JOURNAL OF COMPUTATIONAL BIOLOGY
Volume 19, Number 10, 2012

Pp. 1162–1175
DOI: 10.1089/cmb.2011.0263

SLIQ: Simple Linear Inequalities for Efficient Contig Scaffolding

RAJAT S. ROY,[1] KEVIN C. CHEN,[2] ANIRVAN M. SENGUPTA,[3] and ALEXANDER SCHLIEP[4]

ABSTRACT

Scaffolding is an important subproblem in *de novo* genome assembly, in which mate pair data are used to construct a linear sequence of contigs separated by gaps. Here we present SLIQ, a set of simple linear inequalities derived from the geometry of contigs on the line that can be used to predict the relative positions and orientations of contigs from individual mate pair reads and thus produce a contig digraph. The SLIQ inequalities can also filter out unreliable mate pairs and can be used as a preprocessing step for any scaffolding algorithm. We tested the SLIQ inequalities on five real data sets ranging in complexity from simple bacterial genomes to complex mammalian genomes and compared the results to the majority voting procedure used by many other scaffolding algorithms. SLIQ predicted the relative positions and orientations of the contigs with high accuracy in all cases and gave more accurate position predictions than majority voting for complex genomes, in particular the human genome. Finally, we present a simple scaffolding algorithm that produces linear scaffolds given a contig digraph. We show that our algorithm is very efficient compared to other scaffolding algorithms while maintaining high accuracy in predicting both contig positions and orientations for real data sets.

Key words: algorithms, biochemical networks, combinatorics, computational molecular biology, evolution, gene expressions, gene networks, genetic variation, graphs and networks, graph theory, next generation sequencing.

1. INTRODUCTION

DE NOVO GENOME ASSEMBLY is a classical problem in bioinformatics, in which short DNA sequence reads are assembled into longer blocks of contiguous sequence (contigs), which are then arranged into linear chains of contigs separated by gaps (scaffolds). Previously, only single-end short reads were available from sequencing experiments. Modern genome sequencing technology allows reporting reads in pairs, commonly known as mate pairs or paired end. The distance between the two reads of a pair plus the two read lengths (the insert length) approximately follows a normal distribution determined during the experimental construction

[1]Department of Computer Science, Rutgers The State University of New Jersey, Piscataway, NJ.
[2]Department of Genetics, [3]Department of Physics and Astronomy, [4]Department of Computer Science, BioMaPS Institute for Quantitative Biology, Rutgers The State University of New Jersey, Piscataway, NJ.

of the library. Some genome projects also include mate-pair libraries with several different insert lengths. Although there are experimental differences between mate pairs and paired-end reads, we will refer to them interchangeably as mate pairs since we can treat them identically from an algorithmic point of view. Mate pairs are particularly important for *de novo* assembly since, in addition to building contigs, we can now hypothesize about neighbors of a contig whenever the reads of a pair fall on different contigs. This opens the possibility of scaffolding contigs.

Computational genome assembly is typically performed in at least two stages—the contig building stage and the scaffolding stage. In this article we do not address the contig building problem but rather assume that we have access to a set of contigs produced by an independent algorithm. However, we discuss the relationship of the contig building and scaffolding stages later in the discussion. The scaffolding problem tries to string contigs into a chain such that the order of the contigs in the scaffold reflects their real order in the genome. For the scaffolding problem, the most popular strategy is to construct the contig graph in which nodes represent contigs and edges represent sets of mate pairs connecting two contigs (i.e., the two reads of the mate pair fall in the two different contigs). The edges are given weights equal to the number of mate pairs connecting the two contigs.

We then try to find a walk in the graph such that the minimum number of mate pairs are violated. A mate pair is violated when the contigs it is connecting do not have the relative orientation or position suggested by the mate pair. Just finding the optimal orientation assignment is reducible to the Maximum Cut problem, which is known to be NP-complete (Garey, 1979). Consequently, finding the optimal walk to get the optimal scaffolding is also NP-complete. The genome can (and often does) have repeated regions; e.g., approximately 50% of human genome is accounted for by repeats (Haubold and Wiehe, 2006). But the contig builder is likely to report one contig per repeated region. This repetitive structure of the genome makes scaffolding harder as it introduces loops and cycles in the contig graph. We also have false edges resulting from misassembly of reads into contigs. Unfortunately, the number of false edges is not negligible, and so, filtering them is a major preprocessing step.

A common procedure is to filter out unreliable edges by picking a small threshold (commonly 2–5) and removing all edges with weight less than that threshold. For the remaining edges, a majority vote is used to decide on the relative orientation and position of the contigs. This simple majority voting strategy is implemented in a number of commonly used assemblers and stand-alone scaffolders, including ARACHNE (Batzoglou et al., 2002), BAMBUS (Pop et al., 2004), SOPRA (Dayarian et al., 2010), and SOAPdenovo (Li et al., 2010), with various choices of threshold. Opera (Gao et al., 2011) and the Greedy Path-Merging algorithm (Huson et al., 2002) use a different strategy to bundle edges. Given a set of mate pairs connecting two contigs, these algorithms calculate the median and standard deviation of the insert lengths of the set of mate pairs and create a bundle using only mate pairs with insert length that are close to the median. ALLPATHS (Butler et al., 2008) and VELVET (Zerbino and Birney, 2008) do not build the contig graph and thus do not have a read-filtering step similar to the other assemblers mentioned. The majority voting procedure implicitly assumes that misleading mate pairs are random and independently generated and that majority voting should eliminate the problematic mate pairs. However, this assumption is often not true because of the complex repeat structure of large genomes, such as human.

In this article, we show that unreliable mate pairs can be reliably filtered using SLIQ, a set of simple linear inequalities derived from the geometry of contigs on the line. Thus, SLIQ produces a reduced subset of reliable mate pairs and thus a sparser graph, which results in a simpler optimization problem for the scaffolding algorithm. More importantly, SLIQ can be used to predict the relative positions and orientations of the contigs, yielding a *directed* contig graph. Our experiments show that both SLIQ and majority voting are very accurate at predicting relative orientations, but SLIQ is clearly more accurate at predicting relative positions for complex genomes.

The simplicity of SLIQ makes it very easy to integrate as a preprocessing step to any existing scaffolders, including recent scaffolders such as MIP scaffolder (Salmela et al., 2011), Bambus 2 (Koren et al., 2011), and SSPACE (Boetzer et al., 2011). To illustrate the effectiveness of SLIQ, we implemented a naive scaffolding algorithm that produces linear scaffolds from the contig digraph. We show that despite its simplicity, our naive scaffolder provides very accurate draft scaffolds, comparable to or improving upon the more complicated state of the art, very quickly. These scaffolds can either be output directly or used as reasonable starting points for further refinement with more complex scaffolding algorithms. An implementation of naive assembler using the inequalities is available online.

