The Greedy Path-Merging Algorithm for Contig Scaffolding

DANIEL H. HUSON

Tübingen University, Tübingen Germany

KNUT REINERT

Free University Berlin, Germany

AND

EUGENE W. MYERS

University of California Berkeley, USA

Abstract. Given a collection of contigs and mate-pairs. The *Contig Scaffolding Problem* is to order and orientate the given contigs in a manner that is consistent with as many mate-pairs as possible. This paper describes an efficient heuristic called the *greedy-path merging algorithm* for solving this problem. The method was originally developed as a key component of the *compartmentalized assembly* strategy developed at Celera Genomics. This interim approach was used at an early stage of the sequencing of the human genome to produce a preliminary assembly based on preliminary whole genome shotgun data produced at Celera and preliminary human contigs produced by the Human Genome Project.

Categories and Subject Descriptors: F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*computations on discrete structures*; J.3 [**Life and Medical Sciences**]: *biology and genetics*

General Terms: Algorithms, Experimentation

Additional Key Words and Phrases: Genome assembly

1. *Introduction*

In February 2001, the publicly-funded Human Genome Project (HGP) reported on the "initial sequencing and analysis of the human genome" [International Human Genome Sequencing Consortium 2001], based on a "clone-by-clone" assembly of the human genome [U. S. Dept. of Energy et al. 1997]. Simultaneously, Craig

Authors' addresses: D. H. Huson, Institute of Computer Science, Tübingen University, Sand 14, D-72076 Tübingen, Germany, e-mail: huson@informatik.uni-tuebingen.de; K. Reinert, Institute of Computer Science, Free University Berlin, Takustr. 9, D-14195 Berlin, Germany, e-mail: reinert@inf.fu-berlin.de.; E. W. Myers, Department of Computer Science, University of California Berkeley, Berkeley, CA 94720, e-mail: gene@eecs.berkeley.edu.

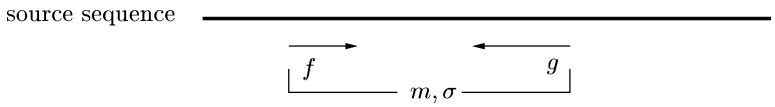

FIG. 1. Two fragments f and g that form a mate-pair with known mean distance m and standard deviation σ. Note their relative orientation in the source sequence.

Venter and colleagues at Celera Genomics published an initial assembly of the human genome [Venter et al. 2001], based on the "whole genome shotgun (WGS)" approach [Webber and Myers 1997], using Celera whole genome shotgun data and shredded public contigs.

Additionally, Venter et al. [2001] describes a *compartmentalized shotgun assembler* [Huson et al. 2001] that adds a hierarchical compartmentalization step to the whole genome shotgun approach and was used at an early stage of the sequencing project. The aim of this article is to introduce the *Contig Scaffolding Problem*, which arises in this context, and to present a new method, the *greedy path-merging algorithm*, that we have developed and employed to heuristically solve the problem.

Although current technology for DNA sequencing is highly automated and can determine large numbers of base pairs very quickly, only (on average) about 550 *consecutive* base pairs can be reliably read at a time [Sanger et al. 1977]. Thus, a larger stretch of consecutive DNA can only be determined by assembling it from fragments.

In the HGP's clone-by-clone approach, one first constructs a tiling of the genome by overlapping pieces, each typically of length up to 150 kb, and then concentrates on determining the sequence of each such piece. We will call these pieces *BAC clones* or simply *BACs*, since they are usually cloned using "Bacterial Artificial Chromosome" vectors.

One then concentrates on determining the sequence of each BAC using *shotgun sequencing* [Sanger et al. 1992]: To this end, the BAC is randomly broken into many small fragments which are then each individually cloned and sequenced. For a successful assembly, this has to be done with a sufficient amount of oversampling. Statistical calculations [Lander and Waterman 1988] suggest that the average number of fragments covering any given site in the BAC should be about 7. The sequenced fragments are then run through an assembly program that attempts to construct the full sequence of the BAC from them, by determining how these fragments of sequence overlap with each other. The result is one or more *contigs*, that is, pieces of contiguous sequence. More recently, the fragments are collected in *mate-pairs*, see Figure 1. This is done by sequencing larger pieces of DNA from both ends, thus producing pairs of sequenced fragments with known relative orientation and approximate distance (typically, employing a mixture of 2-kb, 1-kb, 50-kb, and 150-kb clones).

In the WGS strategy, the *whole* genome is randomly broken into small pieces that are individually sequenced, again with ideally at least 7-fold sequence coverage. These reads are then assembled using a whole genome shotgun assembler [Myers et al. 2000; Venter et al. 2001]. Due to the abundance of repeats in genomic DNA, a purely overlap-based approach to WGS assembly is not feasible and the use of mate-pairs is crucial.

