



Fragment Assembly with Double-Barreled Data

Pavel A. Pevzner¹ and Haixu Tang²

¹Department of Computer Science and Engineering, University of California at San Diego, La Jolla, CA 92093, and ²Department of Mathematics, University of Southern California, Los Angeles, CA 90089,

ABSTRACT

For the last twenty years fragment assembly was dominated by the “overlap - layout - consensus” algorithms that are used in all currently available assembly tools. However, the limits of these algorithms are being tested in the era of genomic sequencing and it is not clear whether they are the best choice for large-scale assemblies. Although the “overlap - layout - consensus” approach proved to be useful in assembling clones, it faces difficulties in genomic assemblies: the existing algorithms make assembly errors even in bacterial genomes. We abandoned the “overlap - layout - consensus” approach in favor of a new Eulerian Superpath approach that outperforms the existing algorithms for genomic fragment assembly (Pevzner et al., 2001). In this paper we describe our new EULER-DB algorithm that, similarly to the Celera assembler takes advantage of clone-end sequencing by using the double-barreled data. However, in contrast to the Celera assembler, EULER-DB does not mask repeats but uses them instead as a powerful tool for contig ordering. We also describe a new approach for the Copy Number Problem: “How many times a given repeat is present in the genome?”. For long nearly-perfect repeats this question is notoriously difficult and some copies of such repeats may be “lost” in genomic assemblies. We describe our EULER-CN algorithm for the Copy Number Problem that proved to be successful in difficult sequencing projects.

Contact: ppevzner@ucsd.edu

INTRODUCTION

Children like puzzles and they usually assemble them by trying all possible pairs of pieces and putting together pieces that match well. Biologists assemble genomes in a surprisingly similar way, the major difference being that the number of pieces is much larger. For the last twenty years fragment assembly in DNA sequencing mainly followed the “overlap - layout - consensus” paradigm, that is used in all currently available software tools for fragment assembly (Green, 1994; Bonfield et al., 1995; Sutton et al., 1995; Huang and Madan, 1999). Trying all possible pairs of pieces corresponds to the overlap step while putting the pieces together corresponds to the layout step of the fragment assembly algorithms. Our

new EULER algorithm is very different from this natural approach - we never even try to match the pairs of fragments together and we don't have the overlap step at all. Instead we do a very counter-intuitive (some would say childish) thing: we cut the existing pieces of a puzzle into even smaller pieces of regular shape thus increasing the number of pieces. Although it indeed looks childish and irresponsible, we do it on purpose rather than for the fun of it. This operation transports the puzzle assembly from the hostile world of a difficult Layout Problem into the world of the Eulerian Path Problem with polynomial algorithms for puzzle assembly (in the context of DNA sequencing).

Although the classical approach culminated in some excellent fragment assembly tools (Phrap, CAP3, TIGR, and Celera assemblers are among them), critical analysis of the “overlap - layout - consensus” paradigm reveals some weak points. The major difficulty is that there is still no efficient algorithm for the solution of the layout problem. The layout problem was formulated as the *Shortest Superstring Problem* in the early computational studies of DNA sequencing:

Shortest Superstring Problem (SSP). Given a set of strings $S = \{s_1, \dots, s_n\}$, find the shortest string s such that each s_i appears as a substring of s .

Since the Shortest Superstring Problem is known to be NP-hard, a number of heuristics have been proposed. The early DNA sequencing algorithms (Peltola et al., 1984) used a simple *greedy* strategy: repeatedly merge a pair of strings with maximum overlap until only one string remains, not very different from the way children assemble puzzles. This strategy, however, failed even for relatively small viral fragment assembly projects in mid80s. In fact, till early 90s, it was commonly believed that cosmids represent the limit of the shotgun approach. These difficulties led to more sophisticated fragment assembly algorithms (Kececicoglu and Myers, 1995; Myers, 1995) that adequately addressed important practical issues (repeat collapsing, sequencing errors, double-stranded DNA, etc.), but did not make the fragment assembly any simpler - from the algorithmic perspective it still has a flavor of SSP with no efficient algorithm in sight. However, these algorithmic development pushed the limits of solvable fragment assembly instances and allowed biologists