2. ALGORITHMS

We begin with a high level outline of our algorithm for constructing a directed contig graph (Algorithm 1). The crux of the algorithm is SLIQ, a set of simple linear inequalities that are used to filter mate pairs and predict the relative position and orientation of contigs. In subsequent sections, we will present proofs for the SLIQ inequalities and a detailed version of the digraph construction algorithm (Algorithm 2). Finally, we will present a simple scaffolding algorithm (Algorithm 3) that uses the contig digraph to construct draft scaffolds. Throughout the article, we will abbreviate mate-pair reads as *MPR*.

Algorithm 1. Construct Contig Digraph (Outline)

Require: *input:* P = a set of MPRs that connect two contigs, C = a set of contigs

1: Construct the contig graph G with vertex set C and edges representing MPRs from P that pass a certain majority cutoff.
2: Find a good orientation assignment for the contigs ($\Theta = \{\Theta_1, \Theta_2, \ldots\}$) where Θ_i is the orientation of the ith contig, for example, by finding a spanning tree of G.
3: Define M_p to be the set of MPRs that satisfies the SLIQ inequalities
4: Construct a directed contig graph G_d with vertex set C and edges representing MPRs from M_p that pass certain criteria.

2.1. Definitions and assumptions

For the sake of deriving the SLIQ inequalities, we assume that we know the position of the contigs on the reference genome. However, this information cancels out later on, which allows us to analyze the MPRs without access to prior contig position information. For the derivation, we also assume that all the contigs have the same orientation. Later, we will not need this information.

Let P_i be the position of contig C_i in the genome, and l_i be the length of the contig (Fig. 1). We define gap g_{ij} to be the difference between the start position of contig C_j, and the end position of contig C_i, and similarly for g_{ji}:

$$\begin{aligned} g_{ij} &= P_j - P_i - l_i, \\ g_{ij} &= P_i - P_j - l_j. \end{aligned} \tag{1}$$

We assume that the maximum overlap of two contigs is one read length, R. In practical contig-building software based on De Bruijn graphs, the maximum overlap is usually one k-mer where $R > k$, so our assumption is valid.

2.2. Derivation of two gap equations

If we assume that $P_i < P_j$ as in Fig. 1, and that the maximum overlap between two contigs is R (i.e., the minimum gap g_{ij} is $-R$), then

$$\begin{aligned} P_j - P_i - l_i &\geq -R, \\ P_j - P_i &\geq l_i - R. \end{aligned} \tag{2}$$

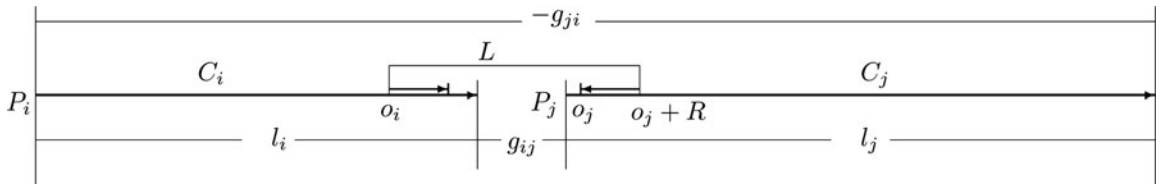

FIG. 1. The geometry of two contigs, C_i and C_j, arranged on a line with relevant quantities indicated. Here, L is the insert length, P_i is the start position of contig C_i, l_i is the length of the contig C_i, o_i is the offset of the read of the mate-pair read (MPR) that falls on C_i, R is the read length. $g_{ij} = P_j - P_i - l_i$. The quantities for C_j are defined similarly.

Now consider the quantity $g_{ij}-g_{ji}$. Using Equation (1), we can derive the following inequality, which we call Gap Equation 1

$$\begin{aligned} g_{ij}-g_{ji} &= 2,(P_j-P_i)+(l_j-l_i) \\ &\geq 2l_i-2R+l_j-l_i \\ &\geq l_i+l_j-2R. \end{aligned} \tag{3}$$

Therefore, we have shown that $(P_i < P_j) \Rightarrow (g_{ij}-g_{ji} \geq l_i+l_j-2R)$. Next consider the quantity $g_{ij}+g_{ji}$. We can easily derive Gap Equation 2:

$$g_{ij}+g_{ji} = -(l_j+l_i). \tag{4}$$

Now, we will prove the other direction of the implication in Gap Equation 1, and show that $(g_{ij}-g_{ji} \geq l_i+l_j-2R) \Rightarrow (P_i < P_j)$. Using Gap Equation 1 and Equation (1), we get

$$\begin{aligned} g_{ij}-g_{ji} &\geq l_i+l_j-2R, \\ 2(P_j-P_i)+(l_j-l_i) &\geq l_i+l_j-2R, \\ 2(P_j-P_i) &\geq 2l_i-2R, \\ P_j-P_i &\geq l_i-R. \end{aligned} \tag{5}$$

No contig length can be less than R, the length of a read. In practice, contigs of lengths R are not very reliable. Our experiments show that such contigs almost always fail to align to the reference. We suggest scaffolders enforce a minimum contig length, which is $> R$. We make the assumption $l_i-R > 0$ and that gives us $P_j-P_i > 0$ or $P_i < P_j$. Therefore, $(g_{ij}-g_{ji} \geq l_i+l_j-2R) \Rightarrow (P_i < P_j)$ and together we have proven,

$$(g_{ij}-g_{ji} \geq l_i+l_j-2R) \Longleftrightarrow (P_i < P_j). \tag{6}$$

2.3. *Using the gap equations to predict relative positions*

Our definitions in Equation (1) used the quantities P_i and P_j, which are not available in practice in *de novo* assembly. Thus, we need to define the gaps g_{ij} and g_{ji} in terms of quantities we know, such as the insert length L and the read offsets relative to the contigs o_i and o_j. Note that the insert length for each MPR is an unknown constant, so treating it as a constant in the proof is justified. In practice, we use $L = \bar{L}+2\sigma$, where $\bar{L}$ is the reported or computed mean and σ is the standard deviation of the insert length distribution.