For the purposes of this article, a BAC is simply a collection of contigs that are assumed to come from a common "source region" of approximately 150 kb of

TABLE I. LISTED BY PHASE, WE REPORT THE NUMBER OF BACS, AND THE AVERAGE NUMBER AND LENGTH OF CONTIGS IN GENBANK ON SEPTEMBER 1, 2000

Phase	# BACs	avg. # btgs	avg. length (bp)
0	2967	91.5	784
1/2	20960	19.8	8102
3	9494	1	94309

TABLE II. FRAGMENTS ARE ORGANIZED IN MATE-PAIR LIBRARIES, WHICH EACH HAVE A MEAN DISTANCE AND STANDARD DEVIATION ASSOCIATED WITH THEM

Library types	# fragments	# mate-pairs	mean distance	standard deviation
2k	10 m	5 m	2000	100
10k	8 m	4 m	10000	1000
50k	1.8 m	0.9 m	40000–60000	5000–15000
100k	0.7 m	0.35 m	90000–150000	25000–40000

This table was compiled from 24 individual libraries and reflects the numbers of mate-pairs for the 28 million fragments available at Celera as of May 2000.

contiguous DNA in the human genome, using a shotgun sequencing and assembly process [Green 1994].

The BACs sequenced by the HGP are submitted to GenBank [Benson et al. 2000] on a regular basis. Such a GenBank entry usually evolves over time, as more work is done to determine the BACs full sequence: Originally, a BAC may start out as a *phase-0* entry, which means that it typically consists of about 60–100 contigs, each of length 700–1000. A *phase-1* or *phase-2* BAC will typically consist of 15–25 contigs of length 5000–20000, whereas a *phase-3* BAC consists of precisely one contig that represents the full source sequence, the latter phase reached by means of finishing reads. Note that contigs associated with a phase-0 or phase-1 BAC are considered unordered and unoriented, whereas a phase-2 BAC comes with additional information on how the contigs are ordered and oriented with respect to each other. As of September 1, 2000, GenBank contained 33421 relevant human BACs (see Table I).

Celera's fragment data consists of about 28 million fragments of human DNA, each between 150 and 800 bp long. The majority of them come in mate-pairs of known relative orientation and approximate distance. Paired mates are organized in *libraries*, each with an associated mean distance and standard deviation, see Table II.

The compartmentalized assembler takes as input the BACs and Celera's fragments and proceeds in the following steps: (1) it significantly increases the level of assembly of BACs using the information given by the Celera fragments and mate-links, (2) it assembles regions of the genome not covered by BAC sequence into "scaffolds" (a *scaffold* is a collection of pieces of sequence of known relative orientation and approximate distance), (3) it produces an accurate tiling of the ordered BACs and Celera scaffolds, and (4) it assembles the connected components of this tiling using Celera's WGS assembler.

In this article, we address problem (1), namely to significantly increase the level of assembly of phase-1/2 BACs using the additional information given by the Celera fragments and mate-links, ideally promoting BACs from phase-1/2 to phase-3.

We set up some graph theoretical notation and introduce our main data structure, the *contig-mate-pair graph*, in Section 2. Then, in Section 3, we formulate the *Contig Scaffolding Problem*, and show that the associated decision problem is NP-complete. In Section 4, we discuss some simple strategies that one can employ to avoid problems caused by polymorphisms, repeats and misassembled contigs. We then present our main algorithm, called the *greedy path-merging algorithm*, in Section 5. In Section 6 we describe some additional combinatorial constraints that can be utilized by the main algorithm. This is followed in Section 7 by a discussion of how to use fragments to extend contigs and even completely close gaps between consecutive contigs. Finally, we discuss the performance of the algorithm applied to human BACs in Section 8.

For concreteness, in this article, we formulate and discuss the Contig Scaffolding Problem in the context of ordering and orienting the contigs of human BACs. We would like to emphasize, however, that the algorithms presented here apply very generally to the problem of ordering and orienting contigs using paired reads, independent of how the contigs were obtained. For example, we used this approach to scaffold partial human cDNAs using mate-pairs. Additionally, this method can be employed to create a "syntenic assembly" for example, of mouse using human: mouse contigs are ordered and oriented using human mate-pairs. We are currently studying the problem of scaffolding contigs produced by purely overlap-oriented assemblers [Green 1994], using mate-pairs. Finally, we would like to point out that the algorithms describe here are a useful model for understanding the scaffolding stage of heavy-weight assemblers such as the Celera WGS assembler.

2. *The Contig-Mate-Pair Graph*

Consider a BAC $\mathcal{B} = \{B_1, B_2, \ldots, B_n\}$ consisting of n contigs $B_1, B_2, \ldots, B_n$. Let $l(B_i)$ denote the sequence length of B_i. We say that a fragment f *hits* (or *is embedded in*) a contig B_i, if it (or its reverse complement) aligns to it with high identity. In this case, we use $p_1(f)$ and $p_2(f)$ to denote the positions between 1 and $l(B_i)$ to which the first character (5′-end) and the last character (3′-end) of f aligns to.

In the next few paragraphs, we introduce the *contig-mate-pair graph* $G = (V, E)$, with node set V and edge set E, which is a weighted, undirected multi-graph, without self-loops. It has two kinds of edges, namely *contigs edges* that represent the contigs of the given BAC and *mate-edges* that represent mate-pair links between fragments that are embedded in different contigs. The *weight* of a mate edge represents the amount of evidence that supports the edge.

2.1. The Initial Graph.

We obtain the *initial* contig-mate-pair graph as follows: Each contig B_i gives rise to precisely two nodes v, w in the graph, and these are connected by an edge e, labeled B_i, and of *length* $l(e) := l(B)$. We associate one end of B_i with the node $s(B_i) := v$ and the other end with $t(B_i) := w$.