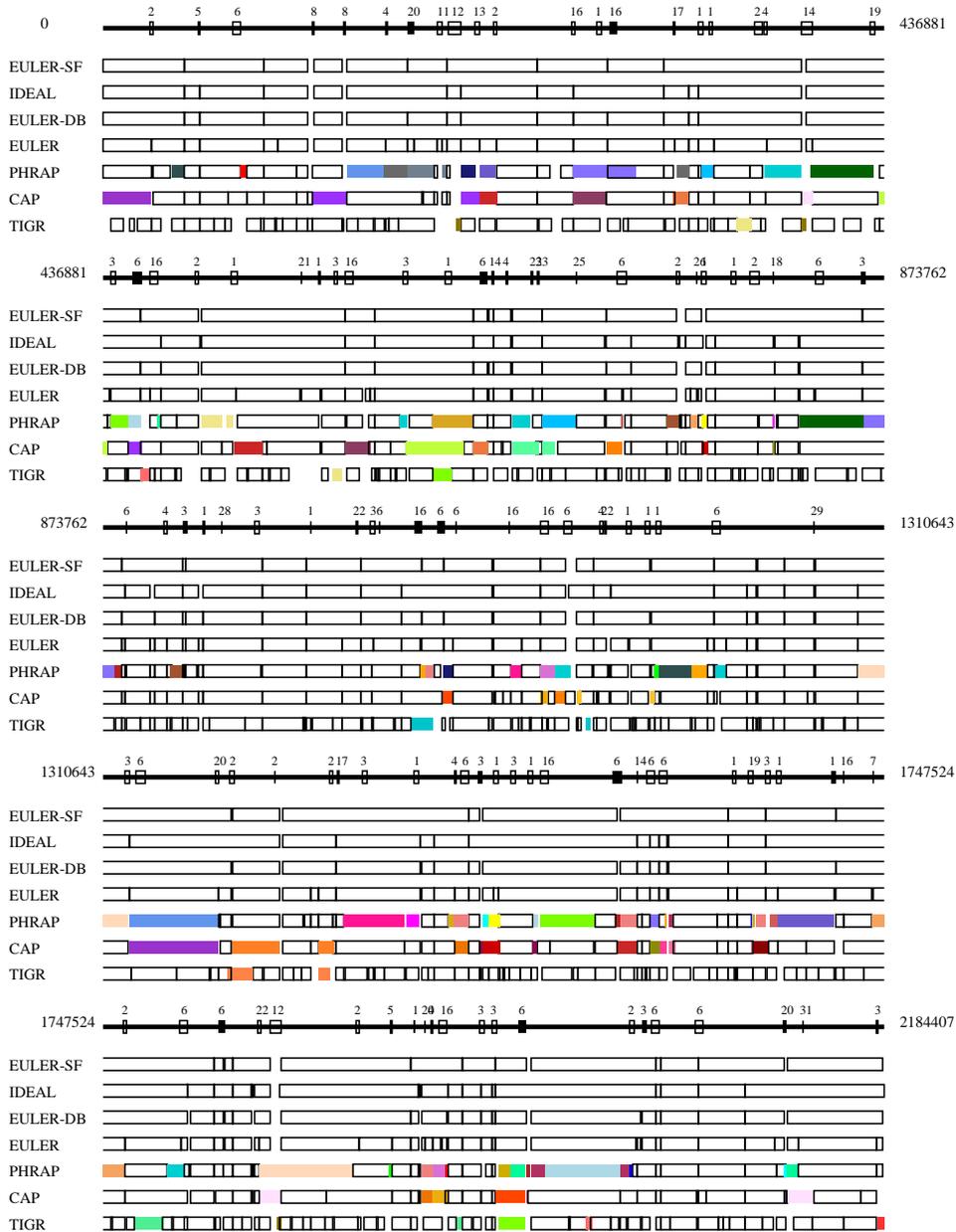


Fig. 1. Comparative analysis of EULER, Phrap, CAP and TIGR assemblers (*N. meningitidis* sequencing project). Every box corresponds to a contig in NM assembly produced by these programs with colored boxes corresponding to assembly errors. Boxes in the IDEAL assembly correspond to islands in the read coverage. Boxes of the same color show misassembled contigs, for example two identically colored boxes in different places show the positions of contigs that were incorrectly assembled into a single contig. In some cases, a single colored box shows a contig that was assembled incorrectly (i.e., there was a rearrangement within this contig). The tangles and repeats with similarity higher than 95% are indicated by numbered boxes at the solid line showing the genome. To check the accuracy of the assembled contigs we fit each assembled contig into the genomic sequence. Inability to fit a contig into the genomic sequence indicates that the contig is misassembled. For example, Phrap misassembles 17 contigs in the *N. meningitidis* sequencing project, each contig containing from two to four fragments from different parts of the genome.

to assemble BAC-sized clones.