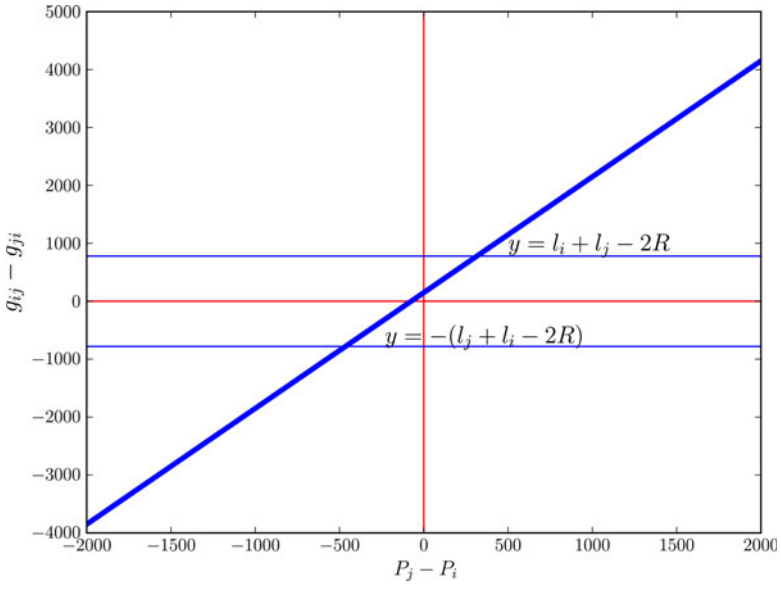

FIG. 2. Plot of Equation (5) showing the dependence of the quantity $g_{ij}-g_{ji}$ on the relative positions of the contigs.

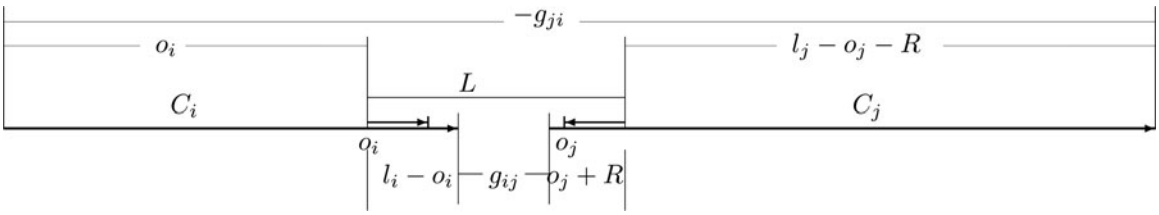

FIG. 3. The geometry of two contigs arranged on a line in terms of quantities known in *de novo* assembly.

Let L be the insert length, o_i and o_j be the offsets of the start positions of the paired reads in C_i and C_j, respectively, and Θ_i and Θ_j be the orientations of C_i and C_j, respectively. To simplify the notation we abbreviate $\Theta_i = \Theta_j$ as $\Theta_{i=j}$ and $\Theta_i \neq \Theta_j$ as $\Theta_{i\neq j}$. Then, if $P_i < P_j$ and $\Theta_{i=j}$ (Fig. 3), we can redefine the gaps g_{ij} and g_{ji} without using the contig start positions P_i and P_j:

$$\begin{aligned} g_{ij} &= L - l_i + o_i - o_j - R, \\ g_{ji} &= -L - l_j + o_j + R - o_i. \end{aligned} \tag{7}$$

Note that these definitions remain consistent with Gap Equation 2 [Equation (4)]. Taking the difference of Equations (6) and (7), we can similarly remove P_i and P_j from Gap Equation 1:

$$g_{ij} - g_{ji} = 2L - 2R + 2(o_i - o_j) + (l_j - l_i). \tag{8}$$

Using Equations (8) and (5), we derive the following inequality:

$$\begin{aligned} 2L - 2R + 2(o_i - o_j) + (l_j - l_i) &\geq l_i + l_j - 2R, \\ 2L + 2(o_i - o_j) + (l_j - l_i) &\geq l_i + l_j, \\ L + (o_i - o_j) &\geq l_i. \end{aligned}$$

Consequently, we obtain that $(P_i < P_j) \wedge \Theta_{i=j} \Rightarrow L + (o_i - o_j) \geq l_i$. Negating the implication gives

$$\begin{aligned} \neg(L + (o_i - o_j) \geq l_i) &\Rightarrow \neg((P_i < P_j) \wedge \Theta_{i=j}), \\ L + (o_i - o_j) < l_i &\Rightarrow (P_i > P_j) \vee \Theta_{i\neq j}. \end{aligned}$$

Now, without loss of generality, we can assume that $\Theta_{i\neq j}$ is false. This is possible because our experiments later show that the SLIQ or majority voting procedures are both very accurate at predicting relative orientation (Table 2) so we can first determine the relative orientations of the contigs and flip the orientation of one contig if required. Thus we have

$$L + (o_i - o_j) < l_i \Rightarrow (P_i > P_j). \tag{9}$$

In addition, we introduce two filters that are very useful in practice for removing unreliable MPRs. To derive the first filter, if $P_j < P_i$,

$$\begin{aligned} L &= l_j - o_j + g_{ji} + o_i + R, \\ &\geq l_j - o_j - R + o_i + R, \\ o_j - o_i &\geq l_j - L, \\ o_i - o_j &< -l_j + L. \end{aligned} \tag{10}$$

The second filter is to discard an MPR if it passes the test for both $P_i < P_j$ and $P_j < P_i$.

2.4. Using the Gap Equations to Predict Relative Orientations

So far, we have only predicted relative positions when $\Theta_{i=j}$. Now we show that we can also use the gap equations to infer the relative orientations of the contigs. First, if $(P_i < P_j)$ and the minimum gap is $-R$, then we have

$$g_{ij} = L - l_i + o_i - o_j - R \geq -R. \tag{11}$$

Similarly, if $(P_j < P_i)$, then we define $\bar{g}_{ji}$ and write

$$\bar{g}_{ji} = L - l_j + o_j - o_i - R \geq -R. \tag{12}$$

Note that $\bar{g}_{ji}$ is different than g_{ji}, which we defined under the assumption $P_i < P_j$ in Equation (7).