Now consider two fragments f and g that form a mate-pair, that is, whose relative orientation and approximate distance are known. If both f and g each hit precisely one contig in $\mathcal{B}$, say B_i and B_j, and if these differ, then this mate-pair induces a relative orientation and approximate distance between the two contigs, as indicated in Figure 2. This is represented by a *mate-edge* e whose *length* $l(e)$ is set to the mate-pair distance minus the distance by which the two contigs extend

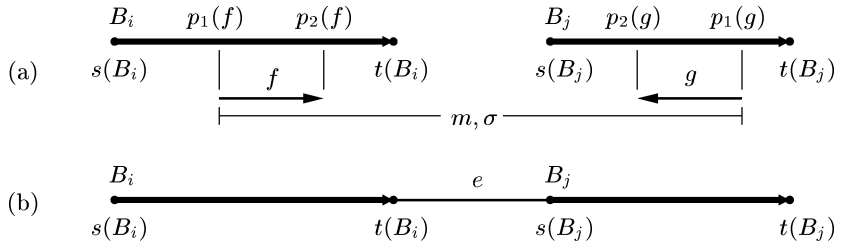

FIG. 2. (a) A pair of mated fragments f and g of mean distance m and standard deviation σ that uniquely hit two different contigs B_i and B_j, gives rise (b) to an edge e between (in this case) the nodes $t(B_i)$ and $s(B_j)$. In this example, the length of e is set to $l(e) := m - (l(B_i) - p_1(f)) - p_1(g)$.

into the mate-pair. Note that $l(e)$ may be negative, which indicates that the two contigs should *overlap* by approximately $-l(e)$ base pairs. We also associate a standard deviation $\sigma(e)$ with such a mate-edge e, coming from the mate-pair library (see Table II).

2.2. EDGE BUNDLING. If there is more than one simple mate-link between the same ends of two contigs B_i and B_j, then the corresponding mate-edges can be *bundled*, if their lengths are similar. Let M denote the set of mate-edges between nodes v and w. We perform bundling by first greedily choosing a median-length mate-edge $e \in M$, then determining all mate-edges $e' \in M$, whose length $l(e')$ is within $3\sigma(e)$ of $l(e)$, and then, finally, bundling all of these with e. We repeat this construction on the remaining edges, until no edges remain. If an edge e was obtained by bundling edges $e_1, e_2, \ldots, e_k$, then we set $l(e) := p/q$ and $\sigma(e) := 1/\sqrt{q}$, with

$$p := \sum \frac{l(e_i)}{\sigma(e_i)^2} \quad \text{and} \quad q = \sum \frac{1}{\sigma(e_i)^2},$$

following standard statistical practice for combining multiple measurements.

We set the *weight* $w(e)$ of a mate-edge e to 1, if e is a simple mate-edge, and to $\sum_{i=1}^{k} w(e_i)$, if e was obtained by bundling k mate-edges $\{e_1, e_2, \ldots, e_k\}$.

2.3. TRANSITIVE REDUCTION. After bundling, we attempt to *transitively reduce* long mate-edges: Consider two nodes v and w that are connected by an alternating path $P = (m_1, b_1, m_2, \ldots, m_k)$ of mate-edges $(m_1, m_2, \ldots)$ and contig edges $(b_1, b_2, \ldots)$ from v to w, beginning and ending with a mate-edge. We obtain a mean length and standard deviation for P by setting $l(P) := \sum_{m_i} l(m_i) + \sum_{b_i} l(b_i)$ and $\sigma(P) := \sqrt{\sum_{m_i} (\sigma(m_i))^2}$ (see e.g., Bevington [1969]).

We say that a mate-edge e from v to w can be *transitively reduced* on to the path P, if e and P approximately have the same length, that is, if $|l(e) - l(P)| \leq C \cdot \max\{\sigma(e), \sigma(P)\}$ for some constant C, typically 3 in our applications. If this is the case, then we can *reduce* e by removing e from the graph and incrementing the weight of every mate-edge m_i in P by $w(e)$.

A simple branch-and-bound enumeration of all reduction paths containing up to 5 mate-edges works well for our purposes, and is sufficiently fast, if we first compute the set of biconnected components of the graph and then only consider paths that stay within the same component, as two nodes must be in the same component to admit a reduction.

2.4. THE FINAL GRAPH. By the *final* contig-mate-pair graph $G = (V, E)$, we mean a graph obtained from the initial contig-mate-pairgraph by first performing bundling and then transitive reduction of mate-edges. For the remaining of the article, we will refer to this simply as *the* contig-mate-pair graph.

3. *The Contig Scaffolding Problem*

Consider a BAC $\mathcal{B} = \{B_1, B_2, \ldots, B_n\}$ consisting of n contigs coming from a source sequence of approximately 150 kb, and additionally, a collection of fragments that hit these contigs, together with the pertinent mate-pair information. Generally speaking, the goal is to assign coordinates to the contigs that reflect, as closely as possible, the true relative positions and orientations of the contigs in the source sequence, by making use of the additional information. We now formulate an optimization problem that captures this goal.

Let $G = (V, E)$ be the contig-mate-pair graph associated with $\mathcal{B}$. An *ordering (and orientation) of $\mathcal{B}$ or G* is a map $\phi : V \to \mathbb{N}$ such that $|\phi(s(B_i)) - \phi(t(B_i))| = l(B_i)$ for all contigs $B_i \in \mathcal{B}$, in other words, an assignment of coordinates to all nodes that preserves contig lengths. Additionally, we require $\{\phi(s(B_i)), \phi(t(B_i))\} \neq \{\phi(s(B_j)), \phi(t(B_j))\}$ for any two distinct contigs.

