 genome.

It is unlikely that the two long insert libraries are flawed at these particular regions. Moreover, considering the fact that D2–D5 occur within repeated clusters of transfer RNA and ribosomal RNA genes, which are known to be difficult to resolve using Sanger sequencing, it would not be that surprising for the MW2 reference sequence to displays these few assembly errors. Therefore, we believe the Edena assembly is correct.

3.3 Comparison with other *de novo* assemblers

Using the SGH-10-168 and MW2 datasets, we compared Edena (v 3.130110) with the four other *de novo* assemblers that are able to use long paired-end data: SOAPdenovo (version 2-src-r223), ABYSS (v.1.3.5), Ray (version 2.2.0) and ALLPATHS-LG (version 45697). Because ALLPATHS-LG requires at least one short insert library for which reads from a pair overlap, it was only suited for assembling the SGH-10-168 dataset. SOAPdenovo was also run with the GapCloser module that allowed filling in most of the gaps that remain in scaffolds. Each program was run using a wide range of *k*-mer size parameter except for ALLPATHS-LG, which was run using default parameter, as recommended by the user manual. All programs were run by taking advantage of their parallelization abilities, by involving 22 threads (or processes for Ray that uses the Message Passing Interface). We report, for each of the programs, the assembly that maximized the N50 value. Before evaluation we cleaned contigs and scaffolds associated with non-bacterial sequences and removed sequences shorter than 200 bp. Except for Edena that does not currently implement a scaffolding step, contigs and scaffolds are evaluated separately.

3.3.1 S.aureus SGH-10-168 In the absence of a suitable reference genome, the SGH-10-168 assemblies were only assessed for global structure against the N315 complete genome sequence, in the same way as performed in Section 3.1. Because all programs agreed with the 48 kb inversion described in Section 3.1, we do not report it as an assembly error.

Edena and Ray produce contigs significantly larger than other assemblers, which is due to the fact that both programs use short and long paired-end reads at the contig stage, whereas others use long paired-end reads at the scaffolding stage only. Edena was the only assembler tested to not introduce a structural discrepancy, whereas the other assemblies all contained at least one independent error. Ray assembled a 650 kb inversion. ALLPATHS-LG scaffold contained a large inversion and a deletion, SOAPdenovo included one of the plasmid at the end of a genomic scaffold whereas ABYSS displayed a large deletion. The summarized results are presented in Table 3. Cumulative and Nx contiguity plot are provided in Supplementary Figure S6.

3.3.2 S.aureus MW2 With the aid of a high quality published reference sequence, we were allowed a more in depth assessment of the assemblies for the MW2 datasets. We compared the produced contigs and scaffolds against the reference sequence using the Quast assemblies assessment tool (Gurevich *et al.*, 2013) version 2.3pre-release. Importantly, we observed that no assembly was in disagreement with Edena concerning the D1–D5 discrepancies reported in Section 3.2, which further confirms these discrepancies to be assembly errors in the MW2 reference sequence. Therefore, they were not accounted in the comparisons. Table 4 summarizes the results provided by Quast. This evaluation tool reports two kinds of misassemblies according to the distance between the alignments that flank the misassembly breakpoint. Briefly, a distance $\geq$1 kb is reported as an extensive misassembly, whereas a distance <1 kb as a local one. We report the sum of extensive and local misassemblies for the contigs, whereas scaffolds are assessed only for extensive ones. Beside the widely used N50 metric, Quast provides the NGA50 metrics, which is computed by breaking the contigs at every misassembly. Assemblies cumulative and Nx plots, which provide a more detailed picture of the assemblie's contiguity, are given in Supplementary Figure S7.

The SOAPdenovo Gap Closer module provides the best raw contiguity with a N50 value of 2624 kb. This assembly, however, contains 21 misassemblies causing the NGA50 metric to drop to 343 kb. A single false-positive local misassembly is reported for Edena. It is caused by a complete contig originating from a plasmid, which displays a 99 bp self-overlap. In the same way, one of the misassembly reported for SOAPdenovo and Ray is also due to the plasmid. ABYSS did not display this particular misassembly because it did not report the MW2 plasmid, which in turn explain the lower reference coverage (99.09%) reported for this program. Other assemblies display a reference coverage that is close to 100%. This value is not reached because some parts of the genome were assembled in contigs <200 pb, which are discarded from the evaluation. Curiously, the total number of nucleotides assembled by Ray exceeds the expected sum by ~10%. This surplus is because of large redundant sequences in the assembly.

3.4 Program and sequencing data availability

Edena is freely available at http://www.genomic.ch/edena. The sequencing reads are available at the NCBI Sequence Read Archive (SRA, http://www.ncbi.nlm.nih.gov/Traces/sra), under the accession numbers SRA059350 for the strain SGH-10-168 and SRA080268 for the strain MW2. The produced assemblies are available at http://www.genomic.ch/edena/results2013/.

