Genome analysis

BOSS: a novel scaffolding algorithm based on an optimized scaffold graph

Junwei Luo¹,², Jianxin Wang¹*, Zhen Zhang¹, Min Li¹ and Fang-Xiang Wu³

¹School of Information Science and Engineering, Central South University, ChangSha 410083, China, ²College of Computer Science and Technology, Henan Polytechnic University, JiaoZuo 454000, China and ³Division of Biomedical Engineering, University of Saskatchewan, Saskatchewan S7N 5A9, Canada

*To whom correspondence should be addressed.

Received on April 18, 2016; revised on June 22, 2016; accepted on September 8, 2016

Abstract

Motivation: While aiming to determine orientations and orders of fragmented contigs, scaffolding is an essential step of assembly pipelines and can make assembly results more complete. Most existing scaffolding tools adopt scaffold graph approaches. However, due to repetitive regions in genome, sequencing errors and uneven sequencing depth, constructing an accurate scaffold graph is still a challenge task.

Results: In this paper, we present a novel algorithm (called BOSS), which employs paired reads for scaffolding. To construct a scaffold graph, BOSS utilizes the distribution of insert size to decide whether an edge between two vertices (contigs) should be added and how an edge should be weighed. Moreover, BOSS adopts an iterative strategy to detect spurious edges whose removal can guarantee no contradictions in the scaffold graph. Based on the scaffold graph constructed, BOSS employs a heuristic algorithm to sort vertices (contigs) and then generates scaffolds. The experimental results demonstrate that BOSS produces more satisfactory scaffolds, compared with other popular scaffolding tools on real sequencing data of four genomes.

Availability and Implementation: BOSS is publicly available for download at https://github.com/bioinformaticsCSU/BOSS.

Contact: jxwang@mail.csu.edu.cn

Supplementary information: Supplementary data are available at Bioinformatics online.

1 Introduction

As the increasing availability of next-generation sequencing technology, many de novo genome assemblers were developed for reconstructing complete and correct genome sequences (Gnerre et al., 2011; He et al., 2013; Luo et al., 2015a). Genome assemblers usually first produce a large number of fragmented contigs. Then, scaffolding in the pipeline of assembly takes the contigs and paired reads as input to produce some scaffolds (Li et al., 2016). A scaffold consists of oriented and ordered contigs. These scaffolds could benefit downstream analysis such as gene order, comparative or functional genomics and patterns of recombination (Hunt et al., 2014).

Most existing scaffolding tools use scaffold graphs as the foundation for scaffolding. In a scaffold graph, a vertex represents a contig, an edge between two vertices represents the relative orientations and orders of them. When constructing a scaffold graph, there are two ways to decide whether an edge should be added between two vertices. One common way is based on the number of paired reads that link two contigs, if that number is larger than a threshold (defined by scaffolding tools or users), an edge is added and the edge weight is equal to that number. Another way is based on the distribution of paired reads that link two vertices. BESST (Sahlin et al., 2014) uses the standard deviation of gap distance and read position...
distribution to decide whether an edge between two vertices should be added.

Due to repetitive regions in genome, sequencing errors and uneven sequencing depth, two non-adjacent contigs possibly have paired reads that link them. As a result some spurious edges, which increases the difficulty of scaffolding, may be introduced in a scaffold graph. The spurious edges usually cause some contradictions among the orientations or orders of contigs which are derived from different edges (Bodily et al., 2016). There are two ways to avoid the negative effect of spurious edges. One way adopts heuristic algorithms to select paths from the scaffold graph that maximize the weight of edges on the paths. SSPACE (Boetzer et al., 2011) uses a greedy heuristic algorithm to select paths. OPERA (Gao et al., 2011) utilizes the dynamic programming to choose paths. BESST (Sahlin et al., 2014) adopts the breadth first search to pick up the path with the maximum weights of edges connecting two larger contigs. ScaFFMatch (Mandric and Zelikovskiy, 2015) proposed a novel optimization formulation representing scaffolding as a maximum-weight acyclic 2-matching problem. Another way transforms the detection of spurious edges as finding a set of edges that minimize the weight of edges and whose removal can lead to no contradictions in the scaffold graph. SCARPA (Donmez and Brudno, 2013) and SILP2 (Lindsay, 2014) adopt the linear programming to remove spurious edges. MIP (Salma et al., 2011) segments the scaffold graph into small sub-graphs, and uses the mixed integer programming to detect and remove spurious edges.

There are two problems which prevent most current scaffolding tools from getting more accurate results. (1) If the weight of an edge is equal to the number of paired reads, the edges among vertices which come from low depth sequencing regions are possibly regarded as spurious. Using the distribution of paired reads can partly resolve problems caused by repetitive regions and uneven sequencing depth. To our knowledge, BESST (Sahlin et al., 2014) is the only scaffolding tool using the distribution of paired reads for constructing a scaffold graph, but BESST only considers paired reads that link two contigs to analyze the distribution and still sets the weight to be the number of paired reads. Yet, the paired reads whose only one mate read is mapped to contigs are also helpful to analyze the reliability of edges. (2) Existing methods prefer to consider all the edges simultaneously to detect spurious edges in scaffold graphs. However, the high weight edges are more possibly correct. It is appealing to develop a new method to take advantage of orientation and order information about high weight edges to detect the spurious edges.