Consider a mate-edge e with nodes v and w. Let B_i denote the contig edge incident to v and let B_j denote the contig edge incident to w. Let v' and w' denote the other two nodes of B_i and B_j, respectively. We call *e happy* (with respect to ϕ), if B_i and B_j have the correct relative orientation, and if the distance between v and w is approximately correct, in other words, we require that either (a) $\phi(v') \leq \phi(v)$ and $|\phi(w) - \phi(v) - m(e)| \leq 3\sigma(e)$ and $\phi(w) \leq \phi(w')$, or (b) $\phi(w') \leq \phi(w)$ and $|\phi(v) - \phi(w) - m(e)| \leq 3\sigma(e)$ and $\phi(v) \leq \phi(v')$. Otherwise, e is *unhappy*.

Problem 3.1 (*Contig Scaffolding Problem*). Given a contig-mate-pair graph G. The *Contig Scaffolding Problem* is to find an ordering ϕ of G that maximizes the sum of weights of happy mate-edges.

THEOREM 3.2. *The corresponding decision problem is NP-complete.*

PROOF. For a given ordering ϕ, we can determine whether the number of happy mate-edges exceeds a given threshold by simple inspection of each of them, thus the problem is in NP. The problem that we will reduce to this is BANDWIDTH [Garey and Johnson 1979], for which the question is to decide, for a given graph $G = (V, E)$ (with node set $V = \{v_1, v_2, \ldots, v_n\}$) and number $K \leq |V|$, whether there exists a permutation ϕ of $1, 2, \ldots, n$ such that for all edges $\{v_i, v_j\} \in E$ we have $|\phi(i) - \phi(j)| \leq K$. Given an instance $G = (V, E)$ of this problem, we construct a contig-mate-pair graph $G' = (V', E')$ in polynomial time as follows: First, set $V' := V$ and $E' := E$, and let these edges be the mate-edges, setting $m(e) := 1 + (K - 1)/2$ and $\sigma(e) := (K - 1)/6$ so as to obtain a happy range of $[1, K]$, and $w(e) := 1$, for every mate-edge e. We call these initial nodes *proper*. Then, for each proper node v, add a new auxiliary node v' to V' and join v and v' by a contig edge of length 0. The answer to the BANDWIDTH question is *true*, if and only if the graph G' has an ordering ϕ such that all mate-edges in G' are happy:

A graph G has BANDWIDTH $\leq K \Leftrightarrow$ there exists a permutation ϕ such that $(v_i, v_j) \in E$ implies $|\phi(i) - \phi(j)| \leq K \Leftrightarrow$ there exists an ordering ϕ such that $(v_i, v_j) \in E$ implies $1 \leq |\phi(i) - \phi(j)| \leq K \Leftrightarrow$ there exists an ordering ϕ such that

$e = (v_i, v_j) \in E$ implies $m(e) - 3\sigma(e) \leq |\phi(i) - \phi(j)| \leq m(e) + 3\sigma(e) \Leftrightarrow$ all mate-edges of G' are happy. □

4. *Polymorphisms, Misassemblies and Repeats*

Given a good solution of the Contig Scaffolding Problem, how confident can we be that it also yields a good solution of the original biological problem, namely to reconstruct the source sequence? Polymorphisms, repeats or misassembled contigs can severely impact the quality of the result.

Both the HGP and Celera used DNA from a number of different individuals, and each individual has two copies of each chromosome. Hence, we can expect many instances of polymorphic sequence. For example, a contig may contain a stretch of inserted sequence that is not present in the source sequence for the fragments, or vice-versa. Also, it is clear that contigs may be misassembled, or "differently assembled", the latter occurring again in view of polymorphisms.

Given a BAC $\mathcal{B} = \{B_1, B_2, \ldots, B_n\}$. For each contig B_i, we compute the *fragment coverage map* $F : [1, l(B_i)] \to \mathbb{N}$ that counts how many fragments hit any given position in the contig. A variant of this is the *core fragment coverage map* $\dot{F}$, which we obtain by disregarding the first and last 30 bp, say, of every fragment. The *clone coverage map* $C : [1, l(B_i)] \to \mathbb{N}$ counts how many pairs of mates both hit B_i and "span" a given position $t \in [1, l(B_i)]$. The *happy-* and *unhappy* clone coverage maps C_H and C_U count only the coverage by happy and unhappy mate-pairs, respectively. (Two mates f and g that hit the same contig B_i are *happy*, if (a) their layout is correct, that is, either $p_1(f) < p_2(f) \leq p_2(g) < p_1(g)$ or $p_1(g) < p_2(g) \leq p_2(f) < p_1(f)$, and, (b) their distance is approximately correct, that is, $||p_1(f) - p_1(g)| - m| \leq 3 \cdot \sigma$.)

In practice, we address the problems caused by polymorphisms and contig misassemblies by cutting contigs into smaller pieces, guided by the coverage maps. For a given contig B_i, we first identify potential cut sites using the core fragment coverage: any (maximal) interval of sites p in B_i with $\dot{F}(p) = 0$ is a possible cut site. Any (maximal) interval of sites p in B_i for which $C_H(p) < C_U(p)$ is considered a problematic region. We cut it at one or more potential cut sites by deleting the cut interval and replacing the original contig by a set of new smaller ones.