However, any hope for development of efficient algorithms for large-scale SSP-style problems may disappear

after reading p.228 of Garey and Johnson, 1979. Biologists were the first to learn it shortly after the bacterial sequencing projects started. Table 1 and Fig. 1 illustrate

that Phrap, CAP3 and TIGR assemblers misassemble up to 19% of contigs in different bacterial sequencing projects. These bacterial genomes were assembled *despite* the fact that there is no error-free fragment assembler today. Biologists “pay” for these errors by the time-consuming finishing step that identifies missassembled contigs and corrects them by PCR experiments. Bioinformaticians are also aware of assembly errors as evidenced by finishing software that supports experiments correcting these errors (Gordon et al., 1998).

The difficulties with fragment assembly led to introduction of the double-barreled DNA sequencing (Roach et al., 1995; Weber and Myers, 1997) that uses additional experimental information for assembling large genomes in the framework of the same “overlap - layout - consensus” paradigm (Myers et al., 2000). The Double Barreled (DB) data were first employed in 1995 in fragment assembly of *H. influenzae* (Fleischmann and et al., 1995). DB data in this project mainly consisted of pairs of reads (mate-pairs) from the ends of 2 Kb or 16 Kb inserts randomly sampled from the genome. It provided a roughly twofold increase in the effective read length since the approximate distance between the reads from the mate-pair is known.

Although some fragment assemblers are able to deal with DB data (Myers et al., 2000; Kent and Haussler, 2001), most sequencing centers use heuristic procedures to utilize DB data. Such approaches are prone to errors and may fail for complex genomes. EULER-DB provides the possibility to analyze DB data in a new way that allows one to transform *mate-pairs* “read1 - GAP of length d - read2” (a roughly twofold increase in the effective read length) into *mate-reads* “read1 - DNA SEQUENCE of length d - read2” in most cases. Assuming that the insert length is 5Kb, it provides a tenfold increase in the effective length of reads and transforms the original fragment assembly problem with N reads of length 500 bp into a new fragment assembly problem with about $N/2$ reads of length 5000 bp. From some perspective, EULER-DB provides an algorithmic shortcut for still unsolved experimental problem of increasing the read length. Since EULER does not mask repeats, a significantly larger portion of mate-pairs can be transformed into mate-reads as compared to other DB assemblers.

Finding the number of copies of a repeat (multiplicity of a repeat) is another important and still unsolved problem in fragment assembly. The difficulty is that, in genomes with complicated repeat structure, it may be impossible to derive the multiplicities of repeats by analyzing only reads closely located to this repeat. In some cases, the information about the reads thousands nucleotides from a repeat may be the only clue to infer the multiplicity of this repeat. The existing programs largely ignore this problem and often output every repeat as a single contig without even trying to find its multiplicity. This is a serious

shortcoming since some repeats have high copy numbers. We describe a new approach to the Copy Number Problem that generated the correct multiplicities for all repeats in the studied bacterial genomes.

EULER: ASSEMBLING PUZZLES BY BREAKING THEM INTO SMALLER PIECES

How can one resolve the layout problem? Surprisingly enough, an unrelated area of DNA arrays provides a hint. *Sequencing by Hybridization (SBH)* is a 10-years old idea that never became practical but (indirectly) created the DNA array industry. Conceptually, SBH is similar to fragment assembly, the main difference being that the “reads” in SBH are much shorter l -tuples (i.e., strings of fixed length l).

Sequencing by Hybridization Problem. Given a set of l -tuples $S = \{s_1, \dots, s_n\}$, find the shortest string s such that each l -tuple s_i appears as a substring of s and every l -tuple from s appears as an l -tuple in S .

Although the SBH problem looks very similar to SSP, it is dramatically different from the perspective of computational complexity. The first naive attempts to solve the SBH problem (Drmanac et al., 1989; Lysov et al., 1988) followed the “overlap-layout-consensus” paradigm. However, the corresponding layout problem leads to the NP-complete *Hamiltonian Path Problem*. (Pevzner, 1989) proposed a different approach that reduces SBH to an easy-to-solve *Eulerian Path Problem* in the *de Bruijn* graph by abandoning the “overlap-layout-consensus” paradigm for SBH.