Since $(P_i < P_j)$ and $(P_j < P_i)$ are mutually exclusive and exhaustive neglecting $P_i = P_j$, at least one of the Equations (11) and (12) will be true. Note that possibly also both could be true. For example, if $P_i < P_j$ then $g_{ij} \geq -R$. Now $(P_j < P_i)$ must be false, but that does not imply that $\bar{g}_{ji} \geq -R$is false. If both Equations (11) and (12) are true, then we can add them to get $2L \geq l_i + l_j$. To summarize,

$$\begin{aligned} ((g_{ij} \geq -R) \wedge (\bar{g}_{ji} \geq -R)) &\Rightarrow 2L \geq l_i + l_j, \\ 2L < l_i + l_j &\Rightarrow (\neg(g_{ij} \geq -R) \vee \neg(\bar{g}_{ji} \geq -R)) \end{aligned}$$

Recalling again that at least one of the Equations (11) and (12) are true, we see that $2L < l_i + l_j$ is a sufficient condition for mutual exclusion (the XOR relation is denoted by $\oplus$):

$$\begin{aligned} \Theta_{i=j} \wedge (2L < l_i + l_j) &\Rightarrow (g_{ij} \geq -R) \oplus (\bar{g}_{ji} \geq -R), \\ \neg((g_{ij} \geq -R) \oplus (\bar{g}_{ji} \geq -R)) &\Rightarrow \neg(\Theta_{i=j} \wedge (2L < l_i + l_j)), \\ \neg((g_{ij} \geq -R) \oplus (\bar{g}_{ji} \geq -R)) &\Rightarrow (\Theta_{i \neq j} \vee (2L \geq l_i + l_j)). \end{aligned}$$

If we use this equation only when the MPR and contigs satisfy the inequality $2L < l_i + l_j$, we can then make the relative orientation prediction

$$\neg((g_{ij} \geq -R) \oplus (\bar{g}_{ji} \geq -R)) \Rightarrow \Theta_{i \neq j}. \tag{13}$$

Intuitively, the condition $2L < l_i + l_j$ means that the contig lengths should be large relative to the insert length in order for the SLIQ method to work. To find contigs of the same orientation, we arbitrarily flip one contig and run the above tests again, only this time if Equation (13) holds, then we conclude that the contigs were actually of the same orientation. Say we flip C_i. We call the new offset $o_{\hat{i}}$. Then

$$\neg((g_{\hat{ij}} \geq -R) \oplus (\bar{g}_{\hat{ji}} \geq -R)) \Rightarrow \Theta_{\hat{i} \neq j} \Rightarrow \Theta_{i=j}.$$

Again, we introduce two additional filters that are very useful in practical applications. First, if we find an MPR that predicts both $\Theta_{i \neq j}$ and $\Theta_{i=j}$, then we leave it out of consideration. Second, if the SLIQ equations imply $\Theta_{i \neq j}$, then we require that both the reads of the MPR have the same mapping directions on the contigs and similarly for $\Theta_{i=j}$.

We summarize our results in the following lemmas and Algorithm 2.

Lemma 1. *If the maximum overlap between contigs is R and $2L < l_i + l_j$, then*

$$\begin{aligned} \neg((g_{ij} \geq -R) \oplus (\bar{g}_{ji} \geq q - R)) &\Rightarrow \Theta_{i \neq j}, \\ \neg((g_{\hat{ij}} \geq -R) \oplus (\bar{g}_{\hat{ji}} \geq -R)) &\Rightarrow \Theta_{i=j}. \end{aligned}$$

Lemma 2. *If the maximum overlap between contigs is R, the contigs have the same orientation, (i.e., $\Theta_{i=j}$), then*

$$(L + (o_i - o_j) < l_i) \Rightarrow (P_i > P_j).$$

We also summarize the SLIQ inequalities,

$$\begin{aligned} g_{ij} - g_{ji} &\geq l_i - l_j - 2R, \\ g_{ij} + g_{ji} &= -(l_j + l_i), \\ (g_{ij} - g_{ji} \geq l_i + l_j - 2R) &\Longleftrightarrow (P_i < P_j), \\ g_{ij} - g_{ji} &= 2L - 2R + 2(o_i - o_j) + (l_j - l_i). \end{aligned}$$

Algorithm 2. Construct Contig Digraph

Require: *input:* M = a set of MPRs connecting contigs, C = a set of contigs, w = cutoff weight
1: Define $E' = \{(C_i, C_j)$: an MPR connects C_i and $C_j\}$
2: Let $wt(i,j)$ = (number of MPRs suggesting that C_i and C_j have the same orientation) - (number of MPRs suggesting that C_i and C_j have different orientations)
3: $E = \{(C_i, C_j) : (i,j) \in E' \wedge wt(i,j) \geq w\}$
4: Construct a contig graph G with vertex set C and edge set E.
5: Find a good orientation assignment ($\Theta = \{\Theta_1, \Theta_2, \ldots\}$) for the contigs, for example, by finding a spanning tree of G.
6: Set $M_p = \{\}$
7: **for all** $p : p \in M$ **do**
8: Let C_i and C_j be the contigs connected by p.
9: **if** $\Theta_{i=j}$ **then**
10: **if** $(L + (o_i - o_j) < l_i)$ AND $(o_i - o_j < -l_i + L)$ **then**
11: predict $P_i > P_j$
12: $M_p = M_p \cup \{p\}$
13: **end if**
14: **if** $(L + (o_j - o_i) < l_j)$ AND $(o_j - o_i < -l_j + L)$ **then**
15: predict $P_i < P_j$
16: $M_p = M_p \cup \{p\}$
17: **end if**
18: **end if**
19: **end for**
20: Let $E(i,j)$ be the set of MPRs from M_p that predict $P_i < P_j$ and $E(j,i)$ be the set of MPRs from M_p that predict $P_j < P_i$.
21: Define $E_d = \{(C_i, C_j) : |E(i,j)| > |E(j,i)|\}$
22: Output a contig digraph G_d with vertex set C and edge set E_d.

2.5. *Illustrative cases and examples from real data*

In this section, we present two illustrative cases that provide the intuition underlying the SLIQ equations. The ideal case for an MPR connecting two contigs is illustrated in Figure 1. In that case, the contigs are long compared to the insert length, and the reads are mapped to the ends of the contigs. However, this situation does not always occur. Suppose the contigs are short such that the two reads of an MPR fall exactly in the center of the contigs. Then, the right-hand side of Equation (8) reduces to $2L - 2R$. So for both cases, $P_i < P_j$ and $P_j < P_i$, the right-hand side of Equation (8) has the same value, making it impossible to predict the relative positions of the two contigs. This situation is illustrated in Figure 4 on the left. It is easy to see that prediction becomes easier as the contigs get longer and the reads move away from the center of the contigs.