The human genome has a high abundance of repeats. We screen fragments for known repeats and take this information into account when computing fragment-contig hits [Venter et al. 2001]. However, not all repeats are known ahead of time and we use two different strategies to detect and remove fragment hits due to unscreened repeats.

For each fragment f that hits a contig B_i, we determine a window of 50 bp in B_i, in which the maximal fragment coverage is minimal. We call this number the *minimum hit rate* $r(f)$ for f. Any fragment f whose minimum hit rate $r(f)$ is larger than 8, say (the precise choice of this number depends on the level of fragment sequence coverage), is deemed to be contained in a repeat region and is excluded from further consideration, *unless* it has a mate g that happily hits the same contig and has a minimal low hit rate $r(g) \leq 8$.

Second, for the purposes of Section 7, we also take fragments into consideration that overlap off the end of a contig. Here, again, we apply a threshold of 8, say: if the number of fragments that overlap off one end of a contig is higher than the

threshold, then we only keep those fragments that have a mate that happily confirms their positioning.

5. *The Path-Merging Algorithm*

In this section, we describe an efficient heuristic algorithm for solving the Contig Scaffolding Problem.

We introduce some further notation. For purposes of the algorithm, every edge e of a contig-mate-pair graph is either *selected*, or not. Throughout the execution of the algorithm, we maintain the invariant that every node is adjacent to at most two selected edges and we refer to the components of the graph induced by the selected edges as *selected paths*. The ordering of contigs induced by such a path is called a *scaffolding* of the contigs. As we will see, the algorithm can introduce new edges to the contig-mate-pair graph, which are called *inferred* edges.

Any alternating chain of selected edges $C = (b_1, m_1, b_2, \ldots, b_k)$, (where b_i is a contig edge, and m_i is a mate-edge, for every i), with nodes $(v_{11}, v_{12}, v_{21}, v_{22}, \ldots, v_{k1}, v_{k2})$ (in the same order) inductively defines an ordering ϕ_C of the associated contigs: $\phi_C(v_{11}) := 0$, $\phi_C(v_{j2}) := \phi_C(v_{j1}) + l(b_j)$ and $\phi_C(v_{(j+1)1}) := \phi_C(v_{j2}) + l(m_j)$. We define $H(C)$ as the sum of weights of all mate-edges e whose nodes are both incident to C and who are happy with respect to ϕ. Similarly, we define $U(C)$ as the sum of weights of all mate-edges e whose nodes are both incident to C and who are unhappy with respect to ϕ.

Algorithm 5.1 (*Greedy Path-Merging Algorithm*). Given a contig-mate-pair graph G. The output of this algorithm is a node-disjoint covering of G by selected paths, each one defining an ordering of the contigs whose edges it covers.

begin
Select all contig edges.
for each *mate-edge e in descending order of weight:*
  if *e is not selected:*
    Let v, w denote the two nodes connected by e
    Let P_1 be the selected path incident to v
    Let P_2 be the selected path incident to w
    if *$P_1 \neq P_2$ and we can merge P_1 and P_2 (guided by e) to obtain P:*
      if $H(P) - (H(P_1) + H(P_2)) \geq U(P) - (U(P_1) + U(P_2))$:
        Replace *P_1 and P_2 by P*
end.

Initially, all contig edges are selected. As a consequence of this, both ends of an unselected mate-edge e are each adjacent to precisely one selected path, say P_1 and P_2. If $P_1 = P_2$, then e is a chord of P_1 and no merging is necessary. If $P_1 \neq P_2$, then, guided by e, we attempt to merge the two paths to obtain a new selected path P containing precisely those contig edges that are present in P_1 and P_2. If such a merging is possible (see Algorithm 5.2), then we check whether we have gained as least as much weight for happy edges as for unhappy ones. If this is the case, we *replace* the edge selections associated with P_1 and P_2 by the edge selections associated with P. In other words, we first deselect all edges in P_1 and P_2, and then select all edges in P.

Algorithm 5.2 (*Merging Two Selected Paths*). Given two selected paths P_1 and P_2 and a guiding unselected mate-edge e_0 with nodes v_0 (incident to P_1)