Since the Eulerian path approach transforms a once difficult layout problem into a simple one, a natural question is: “Could the Eulerian path approach be applied to fragment assembly?”. Idury and Waterman, 1995 answered this question by mimicking the fragment assembly problem as an SBH problem. They “cut” every read into a collection of l -tuples thus simulating a DNA chip experiment (instead of DNA sequencing experiment) with an unknown DNA fragment. Every read of length n is transformed into a collection of $n - l + 1$ l -tuples (breaking a puzzle into smaller pieces). At the first glance this transformation of every read into a collection of l -tuples is a very short-sighted procedure since information about the sequencing reads is lost. However, the loss of information is minimal for large l and is well paid for by the computational advantages of the Eulerian path approach in the resulting easy-to-analyze graph.

Unfortunately, the Idury-Waterman approach, while very promising, did not scale up well. The problem is that (i) the sequencing errors transform the *de Bruijn* graph into a tangle of erroneous edges and (ii) repeats pose serious challenges even in the case of error-free data since they make the *de Bruijn* graph even more tangled.

Table 1. Comparison of different software tools for fragment assembly. IDEAL is an imaginary assembler that outputs the collection of islands in clone coverage as contigs. In the IDEAL column the number in parenthesis shows the overall multiplicity of tangles. CJ, NM, and LL correspond to *Campylobacter jejuni*, *Neisseria meningitidis*, and *Lactococcus lactis* sequencing projects.

		IDEAL	EULER	Phrap	CAP3	TIGR assembler
CJ	# of contigs (# of misassembled contigs)	24(5)	29(0)	33 (2)	54 (3)	> 300 (> 10)
	coverage by contigs	99.5%	96.7%	94.0%	92.4%	90.0%
	coverage by misassembled contigs	-	0.0%	16.1%	13.6%	1.2%
NM	# of contigs (# of misassembled contigs)	79(126)	149 (0)	160 (17)	163 (14)	> 300(9)
	coverage by the contigs	99.8%	99.1%	98.6%	97.2%	87.4%
	coverage by misassembled contigs	-	0.0%	10.5%	9.2%	1.3%
LL	# of contigs (# of misassembled contigs)	6(61)	58 (0)	62 (10)	85 (8)	245 (2)
	coverage by the contigs	99.9%	99.5%	97.6%	97.0%	90.4%
	coverage by misassembled contigs	-	0.0%	19.0%	11.4%	0.7%

Pevzner et al., 2001 addressed both these shortcomings of the Idury-Waterman approach and developed a practical fragment assembly software tool EULER. This software produced error-free assemblies for all bacterial sequencing projects that we studied.

EULERIAN SUPERPATH APPROACH

The idea of the Eulerian path approach to SBH is to construct a graph whose edges correspond to l -tuples and to find a path visiting every edge of this graph exactly once.

For a set of reads $S = \{s_1, \dots, s_n\}$ define S_l as the collection of all l -tuples from strings from S . Given a set of reads $S = \{s_1, \dots, s_n\}$, define the de Bruijn graph $G(S_l)$ with vertex set S_{l-1} (the set of all $(l-1)$ -tuples from S) as follows. An $(l-1)$ -tuple $v \in S_{l-1}$ is joined by a directed edge with an $(l-1)$ -tuple $w \in S_{l-1}$, if S_l contains an l -tuple for which the first $l-1$ nucleotides coincide with v and the last $l-1$ nucleotides coincide with w . Each l -tuple from S_l corresponds to an edge in G .

If S contains the only sequence s_1 , then this sequence corresponds to a path visiting each *edge* of G exactly once, an *Eulerian path* (Pevzner, 2000). Finding Eulerian paths is a well-known problem that can be efficiently solved.

Sequencing errors and repeats make implementation of this SBH-style approach to fragment assembly difficult. Taking into account the information about what l -tuples belong to the same reads (that was lost after the construction of the de Bruijn graph) helps us to untangle this graph. Every sequencing read corresponds to a path in the de Bruijn graph called a *read-path*. An attempt to take into account the information about the sequencing reads leads to the problem of finding an Eulerian path that is consistent with all read-paths, an Eulerian Superpath Problem.