In this paper, we utilize two new ideas to address these two problems. To address problem (1), for two contigs which have paired reads that link them, if one read is mapped to one contig, then we can infer the probability that its mate read can be mapped to another contig based on the distribution of insert size. Based on this idea, for two contigs, we consider all reads mapped to one contig no matter whether their mate reads can be mapped to another contig or not, and develop a statistical method to compute the expectation number of paired reads that link them. Then, the expectation number is compared with the real number to get a score which is used to decide whether an edge should be added or not, and the score is set to be the edge weight. In case that two adjacent contigs come from low depth sequencing regions, the edge weight still has a chance to be high by this method. If two non-adjacent contigs have paired reads that link them caused by repetitive regions, the expectation number and the real number of paired reads commonly differ greatly. Therefore, our statistical method can more accurately judge whether an edge between two vertices should be added and how the edge should be weighed. To address problem (2), we adopt an iterative strategy to detect and remove spurious edges. In the first iteration, we extract a sub-graph from the scaffold graph which is induced from edges with high weight, then we iteratively add the rest of edges to the sub-graph from high to low weight. In each iteration, we detect and remove spurious edges based on the sub-graph. The edges which have been confirmed to be non-spurious are used as prior information to guide the removal in subsequent iterations. The orientation and order information about high weight edges can be utilized in this iterative strategy.

Based on these two new ideas, we present a novel scaffolder BOSS (Building Optimized Scaffold graph for Scaffolding) to determine orientations and orders of contigs. BOSS is compared with other popular scaffolding tools on four real datasets, the experimental results demonstrate BOSS can generate more satisfactory scaffolds.

2 Notation

In this paper, a paired read \( pr \) is referred as two reads with opposite orientations which are sequenced from two ends of a particular fragment, \( lr(pr) \) is the left read of \( pr \) while \( rr(pr) \) is the right read of \( pr \). The length of the fragment is called the insert size which approximately follows a normal distribution \( N(\mu, \sigma) \) (Luo et al., 2015b). If a read \( r \) is forwardly mapped to a contig \( ci \), \( crd(\%r, \%ci) \) is the distance between the start mapping position of \( r \) and 3-end of \( ci \). If the read \( r \) is reversely mapped to \( ci \), \( crd(\%r, \%ci) \) is the distance between the start mapping position of \( r \) and 5-end of \( ci \). The gap distance between \( ci \) and \( cj \) is \( gd_{\%ci, \%cj} \). For a paired read \( pr \) that links \( ci \) and \( cj \), we can get the \( crd(lr(pr), ci) \) and \( crd(rr(pr), ci) \) (shown in Fig. 1).

3 Methods

BOSS uses one or more paired read libraries for scaffolding. Each paired read library is stored in two FASTQ files. Before scaffolding, a set of contigs should be generated by an assembler. Each paired read library is mapped to the set of contigs, and the mapping information is stored in a BAM file. Then, BOSS takes the set of contigs and BAM files as input. The procedure of BOSS is outlined as follows: (i) Preprocessing: BOSS first analyzes the BAM files and filters out some ambiguous mapping information. (ii) Constructing an optimized scaffold graph; BOSS utilizes a statistical method to calculate the expectation number of paired reads that link two contigs. Based on the ratio of the expectation number to the real number, BOSS decides whether an edge should be added and how the edge should be weighed. Furthermore, BOSS uses an iterative strategy to detect and remove spurious edges. (iii) Sorting vertices in the scaffold graph; BOSS adopts a heuristic algorithm to sort vertices thus generating scaffolds.

3.1 Preprocessing

Due to repetitive regions and sequencing errors, a read may have multiple mapping positions. BOSS only keeps the read mapping number of paired reads that link \( ci \) and \( cj \), \( lr(pr) \) is forwardly mapped to \( ci \) and \( rr(pr) \) is reversely mapped to \( cj \).

Fig. 1. For one paired read \( pr \) and two contigs \( ci \) and \( cj \), \( lr(pr) \) is forwardly mapped to \( ci \) and \( rr(pr) \) is reversely mapped to \( cj \).

Downloaded from https://academic.oup.com/bioinformatics/article-abstract/33/2/169/2525706
by guest
on 03 March 2018
position with the highest score and discards the rest. Because the insert size approximately follows a normal distribution $N(\mu_r, \sigma_r)$, the probability that the insert size is beyond the range $[\mu_r - 3 \times \sigma_r, \mu_r + 3 \times \sigma_r]$ is less than 5%. For a paired read mapped to the same contig, if its insert size is beyond $[\mu_r - 3 \times \sigma_r, \mu_r + 3 \times \sigma_r]$, it is considered as abnormal and its mapping information will be removed.