Now assume that the working assumption is $P_i < P_j$ but in reality, the reverse ($P_j < P_i$) is true. Then given that the contigs are long and reads map to the edges of the contigs, the insert length L would suggest the scenario depicted in Figure 4 (right side). This would make both g_{ij} and g_{ji} [as calculated from Equations (6) and (7)] smaller than they should be. In reality, the position of the contigs is similar to that shown in Figure 1, where we see that both g_{ij} and g_{ji} are larger than in Figure 4 (right side). These wrong

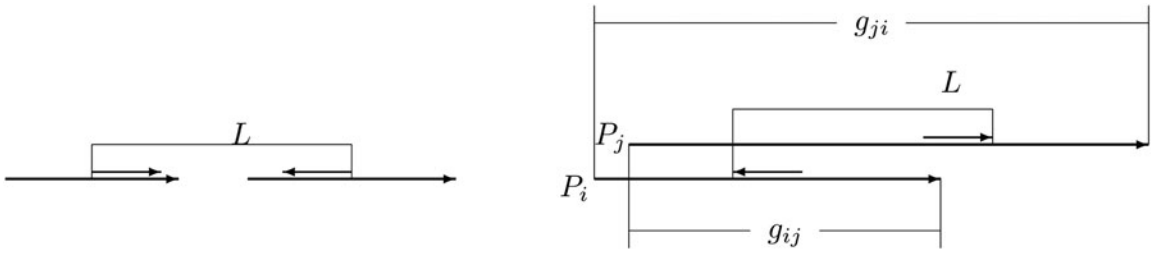

FIG. 4. Illustrative cases in which both reads of the MPR fall in the center of the contigs (left) and the contigs have reversed positions (right).

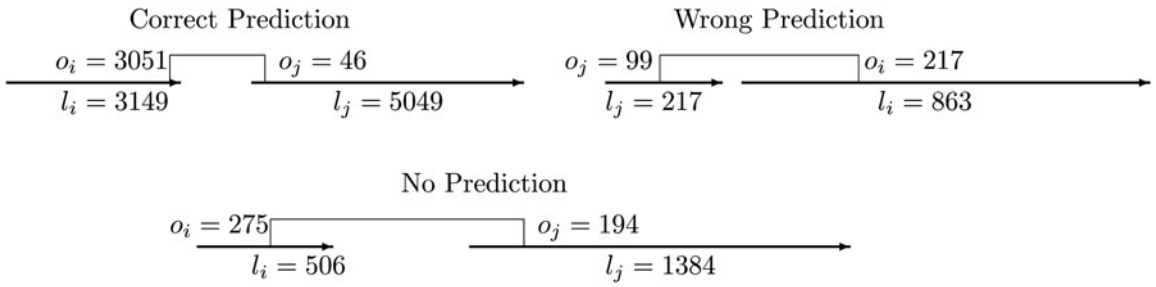

FIG. 5. Three real examples of SLIQ predictions from the PSY dataset. For the correct prediction, the equation $L+(o_i-o_j)<l_i$ evaluates to $3385<5043$. In the wrong prediction, it should have satisfied $L+(o_j-o_i)<l_j$ but one of the contigs is smaller than the insert length so it evaluates to $262<217$ (false). However $L+(o_i-o_j)<l_i$ evaluates to $498<863$ so the wrong prediction is made. In the no-prediction case, the condition $o_i-o_j<-l_j+L$ is violated. Even if that did not fail, since one of the offsets falls almost in the center of a contig, both the conditions $L+(o_j-o_i)<l_j$, $(299<1384)$ and $L+(o_i-o_j)<l_i$, $(461<506)$ are satisfied, and we would not give a prediction for this MPR. To simplify the calculations we used $L=80$.

values would then be too small to satisfy the left-hand side of Equation (5) and this would demonstrate that the working assumption of $P_i < P_j$ is wrong.

It is also instructive to consider examples from real data. We show three cases from a real data set: One in which SLIQ made a correct prediction, one in which SLIQ made a wrong prediction, and one where SLIQ did not make any predictions (Fig. 5). We explain precisely which inequalities are violated in the figure caption. The real examples show the difficulties of making SLIQ predictions when the reads fall close to the center of a contig or when the contig lengths are small relative to the insert size.

2.6. *Naive scaffolding algorithm*

The contig digraph constructed in Algorithm 2 can be directly processed to build linear scaffolds. To illustrate this point, here we present a naive scaffolding algorithm (Algorithm 3).

Algorithm 3. Naive Scaffolder

1: $G(V,E)$ = Construct Contig Digraph (Algorithm 2)
2: Identify and remove junctions from G. Junctions are defined as articulation nodes with degree ≥ 3 that connect at least 3 subgraphs of G of size larger than some given threshold. The size of a subgraph is defined as the sum of all contig sizes in that subgraph.
3: Identify all simple cycles in G and remove the edge with the lowest weight from each simple cycle.
4: If G still contains strongly connected components, those components are removed. G is now a directed acyclic graph.
5: Output each weakly connected component of G as a separate scaffold.
6: The order of contigs in each scaffold is computed by taking the topological ordering of the nodes of their respective weakly connected component in G.

To analyze the computational complexity of the naive scaffolding algorithm, let N be the number of MPRs in the library. Constructing G takes $O(N)$ time. Finding articulation points takes $O(n+m)$ time, where $n = |V|$ and $m = |E|$(Hopcroft and Tarjan, 1973). If we have a articulation nodes, then finding junctions takes $O(an)$ time. Identifying and breaking simple cycles takes $O((n+m)(c+1))$ time, where c is the number of simple cycles (Johnson, 1975). Finally, topological sorting takes $O(n+m)$ time. In total, the complexity of the naive scaffolding algorithm is $O(N)+O(n+m)+O(an)+O((n+m)(c+1)) = O(N)+O(an)+O((n+m)(c+1))$. In practical data sets, a and c are small constants and $N \gg n,m$. Thus, for practical purposes the time complexity of the algorithm is $O(N)$.

3. EXPERIMENTAL RESULTS

To demonstrate the performance of our algorithms in practice, we ran them on five real data sets and two synthetic data sets. The data sets represent genomes ranging in size from small bacterial genomes (3 Mb)

Table 1. Descriptive Statistics About the Datasets

Set ID	*Organism*	*Size*	*Ref. genome*	*Read lib*	*R*	*cov*	*L*	L_r	σ
PSU	*P. suwonensis*	3.42 Mb	CP002446.1	SRR097515	76	870x	300	188.78	18.77
PSY	*P. syringae*	6.10 Mb	NC_007005.1	(Farrer et al., 2009)	36	40x	350	384.11	67.13
SY-CE	*C. elegans*	100.26 Mb	NC_003279-85	SRR006878	35	38x	200	232.13	54.44
PST	*P. stipitis*	15.40 Mb	(Chapman et al., 2011)	(Chapman et al., 2011)	75	25x	3.2K	3.27K	241.50
DS	*D. simulans*	109.69 Mb	NT_167066.1-68.1, NT_167061.1, NC_011088.1-89.1, NC_005781.1	SRR121548, SRR121549	36	62x	N/A	187.99	61.47
SY-HS	*H. Sapiens*	3.30 Gb	NCBI36/ hg18	ERA015743	100	45x	300	310.63	20.74
HS	*H. Sapiens*	3.30 Gb	NCBI36/ hg19	ERA015743	100	45x	300	310.63	20.74

R, read length; cov, coverage; *L*, reported insert length; L_r, the real insert length calculated by mapping reads to the reference genome; σ, standard deviation of L_r.

to large animal genomes (3.3 Gb) (see Table 1 for details). More importantly, they also vary in repetitiveness—from almost nonrepetitive bacteria to moderately repetitive drosophila to highly repetitive human genomes.