A path $v_1 \dots v_n$ in the de Bruijn graph is called a *repeat* if $\text{indegree}(v_1) > 1$, $\text{outdegree}(v_n) > 1$, and $\text{outdegree}(v_i) = 1$ for $1 \leq i \leq n-1$ (Fig. 2). Edges

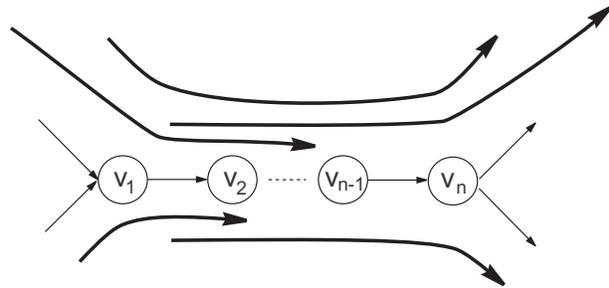


Fig. 2. A repeat $v_1 \dots v_n$ and a system of paths overlapping with this repeat. The uppermost path contains the repeat and defines the correct pairing between the corresponding entrance and exit. If this path were not present, the repeat $v_1 \dots v_n$ would become a tangle.

entering the vertex v_1 are called *entrances* into a repeat while edges leaving the vertex v_n are called *exits* from a repeat. An Eulerian path visits a repeat a few times and every such visit defines a pairing between an entrance and an exit. Repeats may create problems in fragment assembly since there are a few entrances in a repeat and a few exits from a repeat but it is not clear which exit is visited after which entrance in the Eulerian path. However, most repeats can be resolved by read-paths (i.e., paths in the de Bruijn graph that correspond to sequencing reads) *covering* these repeats. A read-path covers a repeat if it contains an entrance into this repeat and an exit from this repeat. Every covering read-path reveals some information about the correct pairings between entrances and exits. However, some parts of the de Bruijn graph are impossible to untangle due to long perfect repeats that are not covered by any read-paths. A repeat is called a *tangle* if there is no read-path containing this repeat (Fig. 2).

Eulerian Superpath Problem. Given an Eulerian graph

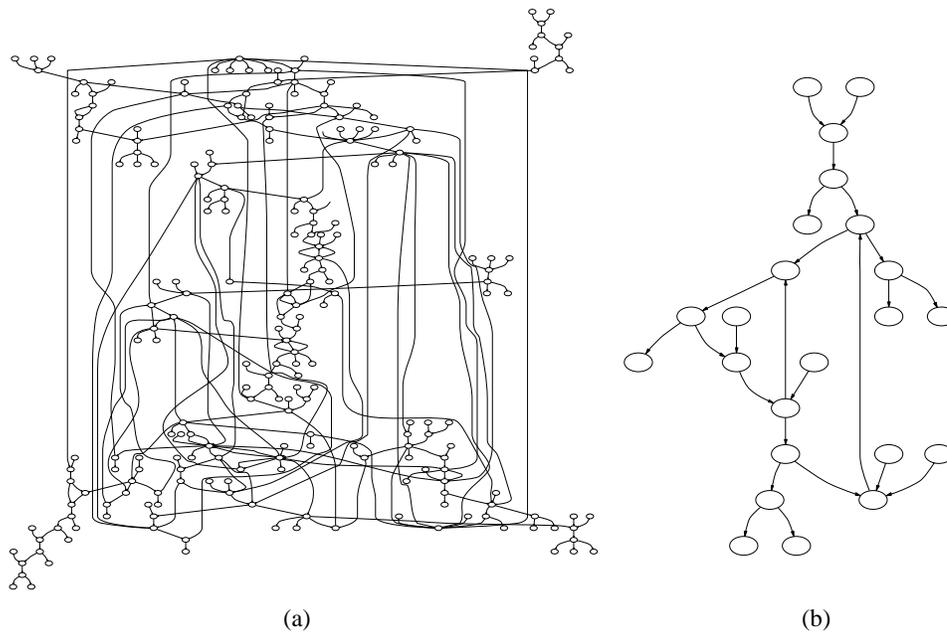


Fig. 3. The largest connected component of the de Bruijn graph for the NM sequencing project after EULER (a) and after EULER-DB (b).

and a collection of paths in this graph, find an Eulerian path in this graph that contains all these paths as subpaths.