For a paired read $pr$ that links two contigs $ci$ and $cj$, if the sum of $crd(r(pr), ci)$ and $crd(r(pr), cj)$ is greater than $\mu_r + 3 \times \sigma_r$, its mapping information will also be removed. Sequencing errors commonly cause that some reads cannot be mapped to any position. BOSS calculates the probability, $pr$, that a read can be mapped to a contig, which is the ratio of the number of mapped reads left to the total number of reads. Next, BOSS calculates read coverage for each position of contigs based on read mapping information, then BOSS computes, $\sigma_r$, the standard deviation of the read coverage and the average of read coverage. BOSS removes paired reads whose one mate read is mapped to the region whose read coverage is $2 \times \sigma_r$ larger than the average.

### 3.2 Constructing a scaffold graph

In this step, BOSS constructs a scaffold graph $G$ with the vertex set $V$ and the edge set $E$. A vertex $vi$ represents a contig $ci$. An edge $ej$ is represented by a six-tuple $(vi, vj, o_i, o_j, gd_{ij}, w_{ij})$, where $vi$ and $vj$ are two vertices, $o_i$ and $o_j$ are mapping orientations of paired reads to $vi$ and $vj$, respectively, $gd_{ij}$ is the gap distance between $vi$ and $vj$ and $w_{ij}$ is the weight of the edge.

#### 3.2.1 Adding edges between vertices

If there exists paired reads that link two vertices $vi$ and $vj$ ($ci$ and $cj$), BOSS uses the statistical method to decide whether an edge should be added between them and how the edge should be weighted. The statistical method includes five steps.

1. **Determining the mapping orientations between $vi$ and $vj$.** If the paired reads that link $vi$ and $vj$ have two or more different mapping orientation pairs ($o_i, o_j$), BOSS only keeps the mapping orientation pair which is supported by the larger number of paired reads and discards paired reads with other mapping orientation pairs. After that, if the number of paired reads that link them is smaller than a threshold $num_{min}$ (default 2), a parameter in BOSS, an edge will not be added between them and the statistical method is terminated.

2. **Computing the gap distance between $vi$ and $vj$.** For the $r$th paired read $pr$ that links $vi$ and $vj$, BOSS calculates $crd(r(pr), vi)$ and $crd(r(pr), vj)$ (an example in Fig. 1). Based on all paired reads that link them, BOSS first calculates $gd_{ij}$ by the following formula:

$$gd_{ij} = \frac{1}{n} \sum_{t=1}^{n} (\mu_r - crd(r(pr_t), vi) - crd(r(pr_t), vj))$$  \hspace{1cm} (1)

where $n$ is the number of paired reads that link $vi$ and $vj$, $pr_t$ is the $t$-th paired read.

3. **Computing the expectation number of paired reads between $vi$ and $vj$ based on reads mapped to $v$.** Given a read $r$ which has been mapped to $vi$ ($ci$) and its mapping orientation is the same as $o_i$, if the mate read of $r$ can be mapped to $vj$, $l_{min}$ and $l_{max}$ are the smallest and largest distance between them. $l_{min}$ is the sum of $crd(r, vj)$, $gd_{ij}$ and $len(r)$. $len(r)$ is the length of $r$. $l_{max}$ is the sum of $crd(r, vi)$, $gd_{ij}$ and $len(r)$. $l_{min}$ is the length of $v_i$, $l_{max}$ is the largest insert size.

### 5.1. Computing the expectation number of paired reads between $vi$ and $vj$ based on reads mapped to $v$. BOSS can also get another expectation number $exp_p$ based on reads mapped to $vj$ in the same way described in the previous step, and get $\rho_{ij}$.

5. **Computing the edge weight.** If the arithmetic mean of $\rho_{ij}$ and $\rho_{ji}$ are greater than a threshold $w_{min}$ (default 0.2), BOSS adds an edge $e_{ij}$ between $vi$ and $vj$. If $o_i$ ($o_j$) is a forward mapping, the edge $e_{ij}$ connects 5’-end of $vi$ ($vj$), else 3’-end. $w_{ij}$ is equal to the arithmetic mean of $\rho_{ij}$ and $\rho_{ji}$. If $\rho_{ij}$ or $\rho_{ji}$ is smaller than $w_{min}$, but $\rho_{ij}$ has no paired reads that link other vertices with the same $o_i$, no matter whether their mate reads are mapped to $vi$ or not. Then, BOSS defines:

$$\rho_{ij} = \min \left( \frac{\exp_p}{n}, \frac{\exp_p}{\exp_p} \right)$$  \hspace{1cm} (4)

where $\rho_{ij}$ is to measure how close the expectation number of paired reads is to the real number.