For each data set, we obtained a publicly available mate-pair library. We used publicly available pre-built contigs for the *Drosophila simulans* (DS) and human (HS) (Gnerre et al., 2011) data sets. Pre built contigs were not available for the three microbial data sets—*P. suwonensis* (PSU), *P. syringae* (PSY), and *P. stipitis* (PST) — so we used the short read assembler VELVET (Zerbino and Birney, 2008) to construct contigs. All software parameters and sources for the data are provided in Table 4. For the two synthetic datasets, *C. elegans* (SY_CE) and human (SY_HS), we constructed contigs by mapping reads back to the reference genome and declaring high-coverage regions to be contigs. So, for these experiments, we have synthetic contigs but real reads. We will discuss the performance of the algorithms on the synthetic data sets at greater length in the Discussion. We mapped the reads to the contigs using the program Bowtie (v. 0.12.7) (Langmead et al., 2009). Below we only report results for the uniquely mapped reads because we know the ground truth for them.

3.1. Comparison of SLIQ and majority voting predictions

On all the real data sets, SLIQ was highly accurate in predicting both relative orientation (>75%) and position (>80%) (Table 2). For orientation prediction, SLIQ and majority filtering produced almost identical accuracies except for the case of *P. stipitis* (PST), where SLIQ had lower accuracy (75% vs 97%). One possible reason might be that the PST library used long mate-pair reads, which may be more inaccurate than the other libraries we tested. Conversely, for PST, majority voting gave far worse accuracy (16.5%) than SLIQ (75%) in relative position prediction, confirming that this data set is an outlier.

Focusing only on the position predictions, SLIQ showed a significant advantage in both the number and accuracy of the predictions compared to majority voting for the more complex genomes — *D. simulans* and human (Fig. 6). Importantly, the improvement was particularly large for the human genome.

Table 2. Summary of the Results of SLIQ vs. Majority Filtering for Contig Graph Edges of Five Real Datasets

Set ID	n	w_e	n_o	e_o	n_p	e_p	w_m	n'_o	e'_o	n'_p	e'_p
PSU	4454	2	2507	99.69%	3803	99.21%	4	3942	99.59%	3925	94.87%
PSY	2086	2	1628	98.40%	1852	95.62%	4	2019	98.56%	1990	98.59%
PST	2291	1	1233	75.18%	1516	87.33%	2	1365	97.87%	1336	16.54%
DS	8738	1	6305	92.18%	7097	80.55%	2	6390	91.87%	5861	77.25%
HS	36346	1	31799	79.56%	31153	89.71%	2	32676	79.14%	25750	75.62%

n, total number of edges connecting two different contigs; w_e, minimum wieght of an edge for SLIQ prediction; n_o, the number of edges for which we can predict relative orientation, e_o, the accuracy of relative orientation prediction, n_p, the number of edges for which we can predict relative position; e_p, the accuracy of relative position prediction; w_m, minimum weight of an edge for majority prediction. The same notations are used for majority filtering except with prime.

Table 3. Comparison of Position Predictions Between the SLIQ and Majority Voting Methods

Set ID	n_a	n_d	n_{de}	n_{dm}	n'_e	e_q	n'_m	e_m
PSU	3089	646	643	3	68	95.58%	190	90.52%
PSY	1519	287	235	52	46	86.95%	184	96.19%
PST	290	794	784	10	432	58.56%	252	25.00%
DS	2447	820	804	16	409	93.15%	2035	76.41%
HS	16425	2017	1852	165	12711	85.67%	7308	52.73%

n_a, the number of predictions where the methods agreed; n_d, the number of predictions where the methods disagreed; n_{de}, the number of predictions not in agreement where SLIQ was correct, n_{dm}, the number of predictions not in agreement where majority voting was correct; n'_e, is the number of predictions made only by SLIQ; e_q, the accuracy of predictions made only by SLIQ; n'_m, the number of predictions made only by majority voting; e_m, the accuracy of predictions made only by majority voting.

Finally, Table 3 gives a more detailed comparison of cases in which the SLIQ and majority voting predictions disagreed. When the two methods disagreed, SLIQ clearly outperformed majority voting procedure. For example, for human, when the methods disagreed, SLIQ was right in 1852 cases and majority voting in only 165 cases. SLIQ was also generally more accurate when considering only the predictions made uniquely by each method, except in one case (PSY).

3.2. Computing the optimal insert length

In our experiments, we found that using a slightly larger value for L (e.g., 20 bp for PSY) than that reported or estimated increased both n_p(by 49), the number of MPRs for which we could make a relative position prediction, and e_p (by 2%), the accuracy of relative position prediction. This may seem surprising at first given Equation (9). However, for n_p, it can be seen from Figure 1 that underestimating L would reduce g_{ij}, which would lead to more overlaps between contigs. Since we assume that the maximum contig overlap is R, underestimating L would remove many MPRs from the predictions. However, at the moment we do not have an explanation for the observed increase in e_p, the prediction accuracy.

On the other hand, using a slightly smaller value for L increased n_o, the number of MPRs for which we could make a relative orientation prediction, while e_o, the prediction accuracy for orientation, remained constant. We suspect that a lower L makes Equations (11) and (12) harder to pass and thus less MPRs are excluded by the mutual exclusion test.