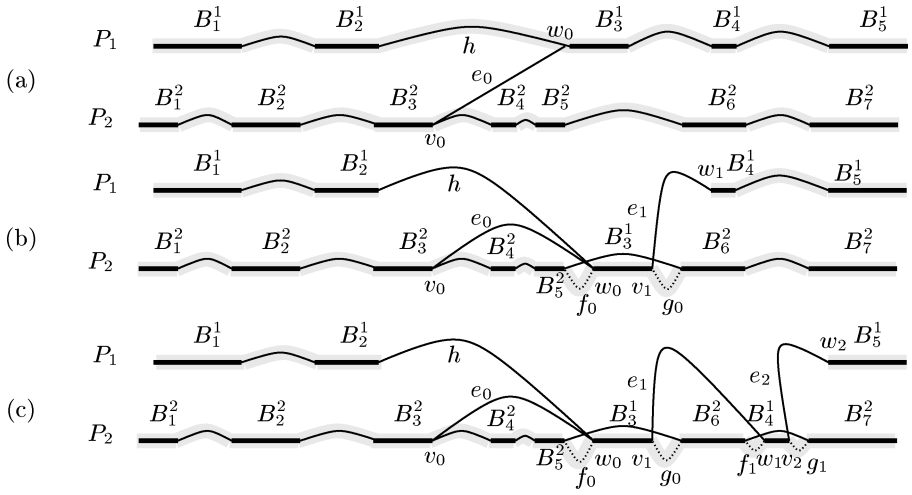

FIG. 3. The main step of merging P_1 into P_2, guided by e_0. Contig edges are depicted as heavy lines, mate (and previously inferred) edges as light lines and newly inferred edges as dotted lines. Currently selected edges are highlighted. The contig adjacent to w_0 is called *active* and the goal is to place it in P_2. (a) In this example, B_3^1 is the active contig and fits well between B_5^2 and B_6^2. (b) We introduce two new inferred edges, f_0 and g_0, that connect B_3^1 to its two new neighbors. Both are assigned a mean length, standard deviation and weight, based on the chain of edges that we implicitly used to infer the positioning of B_3^1. We then update the selection state as indicated. (c) This construction is repeated for edge e_1 with active contig B_4^1, and then for e_2, etc, until the end of either path is reached.

and w_0 (incident to P_2). The algorithm returns true, if it successfully produced a new selected path P containing all contig edges in P_1 and P_2, and false, if it fails.

Merging proceeds as indicated in Figure 3: starting with e_0, we "zipper" P_1 into P_2 to the right, until we reach the end of either path. Then, with the edge labeled h now playing the role of e_0, we "zipper" to the left. Merging is said to fail, if the positioning of the "active" contig B_i^1 implies that it must overlap with some contig in P_2 by a significant amount, but no such alignment (of sufficiently high quality) exists.

THEOREM 5.3. *The greedy-path merging algorithm is correct and runs in $O(mn + m^2)$ time, where m is the number of mate-pair edges and n is the number of contig edges.*

PROOF. The main loop considers each mate-edge at most once, hence it terminates, after $O(m)$ steps. The merge step considers each contig in path P_1 once, and hence also terminates, after $O(n)$ steps. For a given path P, we can compute the total weight of happy mate-edges in $O(m)$ steps. □

6. *Additional Combinatorial Constraints*

Consider the merging step described in Algorithm 5.2 and Figure 3. If the standard deviation of the guiding mate-edge e_i is high, and if the contigs in P_2 are small, then it may not be clear between which two contigs in P_2 the active contig B_i^1 should be placed.

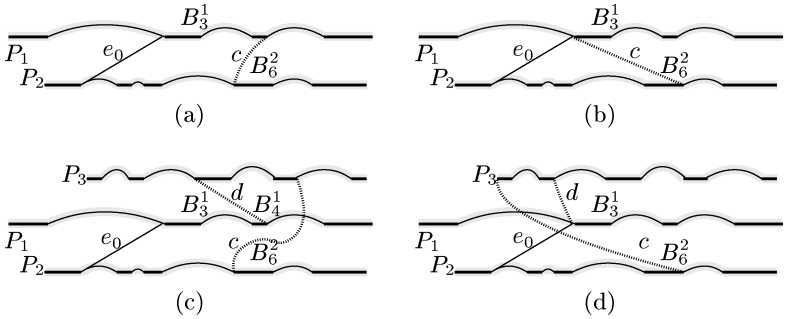

FIG. 4. Combinatorial constraints for placement of active contig B_3^1. (a) Mate-edge c indicates that B_3^1 should be placed before B_6^2. (b) Mate-edge c indicates that B_3^1 should be placed after B_6^2. (c) Making use of a third path P_3, mate-edges c and d indicate that B_3^1 should be placed before B_6^2. (c) Making use of a third path P_3, mate-edges c and d indicate that B_3^1 should be placed after B_6^2.

In this situation, a number of combinatorial constraints can be taken into account to determine the placement of B_i^1. In Figure 4(a), the existence of mate-edge c suggests that contig B_3^1 should be placed before contig B_6^2, whereas in Figure 4(b), contig B_3^1 should rather follow B_6^2. More generally, using ordering information of some third path P_3 in the contig-mate-pair graph, in Figure 4(c) the existence of mate-edges c and d again suggest that contig B_3^1 should be placed before contig B_6^2, whereas Figure 4(d) indicates that contig B_3^1 should follow B_6^2.

The more general constraints in Figure 4(b) and (c) are straightforward to implement: to quickly determine whether mate-edges c and d connect to the same third path P_3, we maintain a numbering of the paths and each node is labeled by the number of the path that it is contained in. Additionally, there are obvious parity conditions that can be checked by maintaining a two-coloring of each path.

7. *Contig Extension and Gap Filling*

Fragment mate-pairs can be used to extend contigs and in many cases, to completely close the gaps between them.

We generalize the notation of a contig-mate-pair graph to that of a *contig-mate-pair-fragment* graph, by introducing two new types of edges: A *fragment edge* represents a fragment f in precisely the same way that a contig edges represents a contig. An *overlap edge* e is used to indicate that a fragment overlaps off the end of a contig. It is assigned a negative length $l(e)$, namely minus the number of base pairs by which the fragment and contig overlap.

Given a BAC $\mathcal{B} = \{B_1, B_2, \ldots, B_n\}$. We first construct the corresponding contig graph and then extend it to a contig-mate-pair-fragment graph as follows: Let f be a fragment that (uniquely) hits some contig B_i in $\mathcal{B}$. If f has a mate g, and if g does not hit any contig in $\mathcal{B}$, then we introduce a fragment edge that represents g and connect it to B_i using an appropriate mate-edge. If f is a mate that is not embedded in any contig in $\mathcal{B}$, but f overlaps off the end of some contig B_i, then we introduce a fragment edge labeled f and connect it to B_i using an appropriate overlap edge.