5. **Computing the edge weight.** If the arithmetic mean of $\rho_{ij}$ and $\rho_{ji}$ are greater than a threshold $w_{min}$ (default 0.2), BOSS adds an edge $e_{ij}$ between $vi$ and $vj$. If $o_i$ ($o_j$) is a forward mapping, the edge $e_{ij}$ connects 5’-end of $vi$ ($vj$), else 3’-end. $w_{ij}$ is equal to the arithmetic mean of $\rho_{ij}$ and $\rho_{ji}$. If $\rho_{ij}$ or $\rho_{ji}$ is smaller than $w_{min}$, but $\rho_{ij}$ has no paired reads that link other vertices with the same $o_i$, no matter whether their mate reads are mapped to $vi$ or not. Then, BOSS defines:

$$\rho_{ij} = \min \left( \frac{\exp_p}{n}, \frac{\exp_p}{\exp_p} \right)$$  \hspace{1cm} (4)

where $\rho_{ij}$ is to measure how close the expectation number of paired reads is to the real number.

5. **Computing the edge weight.** If the arithmetic mean of $\rho_{ij}$ and $\rho_{ji}$ are greater than a threshold $w_{min}$ (default 0.2), BOSS adds an edge $e_{ij}$ between $vi$ and $vj$. If $o_i$ ($o_j$) is a forward mapping, the edge $e_{ij}$ connects 5’-end of $vi$ ($vj$), else 3’-end. $w_{ij}$ is equal to the arithmetic mean of $\rho_{ij}$ and $\rho_{ji}$. If $\rho_{ij}$ or $\rho_{ji}$ is smaller than $w_{min}$, but $\rho_{ij}$ has no paired reads that link other vertices with the same $o_i$, no matter whether their mate reads are mapped to $vi$ or not. Then, BOSS defines:

$$\rho_{ij} = \min \left( \frac{\exp_p}{n}, \frac{\exp_p}{\exp_p} \right)$$  \hspace{1cm} (4)

where $\rho_{ij}$ is to measure how close the expectation number of paired reads is to the real number.

5. **Computing the edge weight.** If the arithmetic mean of $\rho_{ij}$ and $\rho_{ji}$ are greater than a threshold $w_{min}$ (default 0.2), BOSS adds an edge $e_{ij}$ between $vi$ and $vj$. If $o_i$ ($o_j$) is a forward mapping, the edge $e_{ij}$ connects 5’-end of $vi$ ($vj$), else 3’-end. $w_{ij}$ is equal to the arithmetic mean of $\rho_{ij}$ and $\rho_{ji}$. If $\rho_{ij}$ or $\rho_{ji}$ is smaller than $w_{min}$, but $\rho_{ij}$ has no paired reads that link other vertices with the same $o_i$, no matter whether their mate reads are mapped to $vi$ or not. Then, BOSS defines:

$$\rho_{ij} = \min \left( \frac{\exp_p}{n}, \frac{\exp_p}{\exp_p} \right)$$  \hspace{1cm} (4)

where $\rho_{ij}$ is to measure how close the expectation number of paired reads is to the real number.
In this step, BOSS tries to find the optimal assignment of orientations that maximize the sum weight of non-spurious edges by an integer linear programming model \( LP(OB,CS) \) (see Algorithm 1). \( OB \) is the objective function and \( CS \) is the set of constraints. \( s_i \in \{0,1\} \) denotes orientation of \( v_i \). \( \eta_i \in \{0,1\} \) represents whether \( e_{ij} \) is spurious or not. BOSS iterates \( w \) from \( w_{\max} \) to \( w_{\min} \) with a step of 0.1. \( w \) is an edge weight threshold used to construct a sub-graph based on the scaffold graph. In the first iteration, \( w = w_{\max} \), BOSS constructs a sub-graph \( G_s \), which consists of edges whose weights are in \([w,1)\). In the next iteration, \( w \) is set to \( w - 0.1 \). BOSS adds edges whose weights are in \([w,w + 0.1) \) to \( G_s \). In each iteration, BOSS constructs \( LP(OB,CS) \) and solves it. After getting the orientation assignment, BOSS removes spurious edges, while the remaining edges are non-spurious and are used as prior information to guide detection of spurious edges in the subsequent iterations. The iteration will be terminated after \( w \) is smaller than \( w_{\min} \).