3.3. Computing the rank of MPRs

Our experimental results also agree with our illustrative cases (section 2.5) in that the prediction accuracy decreases as $2(o_i - o_j)$ gets closer to $(l_i - l_j)$ which intuitively means that the reads are falling closer to the center of the contigs. To address this issue we can rank the MPRs by the minimum value of c for

Table 4. Parameter Values Used in the Analysis of all Datasets

Data set	v	*Contig construction*	*Contig mapping*
PSU	2	(velvet) Hash length = 21, cov_cutoff = 5, min_contig_lgth = 150	(vmatch) Min match length l = 150, Hamming distance h = 0
PSY	0	(velvet) Hash length = 21, cov_cutoff = 5, min_contig_lgth = 150	(vmatch) Min match length l = 150, Hamming distance h = 0
PST	0	(velvet) Hash length = 35, cov_cutoff = auto, min_contig_lgth = 100	(vmatch) Min match length l = 200, Hamming distance h = 5
SY-CE	1	(synthetic) Cov cutoff = 5, min contig len = L	Available from synthetic construction
DS	2	accession number AASR01000001-AASR01050477	(vmatch) Min match length l = 200, Hamming distance h = 5
SY-HS	2	(synthetic) Cov cutoff = 3, min contig len = $2R$	Available from synthetic construction
HS	3	Accession number AEKP01000001:AEKP01231194	(vmatch) Min match length l = 300, Hamming distance h = 0

v is the number of mismatches allowed in read mapping (Bowtie v.0.12.7).

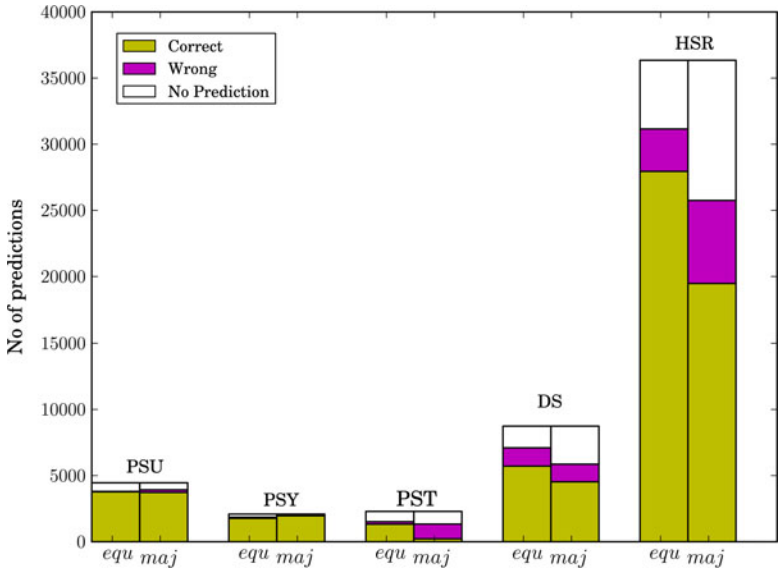

FIG. 6. Comparison of the accuracy of SLIQ and majority voting for relative position prediction using that same data shown in Table 2.

which they fail to pass the more stringent inequality $|2(o_i - o_j) - (l_i - l_j)| > cR$. We say that an MPR has rank c if and only if c is the smallest positive integer such that $|2(o_i - o_j) - (l_i - l_j)| \leq cR$, and MPRs with higher rank are considered more confident with regards to their prediction. Figure 7 shows how the prediction accuracy depends on the rank of the MPRs in the PSY dataset.

3.4. Effect of the number of Mate Pairs

More mate pairs connecting different contigs give better confidence in scaffolding. But we observed that this improvement is significant up to a certain threshold (4–5 for majority voting and 2–3 for the SLIQ equations). After that, the improvement in correctness in scaffolding is not worth the reduction in number of edges in the contig graph. For example, for the DS dataset, if we increase the cutoff by 1, the position prediction improves by 3% but reduces the number of edges by 1520. This reduction also depends on the coverage of the read library. For the high coverage PSU dataset, an increase of 1 in cutoff has almost no effect– a reduction of 50 edges. And of course, all this is assuming that the contigs are of reasonable

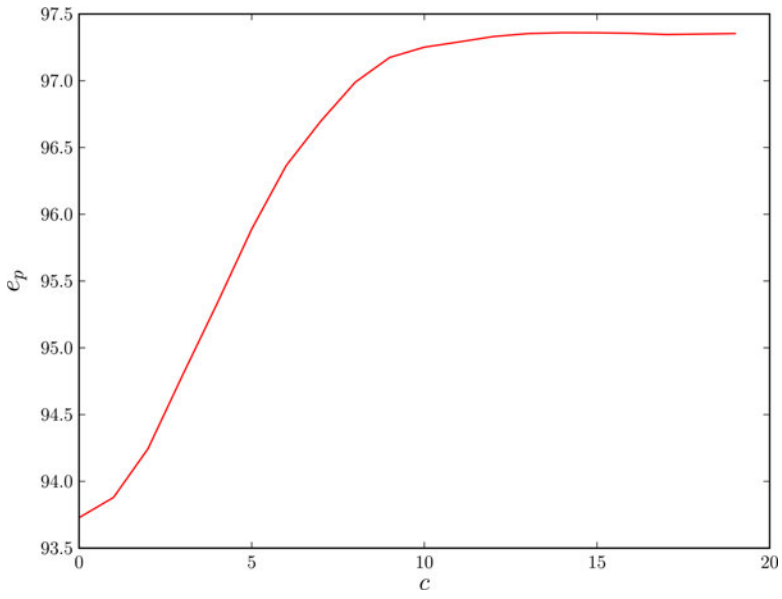

FIG. 7. Change in the prediction accuracy, e_p, as we restrict our analysis to MPRs of higher rank (c).

Table 5. Summary of the Results of Majority Prediction for Synthetic Datasets for *C. elegans* (SY_CE) and Humans (SY_HS)

Data set	n	w_m	n_o	e_o	n_p	e_p
SY-CE	17620	3	17620	99.52%	17532	99.85%
SY-HS	878380	3	878380	98.93%	868877	99.47%

n, total number of edges connecting two different contigs; w_m, minimum weight of an edge for majority prediction; n_o, the number of edges for which we can predict relative orientation; e_o, the accuracy in relative orientation prediction; n_p, the number of edges for which we can predict relative position; e_p, the accuracy in relative position prediction.

Table 6. Summary of the Results of Our Naive Scaffolder on Real Data

Data set	*N50*	*Genome coverage*	*Orientation accuracy*	*Position accuracy*
PSU	17K	116.1%	99.64%	97.95%
PSY	75K	90.98%	98.26%	93.42%
PST	215K	97.89%	98.90%	89.89%
DS	942	59.48%	97.52%	96.07%
HS	18K	79.27%	98.28%	98.03%

N50 is the length n such that 50% of bases are in a scaffold of length at least n. The position accuracy measures how many neighboring contigs in the scaffold were placed in the correct order.