To solve the Eulerian Superpath Problem we transform the graph G and the system of paths \mathcal{P} into a new graph G_1 with a new system of paths \mathcal{P}_1 . Such transformation is called *equivalent* if there exists a one-to-one correspondence between Eulerian superpaths in (G, \mathcal{P}) and (G_1, \mathcal{P}_1) . Our goal is to make a series of equivalent transformations $(G, \mathcal{P}) \rightarrow (G_1, \mathcal{P}_1) \rightarrow \dots \rightarrow (G_k, \mathcal{P}_k)$ that lead to a system of paths \mathcal{P}_k with every path being a single edge. Since all transformations on the way from (G, \mathcal{P}) to (G_k, \mathcal{P}_k) are equivalent, every solutions of the Eulerian Path Problem in (G_k, \mathcal{P}_k) provides a solution of the classical Eulerian Superpath Problem in (G, \mathcal{P}) . Pevzner et al., 2001 described the equivalent transformations that led to the efficient solution of the Eulerian Superpath problem for all bacterial genomes we studied.

EULER-DB ALGORITHM

Figure 3 illustrates the complexity of the repeat structure in the *N meningitidis* genome by showing the largest component of the de Bruijn graph produced by EULER. 126 long perfect repeats in this genome tangle the de Bruijn graph and make it difficult to analyze (although it is the simplest representation of the repeat structure in the NM genome). EULER-DB untangles this graph using DB data (Fig. 3).

EULER-DB maps every read into some edge(s) of the de Bruijn graph (instead of vertices as in conventional fragment assembly approaches). Such mapping provides the same computational advantages as the Eulerian Path approach versus the Hamiltonian Path Approach. After this mapping, most mate-pairs of reads correspond to paths that connect the positions of these reads in the de Bruijn graph (provided the distance between these positions in the graph is approximately equal to the estimated distance between reads from the mate-pairs). EULER-DB views such paths as long artificial mate-reads and analyzes them with the same Eulerian Superpath algorithm that is used for the analysis of standard reads.

Mapping reads into the de Bruijn graph allows one to identify errors in the DB data. In most cases both reads r_1 and r_2 from the mate-pair (r_1, r_2) are mapped to the same graph component (81% percent of correct mate-pairs in the NM project). In this case one can find a path between these reads and compare the length $d(r_1, r_2)$ of this path with an estimated distance $l(r_1, r_2)$ between clone ends (we assume that $d(r_1, r_2) = \infty$ if such path does not exist). In most cases such path is unique and its length approximately matches the clone length ($d(r_1, r_2) \approx l(r_1, r_2)$). In this case we reconstruct the intermediate sequence between reads and transform the mate-pair into the mate-path. In the case the difference between $d(r_1, r_2)$ and $l(r_1, r_2)$ is beyond the acceptable variation in the clone length, it is most likely an error in

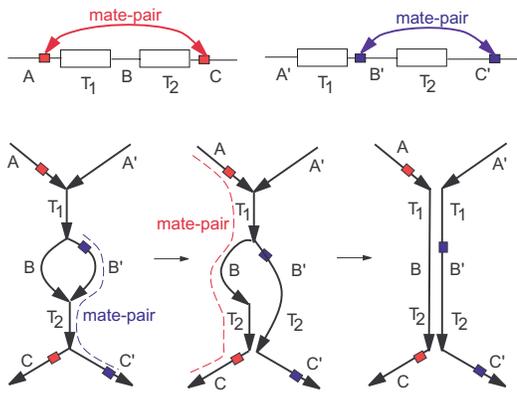


Fig. 4. Tangle resolution in the case when the clone length matches the length of more than one path between reads from the mate-pair.

the DB data.

In the case of multiple paths between r_1 and r_2 in the de Bruijn graph, there are three possibilities:

- The clone length $l(r_1, r_2)$ does not match the length of any path between r_1 and r_2 . Such situations often correspond to errors in the DB data (see below for the case $d(r_1, r_2) = \infty$).
- The clone length $l(r_1, r_2)$ matches the length of exactly one path between r_1 and r_2 . In this case we transform the mate-pair into the corresponding mate-read.
- The clone length $l(r_1, r_2)$ matches the length of more than one path between r_1 and r_2 . In this case there is no sufficient information (yet) to transform the mate-pair into the mate-read. The mate-pair is kept in the DB data in a hope that it will be resolved at the following iterations.