In each iteration, the orientation of each vertex in \( G_s \) is a variable. \( TE \) is a set of edges which are added to \( G_s \) in this iteration. For an edge \( e_{ij} \in G_s \), if \( e_{ij} \notin TE \), and \( o_i \neq o_j \), BOSS adds the following constraint equation (5) to \( CS \) in which \( \eta_{ij} = 1 \). If \( e_{ij} \notin TE \), and \( o_i = o_j \), BOSS adds the following constraint equation (6) to \( CS \) in which \( \eta_{ij} = 1 \). If \( e_{ij} \in TE \), and \( o_i \neq o_j \), BOSS adds the following constraint equation (5) to \( CS \) in which \( \eta_{ij} \) is a variable.

\[
\eta_{ij} \leq s_i + s_j \leq 2 - \eta_{ij} \tag{5}
\]

\[
\eta_{ij} - 1 \leq s_i - s_j \leq 1 - \eta_{ij} \tag{6}
\]

Then BOSS maximizes the sum weight of non-spurious edges, and the objective function is:

\[
OB = \max \left( \sum_{e_{ij} \in TE} (w_{ij} \star \eta_{ij}) \right) \tag{7}
\]

BOSS uses the branch-and-bound algorithm to solve \( LP(OB,CS) \) and gets the approximate optimal assignment. If \( \eta_{ij} = 0 \), \( e_{ij} \) is spurious and BOSS removes \( e_{ij} \) from \( G_s \). If \( \eta_{ij} = 1 \), \( e_{ij} \) is non-spurious. After all iterations, the scaffold graph \( G \) is replaced with \( G_s \). Finally, BOSS reverses and complements some vertices to make sure that two ends of each edge correspond to 3’-end of one vertex and 5’-end of another vertex, respectively.

(2) Detecting spurious edges by assigning the coordinates of vertices.

In this step, BOSS assigns a starting coordinate for each vertex and tries to find the best assignment such that the gap distances between vertices calculated by starting coordinates agree the best with the gap distances suggested by edges. BOSS still uses the iterative strategy to construct sub-graph \( G_s \) as the above process. In each iteration, BOSS constructs a new \( LP(OB,CS) \) based on \( G_s \). \( TE \) is a set of edges which are added to \( G_s \) in this iteration. For \( G_s \), if edge \( e_{ij} \in TE \) which connects 3’-end of \( v_i \) and 5’-end of \( v_j \), BOSS adds the following constraint to \( CS \) (Donmez and Brudno, 2013):

\[
L(\phi_{ij} - 1) < x_i - x_j - len(e_j) - gd_i < L(1 - \phi_{ij}) \tag{8}
\]

\( x_i \in [0, L] \) is an integer and denotes the starting coordinate of vertex \( v_i \). \( L \) is a large constant and equal to twice the sum of the length of all contigs. \( \phi_{ij} \) is a slack variable in the range \([0,1]\) which reflects the consistency between two gap distances suggested by starting coordinates \( x_i \) and \( x_j \) and edge \( e_{ij} \). If edge \( e_{ij} \notin TE \), \( e_{ij} \) is non-spurious, and BOSS adds the following constraint to \( CS \), which is used as prior information to identify spurious edges from \( TE \).

\[
0 < x_j - x_i - len(e_i) < \mu_i + 3 \star \sigma_i \tag{9}
\]

The objective function is:

\[
OB = \max \left( \sum_{e_{ij} \in ES} (w_{ij} \star \phi_{ij}) \right) \tag{10}
\]

After BOSS gets the approximate optimal assignment, for an edge in \( TE \), if the gap distance suggested by starting coordinates is far away from the one suggested by the edge, this edge is spurious and BOSS removes it. After all iterations, the scaffold graph \( G \) is

**Algorithm 1. Detect_orientation(G)**

1: Initialization \( G_s = \emptyset \); \( TE = \emptyset \); \( w = w_{\max} \); \( CS = \emptyset \)
2: while \( w \geq w_{\min} \) do
3:   for each edge \( e_{ij} \in G_s \) do
4:     if \( w < w_{ij} < w + 0.1 \) then
5:       add \( e_{ij} \) to \( G_s \) and \( TE \)
6:     end if
7:   end for
8:   for each edge \( e_{ij} \in G_s \) do
9:     if \( e_{ij} \notin TE \) then
10:       if \( o_i \neq o_j \) then
11:         add “\( s_i + s_j = 1 \)” to \( CS \)
12:       else
13:         add “\( s_i = s_j \)” to \( CS \)
14:       end if
15:     end if
16:   end for
17:   \( OB = \max(\sum_{e_{ij} \in CS} (w_{ij} \star \eta_{ij})) \) and solve \( LP(OB, CS) \)
18:   for each edge \( e_{ij} \in TE \) do
19:     if \( \eta_{ij} = 0 \) then
20:       delete \( e_{ij} \) from \( G_s \)
21:     end if
22:   end for
23:   \( TE = \emptyset \); \( CC = \emptyset \) and \( w = w - 0.1 \)
24: end while
25: replace \( G \) with \( G_s \) and output \( G \)
replaced with $G_s$, and BOSS gets the starting coordinate of each vertex. If two vertices occupy the same coordinates, BOSS traverses the scaffold graph from the two vertices separately to find the first vertex they meet, and removes the edge with the smaller weight entering the first vertex.