Table 7. Run time Comparison of Our Naive Scaffolder with Two Other State-of-the-Art Scaffolders, SOPRA and MIP Scaffolder

Data set	*Naive Scaffolder*	*SOPRA*	*MIP Scaffolder*
PSU	6m40.39s	237m27.237s	23m32.55s
PSY	59.36s	44m57.604s	3m14.03s
PST	67.21s	3009m29.224s	124m42.68s
DS	7m7.449s	N/A	36m42.05s
HS	241m33.928s	N/A	N/A

All times are the sum of the user and system times reported by the Linux time command. We ran all software on a 48 core Linux server with 256GB of memory.

quality. If you have mis-assembled or chimeric contigs, more mate pairs can create more loops and high-degree nodes in the contig graph, which are not removed by the cutoff threshold and result in worse scaffolding.

3.5. Performance of the naive scaffolder

We summarize the results of our naive scaffolder on the five real data sets in Table 6 and Table 7. For all data sets, the orientation accuracy was very high ($>97\%$) and the position accuracy was also high ($>89\%$). While the genome coverages of PSU and DS may appear surprising, note that the PSU library had a very high coverage while the DS library had low coverage and was also made up of a number of different *D. simulans* strains. It is likely that the PSU contigs include misassembled fragments in the contigs, making the total length of the contigs larger than the genome size. For DS, the combination of low coverage and relatively high rates of sequence differences between the different *D. simulans* strains likely resulted in lower genome coverage.

4. DISCUSSION

In conclusion, we have presented a mathematical approach and an algorithm for constructing a contig digraph that encodes the relative positions of contigs based on mate-pair read data. Our main insight is the

derivation of a set of simple linear inequalities derived from the geometry of contigs on the line that we call SLIQ. We can use SLIQ both to efficiently filter out unreliable mate-pair reads (MPR) and predict the relative positions and orientations between contigs. We have shown that SLIQ outperforms the commonly used majority voting procedure for the prediction of relative position of contigs while both methods are very accurate for orientation prediction. The contig digraph can also be directly processed into a set of linear scaffolds and we have presented a simple scaffolding algorithm for doing so. Our naive scaffolder has high accuracy on all data sets tested and is very efficient—for practical purposes, as it takes time linear in the size of the mate pair library and it is also very fast compared to other state-of the art scaffolders. The output of our naive scaffolder can either be used directly as draft scaffolds or used as a reasonable starting point for refinement with more complex optimization procedures used in other scaffolders.

One interesting and unexpected finding of our experiments was that the simple majority voting procedure performs very well for predicting the relative positions of contigs if the contigs have few errors. This can be seen by the performance of the majority voting procedure when using synthetic contigs that are not constructed using *de novo* assembly tools but rather by mapping the reads back to a reference genome and identifying regions of high coverage, which is expected to produce much higher quality contigs (Table 5). This observation suggests a novel way to approach the scaffolding problem in which the contig builder would output smaller but higher quality contigs and allow the scaffolder to handle the remainder of the assembly. We believe this is a significant change in philosophy of genome assembly programs to date in which during the contig building step, one generally attempts greedily to build contigs that are as long as possible. This viewpoint also differs considerably from previous approaches to scaffolding in which the focus was on resolving conflicts between mate pairs that gave conflicting information about the relative orientation and position of contigs.

Finally, we are exploring several possible extensions of the SLIQ method. The first extension is to find the optimal value for L, the insert length, so that we optimize the number and accuracy of relative position and orientation predictions. The second extension is to assign numerical values to the accuracy of prediction of MPRs of a particular rank. Finally, for the multiply mapped MPRs, which were not included in the results, we plan to identify the most likely mapping for the MPR, for example, by using their ranks.

DISCLOSURE STATEMENT

No competing financial interests exist.

REFERENCES

Batzoglou, S., Jaffe, D.B., Stanley, K., et al. 2002. Arachne: a whole-genome shotgun assembler. *Genome Res* 12, 177–189.

Boetzer, M., Henkel, C.V., Jansen, H.J., et al. 2011. Scaffolding pre-assembled contigs using SSPACE. *Bioinformatics* 27, 578–579.

Butler, J., MacCallum, I., Kleber, M., et al. 2008. Allpaths: de novo assembly of whole-genome shotgun microreads. *Genome Res.* 18, 810–820.

Chapman, J.A., Ho, I., Sunkara, S., et al. 2011. Meraculous: de novo genome assembly with short paired-end reads. *PLoS One* 6, e23501.

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

Farrer, R.A., Kemen, E., Jones, J.D.G., et al. 2009. De novo assembly of the pseudomonas syringae pv. syringae b728a genome using Illumina/Solexa short sequence reads. *FEMS Microbiol Lett.* 291, 103–111.

Gao, S., Nagarajan, N., and kin Sung, W. 2011. Opera: Reconstructing optimal genomic scaffolds with high-throughput paired-end sequences. *LNBI.* 6577, 437–451.

Garey, Michael, R, and Johnson, D.S. 1979. Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman, New York.

Gnerre, S., Maccallum, I., Przybylski, D., et al. 2011. High-quality draft assemblies of mammalian genomes from massively parallel sequence data. *Proc Natl Acad Sci U S A.* 108, 1513–1518.

Haubold, B., and Wiehe, T. 2006. How repetitive are genomes? *BMC Bioinformatics.* 7, 541.

Hopcroft, J., and Tarjan, R. 1973. Efficient algorithms for graph manipulation. *Communications of the ACM.* 16, 372–378.

Huson, D.H., Reinert, K., and Myers, E. 2002. The greedy path-merging algorithm for contig scaffolding. *Journal of the ACM.* 49, 603–615.

Johnson, D.B. 1975. Finding all the elementary circuits of a directed graph. *SIAM Journal on Computing.* 4, 77–84.

Koren, S., Treangen, T.J., and Pop, M. 2011. Bambus 2: scaffolding metagenomes. *Bioinformatics*. 27, 2964–2971.

Langmead, B., Trapnell, C., Pop, M., et al. 2009. Ultrafast and memory-efficient alignment of short dna sequences to the human genome. *Genome Biol.* 10, R25.

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

Pop, M., Kosack, D.S., and Salzberg, S.L. 2004. Hierarchical scaffolding with Bambus. *Genome Res.* 14, 149–159.

Salmela, L., Mkinen, V., Vlimki, N., et al. 2011. Fast scaffolding with small independent mixed integer programs. *Bioinformatics* 27, 3259–3265

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

Address correspondence to:
Rajat S. Roy
Department of Computer Science
Rutgers, The State University of New Jersey
110 Frelinghuysen Road
Piscataway, NJ 08854-8019

E-mail: rajatroy@cs.rutgers.edu