Algorithm 7.1 (*Contig Extension and Gap Filling*). Given a contig-mate-pair-fragment graph G and assume that we have a selection of edges produced by the path-merging algorithm. Consider a selected edge e connecting two contig (edges)

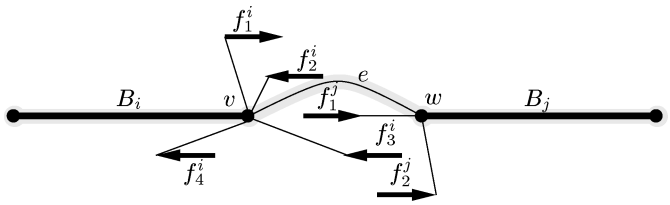

FIG. 5. For any selected edge e between two contigs B_i and B_j, we consider all fragment edges that are connected to either v or w, and order them from left to right, based on the lengths of their connectors. We then apply a pairwise alignment algorithm to confirm and refine the placements moving from v to the right, and, if we are not able to reach w, then we do this also from w to the left.

TABLE III. FOR 20960 PHASE-1/2 BACS, WE REPORT STATISTICS FOR THE NUMBER OF INPUT CONTIGS AND FRAGMENTS (AFTER REPEAT SCREENING) PER BAC. WE THEN LIST STATISTICS FOR THE NUMBER OF OUTPUT CONTIGS AND SCAFFOLDS (I.E. COLLECTIONS OF CONTIGS OF KNOWN ORDERING AND ORIENTATION) NECESSARY TO COVER 90% OF THE TOTAL SEQUENCE OF THE BAC, AS PRODUCED BY OUR ALGORITHM. IN THE FINAL COLUMN, WE REPORT THEIR MEAN SIZES

	Number of pieces					mean
	mean	std. dev.	median	maximum	sum	size
Input contigs	19.8	14.7	17	203	415687	8102
Input fragments	1687.9	844.9	1627	21240	35378100	543
Output contigs	8.9	8.5	7	144	186138	17380.5
Output scaffolds	2.1	3.6	1	109	44134	73303

B_i and B_j, see Figure 5. First, if $l(e) - 3\sigma(e) < 0$, then it is possible that B_i and B_j may overlap, and if an appropriate overlap alignment is found, then B_i and B_j can be replaced by a new contig B' that is obtained by overlap-aligning B_i and B_j. If this is not the case, we proceed as follows: Collect all fragment edges that are adjacent to a node of e by an overlap or mate-edge, and order these fragments from left to right, based on the lengths of the connecting edges. Then greedily extend the left-hand contig B_i to the right by iteratively attempting to overlap the next fragment off the end of the current extension. If the other contig B_j is reached, replace B_i and B_j by a new contig B' obtained as the consensus sequence of a multi-alignment of B_i, B_j and the intermediate fragments. If not, then replace B_i by its extension, reorder the unused fragments from right to left and then apply the same extension step from the right-hand contig B_j to the left.

8. *Experimental Results*

The compartmentalized assembler based on the algorithms presented in this paper was run on BACs obtained from Genbank on September 1, 2000, which is a more recent data set than reported in [Huson et al. 2001b].

BAC files retrieved from Genbank were screened for contaminants, vector etc., and problematic stretches of sequence was removed from contigs, which led to an increase in number, and decrease in size, of contigs, indeed, a number of phase-3 BACs were broken into smaller contigs. In Table III, we summarize the statistics for 20960 phase-1/2 BACs. Due to the presence of many very small contigs that are difficult to order as they lack sufficient fragment hits, our output statistics are based on the number of scaffolds required to capture at least 90% of the base-pairs

TABLE IV. BASED ON A SET OF 100 PHASE-1/2 BACS, WE REPORT STATISTICS FOR THE NUMBER OF THE HAPPY AND UNHAPPY MATE-PAIRS IN THE INPUT (CONTIGS) AND THE OUTPUT (COMPUTED SCAFFOLDS) OF OUR IMPLEMENTATION OF THE GREEDY-PATH MERGING ALGORITHM

	mean	std. dev.	minimum	median	maximum	sum
Input happy mate-pairs	293.3	88.9	118	293.3	682	29332
Input unhappy mate-pairs	2.8	2.2	0	2	10	279
Output happy mate-pairs	487.6	88.3	224	497	732	48760
Output unhappy mate-pairs	11.1	6	0	11	37	1112

available in the input BAC. Note that the reported sum of fragments is very high. This is because the same fragment can hit multiple BACs in the presence of an undetected repeat, or when different BACs cover the same source region.

In a high proportion of cases, our algorithm is able to place 90% or more of the sequence of a phase-1/2 BAC into one scaffold, thus substantially improving the level of assembly of these BACs. Moreover, in many cases we were able to verify the correctness of the assemblies: For any two BACs that come from overlapping source sequence, one can compute the pairwise alignment of the scaffolded sequence and determine whether it displays the signature of an overlap-alignment, or not. (It is easy to determine whether two BACs come from the same region of the genome by counting the number of fragments that hit both BACs simultaneously, but only hit a low number of BACs in general). In this way, we were able to determine that the algorithm makes very few mistakes on phase-1/2 BACs. However, it does not do particularly well on phase-0 BACs, simply because the typical length of a contig edge in this case is in the same range as the standard deviation of the mate-pair edges, and the algorithm sometimes falsely interleaves chains of small contigs.