3.3 Sorting vertices in scaffold graph

In this step, BOSS produces scaffolds by sorting vertices in the scaffold graph $G$. BOSS first extracts simple paths only including long vertices whose lengths are equal to or larger than $\mu_0 + 3 \times \sigma_0$, and these simple paths compose a path set $PS$. Second, BOSS identifies short vertices (whose lengths are smaller than $\mu_0 + 3 \times \sigma_0$) which have edges connecting with two adjacent long vertices in the same path, then BOSS inserts these short vertices into the middle of the two adjacent long vertices. Finally, for a path, BOSS extends it through short vertices which have not been inserted to the paths in $PS$. In the extending process, BOSS adopts the breadth-first search to sort the short vertices based on the scaffold graph, gap distances and contig lengths, these sorted vertices are appended to the paths. When the extension meets one end of another path, these two paths are merged. The process is terminated if there are no vertices that have not been visited. An illustrative example is shown in Figure 3.

4 Experiment

4.1 Datasets

To evaluate the performance of BOSS, experiments are carried out on four real datasets used in Hunt et al. (2014). These datasets include Illumina reads from the Staphylococcus aureus $(S. aureus)$, Rhodobacter sphaeroides $(R. sphaeroides)$, Human chromosome 14 and the Plasmodium falciparum $(P. falciparum)$ clone 3D7 reference genome. Details about four datasets are listed in Table 1. The mapping read ratio is calculated based on Bowtie2 (Langmead and Salzberg, 2012). The first two datasets include only one read library and the last two datasets contain two read libraries with different properties. Contig sets are generated by genome assembler Velvet (Zerbino and Birney, 2008).

4.2 Evaluation metrics

Although some users might think that good scaffolds should have a large N50 and CN50 (Salzberg et al., 2012), these metrics do not necessarily reflect correctly orientated or ordered contigs in a scaffold (Hunt et al., 2014; Mandric and Zelikovsky, 2015). For assessing the quality of scaffolds reliably, a novel evaluation tool is proposed by Hunt et al. (2014) in which each contig is represented by a sequence tag and four key metrics are presented: (1) Correct Joins (CJ). (2) Incorrect Joins (IJ). (3) Skipped Tags (ST). (4) Lost Tags (LT). The last three metrics are bad joins. We set the weights of four metrics of CJ, IJ, LT and ST as 1, 1, 2 and 0.5, respectively, according to the widely accepted rule (Hunt et al., 2014). The number of potential joins is the number of contigs that can be joined to the scaffold which is the number of contigs minus the number of chromosomes. After we get the correct joins (CJ) and bad joins (IJ, ST and LT) of scaffolds, we adopt the $F$-score as a comprehensive metric which is calculated by $TPR$ and $PPV$ (Mandric and Zelikovsky, 2015).

4.3 Statistical method and iterative strategy

For verifying the effectiveness of the statistical method and the iterative strategy presented in this paper, we compare other two different versions of BOSS: BOSS1 and BOSS2. The difference among BOSS1, BOSS2 and BOSS is that they adopt different strategies for constructing a scaffold graph. When constructing a scaffold graph, BOSS1 adds an edge between two vertices if the number of paired reads that links them is larger than $num_{min}$. BOSS2 uses the statistical method (described in section 3.2.1) for constructing a scaffold graph. BOSS not only adopts the statistical method but also the iterative strategy to construct a scaffold graph. The scaffolding results about $S. aureus$ and $P. falciparum$ are shown in Table 2. The scaffolding results about $R. sphaeroides$ and Human chromosome 14 are shown in Table 3.

From Tables 2 and 3, we can find that the $F$-score of BOSS is larger than that of BOSS1, and the $F$-score of BOSS is larger than that of BOSS2 for dataset of $S. aureus$, dataset of $R. sphaeroides$, and dataset of Human chromosome 14.
short insert size dataset of *P.falciparum*, short and long insert size datasets of *P.falciparum*, short insert size dataset of Human chromosome 14, and short and long insert size datasets of Human chromosome 14. For the long insert size dataset of *P.falciparum*, although the F-score of BOSS2 is larger than that of BOSS1, the F-score of BOSS is smaller than that of BOSS2. For the long insert size dataset of Human chromosome 14, the F-score of BOSS1 is the largest and the F-score of BOSS is smaller than that of BOSS2. In conclusion, the statistical method and the iterative strategy are effective for most cases. Below we discuss the case that BOSS produces more satisfactory scaffolding results.

### 4.4 Comparisons of scaffolders and discussion

We compare BOSS with popular scaffolders Bambus (Koren et al., 2011), MIP (Salmela et al., 2011), OPERA (Gao et al., 2011), SCARPA (Donmez and Brudno, 2013), SOPRA (Dayarian et al., 2010) and SSPACE (Boetzer et al., 2011), and scaffolding modules from assemblers ABYSS (Simpson et al., 2009), SOAP2 (Luo et al., 2012) and SGA (Simpson and Durbin, 2012). We compute the F-score of these scaffolders based on scaffolding results which are abstracted from Hunt et al. (2014) whose datasets are used in this paper. Moreover, two recently published scaffolders BEST (Sahlin et al., 2014) and ScaffMatch (Mandric and Zelikovsky, 2015) are also considered, and we compute the F-score of these two scaffolders based on Mandric and Zelikovsky (2015) who adopt the same datasets.