Although the latter situation does not happen frequently, it is important for resolution of very complicated repeats. Fig. 4 presents an example of two tangles T_1 and T_2 separated by non-repeated regions B and B' (red and blue boxes on the edges of the de Bruijn graph show two mate-pairs). The red mate-pair corresponds to multiple paths (AT_1BT_2C and $AT_1B'T_2C$) and cannot be transformed into a mate-read while the blue mate-pair corresponds to an unique path. A transformation of the blue mate-pair into a mate-path and the subsequent equivalent transformation lead to a modification of the de Bruijn graph (Fig. 4) with the red mate-pair now corresponding to the unique path. This path generates a mate-read that leads to resolution of both tangles.

EULER-DB VERSUS OTHER DB ASSEMBLERS

We emphasize important differences between our approach and other DB assemblers. The Celera assembler masks repeats, generates a large set of contigs, and pieces these contigs together using the DB data. There are two types of contigs subject to the DB step of the Celera's algorithm - G-contigs flanked by gaps in read coverage (on at least one side) and R-contigs flanked by repeats (on both sides). In a genome with many repeats, the number of R-contigs may significantly exceed the number of G-contigs. Even in a relatively simple case of bacterial genomes, the number of R-contigs maybe two-three times larger than the number of G-contigs. For example, in the NM sequencing project, Phrap generates 160 contigs and only half of them are G-contigs. In the LL sequencing project only tenth of all contigs are G-contigs

The Celera assembler provides an excellent solution for assembly of G-contigs but it does not distinguish between two types of contigs. After masking repeats, the Celera assembler generates a large number of contigs, only a small portion of them are G-contigs corresponding to an IDEAL assembly. Since the resulting contigs are short, it leads to a rather complicated DB step (Myers et al., 2000) that may cause disassembly for short contigs with limited DB data. In addition, note that valuable information about DB data from repeating regions is ignored.

In contrast to other DB assemblers, EULER-DB distinguishes between G-contigs and R-contigs. EULER-DB does not mask repeats but instead uses them (combined with DB data) as a powerful tool for resolving R-contigs. The NM sequencing project illustrates the advantages of using (rather than masking) repeats in fragment assembly. EULER-DB reduces the number of contigs from 149 to 117 if only mate-pairs from non-repeated regions are used. Use of mate-pairs that partially overlap the repeats further reduces the number of contigs to 91 (Fig. 1). EULER-DB typically resolves all tangles except tangles that are longer than the length of the insert. In the NM project (with insert length up to 1800) EULER left only 5 unresolved tangles of length 3610, 3215, 2741, 2503, and 1831. After completing EULER-DB, we build *scaffolds* using DB data as "bridges" between different contigs (EULER-SF). EULER-SF combines the 91 contigs into 60 scaffolds thus closing most gaps that are shorter than the insert length and further simplifying the finishing step (Fig. 1).

Figure 5 illustrates a potential problem with the "overlap-layout-consensus" approach followed by the DB step. Consider a simple case of two islands ATB and $A'TB'$ (each containing a copy of a long perfect repeat T) with four mate-pairs (two of them are false). The "overlap-layout consensus" algorithm may produce 4 separate contigs AT , B , $A'T$, and B' with two correct and two

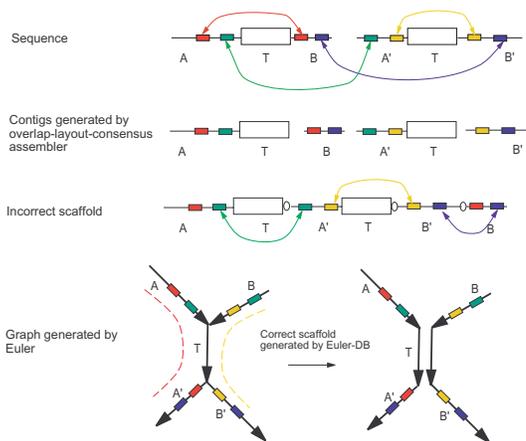


Fig. 5. Filtering out erroneous mate-pairs in EULER-DB.

false (green and blue) mate-pairs. The following DB step will have a hard time deciding what is the correct scaffold of these four contigs and filtering out the incorrect mate-pairs. It will most likely end up in an incorrect scaffold shown in the figure since it “explains” three mate-pairs (the correct solution “explains” only two mate-pairs). In contrast, EULER outputs a connected de Bruijn graph rather than a collection of 4 separate contigs. This graph allows one to filter out incorrect green and blue mate-pairs since there is no path between them in the graph. As a result, EULER easily resolves the tangle T in this case.