In Table IV, we report the increase of happy and unhappy *mate-pairs* obtained by the path-merging algorithm applied to 100 randomly chosen phase-1/2 BACs. On average, we increased the difference of the total number of happy mate-pairs minus the total number of unhappy mate-pairs by 64%. (Note that the input contigs where preprocessed as described in Section 4, and so the mean number of unhappy mate-pairs per input contig is lower than what one would observe for contigs taken straight out of GenBank.) Given these BACs and the fragments that hit them, it took approximately 25 minutes to run our algorithm on all 100 assemblies on a 266 MHZ Pentium II laptop under Linux.

9. *Conclusions and Acknowledgments*

We have presented a simple and elegant algorithm for assembling a mixture of contigs and mate-pairs. It has been applied to 23000 unfinished human BACs obtained from GenBank and our experience suggests that BACs sequenced to 3-4.5× (phase 1–2) can be successfully ordered and oriented with this type of approach, whereas phase-0 quality data cannot.

The presented algorithm has other applications, for example to order and extend gene fragments. Also, it has been used to order and orient all contigs belonging to a given human chromosome, producing whole chromosome assemblies. Another application is syntenic assembly: For example, given a collection of mouse contigs and matches between contigs induced by syntenic matches to the human genome, our algorithm can be used to produce a “humanized”-mouse assembly. Finally, the

algorithms described in this paper are a useful model for the scaffolding stage of existing whole-genome assemblers.

We would like to thank Granger Sutton for many helpful discussions.

REFERENCES

BENSON, D. A., KARSCH-MIZRACHI, I., LIPMAN, D. J., OSTELL, J., RAPP, B. A., AND WHEELER, D. L. 2000. Genbank. *Nuc. Acids Res. 28*, 1, 15–8.

BEVINGTON, P. R. 1969. *Data Reduction and Error Analysis for the Physical Sciences*. McGraw-Hill, Inc., New York.

GAREY, M. R., AND JOHNSON, D. S. 1979. *Computers and Intractability, A Guide to the Theory of NP-completeness*. Bell Telephone Laboratories, Inc.

GREEN, P. 1994. Documentation for Phrap. `http://bozeman.mbt.washington.edu/phrap.docs/phrap.html`.

HUSON, D. H., REINERT, K., KRAVITZ, S. A., REMINGTON, K. A., DELCHER, A. L., DEW, I. M., FLANIGAN, M., HALPERN, A. L., LAI, Z., MOBARRY, C. M., SUTTON, G. G., AND MYERS, E. W. 2001. Design of a compartmentalized shotgun assembler for the human genome. *Bioinformatics* (*Proceedings of ISMB 2001*) *17*, 132–139.

HUSON, D. H., REINERT, K., AND MYERS, E. W. 2001b. The greedy path-merging algorithm for sequence assembly. In *Proceedings of the 5th Annual International Conference on Computational Molecular Biology* (*RECOMB-01*), pp. 157–163.

INTERNATIONAL HUMAN GENOME SEQUENCING CONSORTIUM. 2001. Initial sequencing and analysis of the human genome. *Nature 409*, 6822, 860–921.

LANDER, E. S., AND WATERMAN, M. S. 1988. Genomic mapping by fingerprinting random clones: A mathematical analysis. *Genomics 2*, 231–239.

MYERS, E. W., SUTTON, G. G., DELCHER, A. L., DEW, I. M., FASULO, D. P., FLANIGAN, M. J., KRAVITZ, S. A., MOBARRY, C. M., REINERT, K. H. J., REMINGTON, K. A., ANSON, E. L., BOLANOS, R. A., CHOU, H.-H., JORDAN, C. M., HALPERN, A. L., LONARDI, S., BEASLEY, E. M., BRANDON, R. C., CHEN, L., DUNN, P. J., LAI, Z., LIANG, Y., NUSSKERN, D. R., ZHAN, M., ZHANG, Q., ZHENG, X., RUBIN, G. M., ADAMS, M. D., AND VENTER, J. C. 2000. A whole-genome assembly of Drosophila. *Science 287*, 2196–2204.

SANGER, F., COULSON, A. R., HONG, G. F., HILL, D. F., AND PETERSEN, G. B. 1992. Nucleotide sequence of bacteriophage λ DNA. *J. Mol. Bio. 162*, 4, 729–773.

SANGER, F., NICKLEN, S., AND COULSON, A. R. 1977. DNA sequencing with chain-terminating inhibitors. *Proc. Nat. Acad. Sci. 74*, 12, 5463–5467.

U. S. DEPARTMENT OF ENERGY, OFFICE OF ENERGY RESEARCH, AND OFFICE OF BIOLOGICAL AND ENVIRONMENTAL RESEARCH. 1997. Human genome program report. `http://www.ornl.gov/hgmis/publicat/97pr/`.

VENTER, J. C., ADAMS, M. D., MYERS, E. W., ET AL. 2001. The sequence of the human genome. *Science 291*, 1145–1434.

WEBBER, J. L., AND MYERS, E. W. 1997. Human whole-genome shotgun sequencing. *Gen. Res. 7*, 5, 401–409.

RECEIVED JANUARY 2002; REVISED JUNE 2002; ACCEPTED AUGUST 2002