Some scaffolders contain the mapping process by one or more mapping tools in their own pipeline, others need SAM or BAM files produced by users themselves. Read mapping information produced by different mapping tools usually leads to distinct scaffolding results. For conducting unbiased investigation, three mapping tools: Bowtie (Langmead and Saltzberg, 2012) and BWA (Li and Durbin, 2009) are used. Bowtie maps any read which has no mismatches (when using the option -v 0) or up to three mismatches (when using -v 3). We only show the evaluation results of scaffolders using Bowtie2 except scaffolders containing their own mapping tools. All evaluation results, and the running time and peak memory of BOSS are provided in Supplementary Materials.

For *S.aureus*, there are 170 contigs and 167 potential joins, and the scaffolding results are shown in Table 4. For *R.sphaeroides*, there are 577 contigs and 570 potential joins, and the scaffolding results are shown in Table 4.

---

### Table 2. Performances of BOSS1, BOSS2 and BOSS on *S.aureus* and *P.falciparum*

<table>
<thead>
<tr>
<th>Dataset</th>
<th>TPR</th>
<th>PPV</th>
<th>F-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOSS1</td>
<td>0.725</td>
<td>0.569</td>
<td>0.638</td>
</tr>
<tr>
<td>BOSS2</td>
<td>0.886</td>
<td>0.851</td>
<td>0.868</td>
</tr>
<tr>
<td>BOSS</td>
<td>0.862</td>
<td>0.878</td>
<td>0.870</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dataset</th>
<th>TPR</th>
<th>PPV</th>
<th>F-score</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>P.falciparum</em></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BOSS1</td>
<td>0.585</td>
<td>0.737</td>
<td>0.652</td>
</tr>
<tr>
<td>BOSS2</td>
<td>0.634</td>
<td>0.896</td>
<td>0.742</td>
</tr>
<tr>
<td>BOSS</td>
<td>0.642</td>
<td>0.921</td>
<td>0.757</td>
</tr>
</tbody>
</table>

### Table 3. Performances of BOSS1, BOSS2 and BOSS on R. sphaeroides and Human chromosome 14

<table>
<thead>
<tr>
<th>Dataset</th>
<th>TPR</th>
<th>PPV</th>
<th>F-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOSS1</td>
<td>0.751</td>
<td>0.638</td>
<td>0.690</td>
</tr>
<tr>
<td>BOSS2</td>
<td>0.863</td>
<td>0.801</td>
<td>0.831</td>
</tr>
<tr>
<td>BOSS</td>
<td>0.870</td>
<td>0.881</td>
<td>0.876</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dataset</th>
<th>TPR</th>
<th>PPV</th>
<th>F-score</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>R.sphaeroides</em></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BOSS1</td>
<td>0.661</td>
<td>0.528</td>
<td>0.587</td>
</tr>
<tr>
<td>BOSS2</td>
<td>0.789</td>
<td>0.692</td>
<td>0.738</td>
</tr>
<tr>
<td>BOSS</td>
<td>0.766</td>
<td>0.783</td>
<td>0.775</td>
</tr>
</tbody>
</table>