4 DISCUSSION

We present a *de novo* assembly approach that succeeded in obtaining a finished genome sequence of a new *S.aureus* isolate, a public health implied pathogen. The assembly was performed using 100 bp paired-end reads obtained from the broadly available Illumina platform. Only minor human intervention was required. The presented approach is implemented in the publicly available Edena software application. Our method relies on simultaneously exploiting the information provided by short and long paired-end sequence reads to discover paths in the assembly graph. The process uses a search tree structure that is progressively pruned. Our approach works for inserts of potentially any length by constraining possible paths. Such long paired-end connections can be intractable to establish directly through the assembly graph, as the number of possible paths increases exponentially with the searched distance. We present the search tree structure in the context of an overlaps graph resulting from 100 bp reads. However, it is in all likelihood

Table 3. Comparative assembly results for the *S.aureus* SGH-10-168 isolate

SGH-10-168	Contigs (kb)					Scaffolds (kb)					Hardware	
Program name (*k*)	Number of seq	Sum	N50	Max	Number of large discr	Number of seq	Sum	N50	Max	Number of large discr	CPU	Mem
Edena (60)	4	2846	1649	1649	0	n/a	n/a	n/a	n/a	n/a	3.8	19.3
SOAPdenovo (91)	100	2820	88.6	211	0	18	2842	1484	1484	1	0.95	22.8
SOAPdenovo + GC (91)	22	2839	608	1008	0	18	2843	1487	1487	1	1.0	22.8
ABYSS (81)	34	2861	210	387	0	12	2848	693	1123	1	5.2	2.2
Ray (21)	7	2822	2252	2252	1	7	2822	2252	2252	1	278	43.9
ALLPATHS-LG (auto)	42	2853	170	550	0	19	2873	1990	1990	2	38.6	27.3

Note: Contigs and scaffolds metrics as well as hardware usages are presented for each program. 'SOAP + GC' refers to SOAPdenovo scaffolds post-processed by the GapCloser module. Note that Edena scaffold metrics are not available (n/a), as it does currently not include a scaffolding step. 'Number of seq' is the number of contigs after the removal of non-bacterial contamination. 'Sum' and 'Max' are the total assembly size and the largest contig length, respectively. 'Number of large discr' reports the number of large structural discrepancies with the genome sequence of the closest isolate, which are likely to correspond to assembly error. 'CPU' and 'Mem' are the CPU-time (h) and the memory usage (Gb), respectively.

Table 4. Comparative assembly results for the *S.aureus* strain MW2 as reported by QUAST

MW2	Contigs (kb)							Scaffolds (kb)						Hardware	
Program name (*k*)	Number of seq	Sum	N50	NGA50	Max	Number of tot mis	% ref	Number of seq	Sum	N50	NGA50	Max	Number of ext mis	CPU	Mem
Edena (60)	9	2840	622	622	1150	1	99.96	n/a	n/a	n/a	n/a	n/a	n/a	0.8	3.74
SOAP (93)	101	2817	93	93	267	1	99.01	11	2842	2816	2784	2816	2	0.17	10.9
SOAP + GC (93)	12	2846	2642	343	2642	21	99.86	11	2846	2821	1859	2821	6	0.2	10.9
ABYSS (87)	16	2833	384	384	1228	0	99.09	4	2821	2601	1427	2601	2	1.0	0.9
Ray (35)	11	3136	755	267	823	12	99.97	7	3155	823	755	843	5	42.4	7.9

Note: Assembly metrics as provided by QUAST. '% tot mis' is the number of local and extensive misasssemblies, 'Number of ext mis' is the number of extensive misassemblies only. NGA50 is the N50 metrics computed from the contigs broken at every misassembly '% ref' is the percentage of the genome that is covered by the contigs. Remaining columns are the same as in Table 3.

generalizable to other approaches as well, such as the DeBruijn graph or graphs resulting from longer sequence reads. The used statistics as well as the algorithm that integrates the paired-end connections to select the correct elongations can be further developed. For example, an avenue worth exploring would be to measure the distribution drift that can occur with false-positive paired-end connections.

Sequence accuracy is a major concern and we developed Edena by prioritizing accuracy over contiguity. Being able to have confidence at the single nucleotide level of an assembly removes many of the downstream complications during analysis. Even if no *de novo* assembler can be claimed to be perfectly reliable in all situations, Edena achieved a high level of accuracy, as demonstrated by the assembly of *S.aureus* strain MW2 data. Moreover, all suspicious cases we checked with independent methods were finally settled in favor of the assembly produced by Edena.

The contextual i-edges cleaning procedure significantly reduces the parameterization effort for the end users. A satisfactory, if not optimal, assembly can often be obtained with default parameters. By default, the minimum overlap size is set to half of the reads length, which is conservative. However, according to the achieved coverage, it is worth increasing this setting, which acts as a first rough-hewing and facilities further processing. For example, given the high coverage achieved for the SGH-10-168 and MW2 projects, we set the minimum overlaps value to be 60. However, the same or similar assemblies could also be obtained with default parameters, or with a larger value such as 80. This makes Edena particularly suited for batch automatic assembly of numerous bacterial genomes, for which individual parameterization is not feasible.

Even if the required CPU-time and memory displays significant differences between the programs, this requirement was a limitation for none of them. It is important mentioning that these requirements may strongly depend on the parameterization and particularly on the *k* parameter. We did not optimize the programs settings regarding the hardware requirement, but only regarding the assemblies' contiguity.

It is likely that not all bacterial genomes could be so easily assembled. Each sequencing project is an individual story. A recurrent issue is the tandem repeats that require sharp inserts distribution to be properly resolved. The quality of the sequencing data and particularly the accuracy of the inserts length are determinant parameters for a successful assembly. It is, however, possible provided human intervention, to manually solve many assembly ambiguities by investigating the paired-end mapping together with the node coverage and graph topology. The interactive shell of Edena was initially implemented for the development purpose but has shown to be an invaluable tool in the process of checking particular regions and in producing a finished genome.

ACKNOWLEDGEMENT

Most computations were performed at the Vital-IT (http://www.vital-it.ch) Center for high-performance computing of the SIB Swiss Institute of Bioinformatics.

Funding: Swiss National Science Foundation (3100A0-112370/1 to J.S.).

Conflict of Interest: none declared.

REFERENCES

Altschul,S.F. *et al.* (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res.*, **25**, 3389–3402.

Baba,T. *et al.* (2002) Genome and virulence determinants of high virulence community-acquired MRSA. *Lancet*, **359**, 1819–1827.

Boetzer,M. and Pirovano,W. (2012) Toward almost closed genomes with GapFiller. *Genome Biol.*, **13**, R56.

Boisvert,S. *et al.* (2010) Ray: simultaneous assembly of reads from a mix of high-throughput sequencing technologies. *J. Comput. Biol.*, **17**, 1519–1533.

Bresler,M. *et al.* (2012) Telescoper: *de novo* assembly of highly repetitive regions. *Bioinformatics.*, **28**, i311–i317.

Butler,J. *et al.* (2008) ALLPATHS: *de novo* assembly of whole-genome shotgun microreads. *Genome Res.*, **18**, 810–820.

Gnerre,S. *et al.* (2011) High-quality draft assemblies of mammalian genomes from massively parallel sequence data. *Proc. Natl Acad. Sci. USA*, **108**, 1513–1518.

Grigoriev,A. (1998) Analyzing genomes with cumulative skew diagrams. *Nucleic Acids Res.*, **26**, 2286–2290.

Gurevich,A. *et al.* (2013) QUAST: quality assessment tool for genome assemblies. *Bioinformatics*, **29**, 1072–1075.

Hernandez,D. *et al.* (2008) *De novo* bacterial genome sequencing: millions of very short reads assembled on a desktop computer. *Genome Res.*, **18**, 802–809.

Krzywinski,M. *et al.* (2009) Circos: an information aesthetic for comparative genomics. *Genome Res.*, **19**, 1639–1645.

Kuroda,M. *et al.* (2001) Whole genome sequencing of meticillin-resistant *Staphylococcus aureus. Lancet*, **357**, 1225–1240.

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

Li,H. and Durbin,R. (2009) Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics.*, **25**, 1754–1760.

Li,H. *et al.* (2009) The sequence alignment/Map format and SAMtools. *Bioinformatics*, **25**, 2078–2079.

Li,R. *et al.* (2010) *De novo* assembly of human genomes with massively parallel short read sequencing. *Genome Res.*, **20**, 265–272.

Myers,E.W. (2005) The fragment assembly string graph. *Bioinformatics*, **21** (**Suppl. 2**), ii79–ii85.

Pevzner,P.A. and Tang,H. (2001) Fragment assembly with double-barreled data. *Bioinformatics*, **17** (**Suppl. 1**), S225–S233.

Pevzner,P.A. *et al.* (2001) An Eulerian path approach to DNA fragment assembly. *Proc. Natl Acad. Sci. USA*, **98**, 9748–9753.

Pevzner,P.A. *et al.* (2004) *De novo* repeat classification and fragment assembly. *Genome Res.*, **14**, 1786–1796.

Pop,M. (2009) Genome assembly reborn: recent computational challenges. *Brief Bioinform.*, **10**, 354–366.

Ribeiro,F. *et al.* (2012) Finished bacterial genomes from shotgun sequence data. *Genome Res.*, **22**, 2270–2277.

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

Slater,G.S. and Birney,E. (2005) Automated generation of heuristics for biological sequence comparison. *BMC Bioinformatics*, **6**, 31.

Tsai,I.J. *et al.* (2010) Improving draft assemblies by iterative mapping and assembly of short reads to eliminate gaps. *Genome Biol.*, **11**, R41.

Untergasser,A. *et al.* (2012) Primer3–new capabilities and interfaces. *Nucleic Acids Res.*, **40**, e115.

Wetzel,J. *et al.* (2011) Assessing the benefits of using mate-pairs to resolve repeats in *de novo* short-read prokaryotic assemblies. *BMC Bioinformatics*, **12**, 95.