### Table 4. Evaluation results about *S.aureus* and *R.sphaeroides*

<table>
<thead>
<tr>
<th>Dataset</th>
<th>TPR</th>
<th>PPV</th>
<th>F-score</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>S.aureus</em></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ABYSS</td>
<td>619764</td>
<td>619764</td>
<td>0.593</td>
</tr>
<tr>
<td>Bambus2</td>
<td>242814</td>
<td>242650</td>
<td>0.569</td>
</tr>
<tr>
<td>MIP</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>OPERA</td>
<td>1084108</td>
<td>686577</td>
<td>0.671</td>
</tr>
<tr>
<td>SCARPA</td>
<td>112264</td>
<td>112083</td>
<td>0.461</td>
</tr>
<tr>
<td>SGA</td>
<td>309286</td>
<td>309153</td>
<td>0.497</td>
</tr>
<tr>
<td>SOAP2</td>
<td>643384</td>
<td>621109</td>
<td>0.784</td>
</tr>
<tr>
<td>SOPRA</td>
<td>112278</td>
<td>112083</td>
<td>0.240</td>
</tr>
<tr>
<td>SSPACE</td>
<td>323784</td>
<td>261710</td>
<td>0.629</td>
</tr>
<tr>
<td>BESTT</td>
<td>1716351</td>
<td>335064</td>
<td>0.671</td>
</tr>
<tr>
<td>ScafIMatch</td>
<td>1476925</td>
<td>351546</td>
<td>0.832</td>
</tr>
<tr>
<td>BOSS</td>
<td>1035497</td>
<td>396371</td>
<td>0.862</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dataset</th>
<th>TPR</th>
<th>PPV</th>
<th>F-score</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>R.sphaeroides</em></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ABYSS</td>
<td>280984</td>
<td>276804</td>
<td>0.674</td>
</tr>
<tr>
<td>Bambus2</td>
<td>146002</td>
<td>145952</td>
<td>0.577</td>
</tr>
<tr>
<td>MIP</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>OPERA</td>
<td>108182</td>
<td>108172</td>
<td>0.554</td>
</tr>
<tr>
<td>SCARPA</td>
<td>37667</td>
<td>37581</td>
<td>0.367</td>
</tr>
<tr>
<td>SGA</td>
<td>488095</td>
<td>487941</td>
<td>0.735</td>
</tr>
<tr>
<td>SOAP2</td>
<td>42825</td>
<td>42722</td>
<td>0.407</td>
</tr>
<tr>
<td>SOPRA</td>
<td>32232</td>
<td>30492</td>
<td>0.425</td>
</tr>
<tr>
<td>SSPACE</td>
<td>2522483</td>
<td>2522482</td>
<td>0.821</td>
</tr>
<tr>
<td>BESTT</td>
<td>1021151</td>
<td>1020921</td>
<td>0.846</td>
</tr>
<tr>
<td>ScafIMatch</td>
<td>2547006</td>
<td>2528248</td>
<td>0.644</td>
</tr>
<tr>
<td>BOSS</td>
<td>2548917</td>
<td>2546780</td>
<td>0.870</td>
</tr>
</tbody>
</table>
results are also shown in Table 4. For *P. falciparum*, there are 9318 contigs and 9302 potential joins. All scaffolders use two datasets for scaffolding separately, and the scaffolding results are shown in Table 5. For Human chromosome 14, there are 19936 contigs and 19935 potential joins, and the scaffolding results using two datasets separately are shown in Table 6. For *P. falciparum* and Human chromosome 14, the scaffolding results using the combination of the short and long insert size datasets are shown in Table 7. ‘–’ in Tables 4, 6 and 7 means that the results cannot be got by corresponding scaffolders.

In Table 4, 5 and 6, the bold values represent the best values of CN50 or F-score. For eight datasets, BOSS has the best F-score for three datasets and the best CN50 for three datasets. SOAP2 has the best F-score for four datasets and the best CN50 for two datasets. OPERA has the best CN50 for two datasets. ScaffMatch has the best F-score for one dataset. MIP has the best CN50 for one dataset.

### 4.5 Discussion

By investigating and analyzing the scaffolding results of BOSS, we find that BOSS performs better on datasets whose ratio of the...
number of mapping reads to the genome size is large. The ratio is symbolized as $\omega$. We can calculate $\omega$ based on the read number, genome size and mapping read ratio which are shown in Table 1. For datasets of S.aureus and the short insert size dataset of P.falciparum, their values of $\omega$ are 0.661 and 1.753, respectively. BOSS gets the best F-score compared with all other scaffolders. We conjecture that BOSS produces more satisfactory scaffolding result when the value of $\omega$ for datasets is large.

For the dataset of R.sphaeroides, long insert size dataset of Plasmodium falciparum, short insert size dataset of Human chromosome 14, and long insert size dataset of Human chromosome 14, their values of $\omega$ are 0.316, 0.219, 0.188 and 0.024, and BOSS also can produce acceptable scaffolding results. The reason why the value of $\omega$ influences BOSS is that the paired reads between two actual adjacent contigs usually is large when the value of $\omega$ is large, and this is helpful in more precisely weighting edges based on the statistical method proposed in this paper.

In BOSS, the insert size and its standard deviation of the dataset are parameters provided by users. For further examining the impact of standard deviation on the scaffolding results, we conduct BOSS with different standard deviations based on the mapping tool Bowtie2. The scaffolding results are shown in Supplementary Material. From the results, we can see that different standard deviations influence the scaffolding results, but the range of F-score variation is not large. BOSS adopts $\sigma_{\omega} = 0.07 * \mu_{\omega}$ in default.

5 Conclusion

In this paper, we have presented a novel scaffold BOSS for determining orientations and orders of contigs. BOSS employs a new statistical method to decide whether an edge between contigs should be added and how the edge should be weighed. In addition, BOSS adopts an iterative strategy to detect and remove spurious edges in the scaffold graph. Finally, BOSS sorts vertices in the scaffold graph, thus producing oriented and ordered vertices which correspond to scaffolds. The experiments have been conducted on four datasets. The results have illustrated that BOSS outperforms other competing scaffolders when the value of $\omega$ for datasets is large, and also can produce comparable scaffolding results in the other cases.

Funding

This work was supported in part by the National Natural Science Foundation of China under Grant No. 61232001, No. 61420106009, No. 61379108, No.61602156 and The National Science Fund for Excellent Young Scholars under Grant No. 61622213.

Conflict of Interest: none declared.

References

Lindsay,J. et al. (2014) Ilp-based maximum likelihood genome scaffolding. BMC Bioinformatics, 15, 59.