SCAFFOLDING PROBLEM

After completing EULER-DB, we build *scaffolds* using DB data as “bridges” between different contigs (EULER-SF algorithm). Myers et al., 2000 described an excellent solution of the scaffolding problem. Recently, Kent and Haussler, 2001 described a different greedy scaffolding approach that resulted in a successful draft assembly of the human genome. EULER-SF uses a similar strategy but it scaffolds a smaller set of longer contigs (mainly G-contigs). Since these contigs are long and reliable, the algorithmic aspects of EULER-SF become almost trivial as compared to other approaches. The only distinctive feature of EULER-SF is its ability to use repeats and mate-pairs from repeated regions to form larger scaffolds.

We first describe EULER-SF for a simple (and non-practical) case of error-free DB data. A vertex v is called a *source* if $\text{indegree}(v) = 0$, and a *sink* if $\text{outdegree}(v) = 0$. An edge (v, w) in a graph G is called an *initial (terminal)* edge if it is the only edge incident to a source (sink) in the graph. EULER-SF adds some edges between sinks and sources of the de Bruijn graph in an

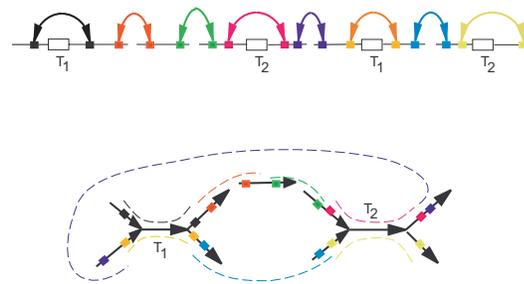


Fig. 6. Using the de Bruijn graph for scaffolding. The mate-pairs suggest a set of additional edges that resolve the tangles and connect all islands into a single scaffold. Four mate-pairs in this graph are processed with EULER-DB while four others “connect” different components and require EULER-SF to be processed.

attempt to “explain” as many mate-pairs as possible.

Fig. 6 illustrates a small sequencing project resulted in 5 islands with 2 tangles of multiplicity 2. The de Bruijn graph for this project consists of 3 connected components with 11 edges. Every mate-pair with $d(r_1, r_2) = \infty$ “suggests” an edge connecting a sink with a source in the de Bruijn graph. An Eulerian path in the resulting graph defines the correct scaffold (Fig. 6).

In practice this approach needs to be taken with caution since chimeric clones and errors in DB data “suggest” wrong edges in the de Bruijn graph. Below we describe the Scaffolding Problem that models DB data with errors.

A vertex v in a graph is called *balanced* if $\text{indegree}(v) = \text{outdegree}(v)$. A graph is *semi-balanced* if (i) all its vertices except sources and sinks are balanced, and (ii) all its connected component contain some sources and sinks. A semi-balanced graph can be transformed into an Eulerian graph (Fig. 6) by adding a number of edges connecting some sinks with some sources (we assume that both edges of G and newly added edges have associated lengths). We search for a transformation of the de Bruijn graph into an Eulerian graph that “explains” as many mate-pairs as possible.

A mate-pair corresponds to a pair of *positions* in the graph G (these positions correspond to the start/end of the reads from the mate-pair). A mate-pair (r_1, r_2) in a graph G is *feasible* if the length of at least one path between these positions in G matches $l(r_1, r_2)$. Addition of edges in G makes previously infeasible mate-pairs feasible and motivates the following:

Scaffolding Problem. Given a semi-balanced graph G with a set of mate-pairs, transform it into an Eulerian graph with maximum number of feasible mate-pairs.

Kent and Haussler, 2001 described a greedy approach to the Scaffolding Problem in a simple case when the graph

G is a collection of disconnected edges corresponding to contigs. Their approach adequately solves the scaffolding problem if the original contigs are long and reliable. Since GigAssembler has no way to break the misassembled BAC-contigs, it is important to have reliable BAC-assemblies as an input for GigAssembler.

Let (r_1, r_2) be a mate-pair with read r_1 mapped to a terminal edge and read r_2 mapped to an initial edge in the de Bruijn graph. Such mate-pair suggests a new *gap* edge e in the de Bruijn graph that connects the corresponding sink and source. Denote $\overline{r_1}$ as an extension of read r_1 till the end of the terminal edge it belongs to. Similarly, define $\overline{r_2}$ as an extension of r_2 from the beginning of the initial edge till the end of r_2 . EULER-SF substitutes the mate-pair (r_1, r_2) by a new *extended* mate-pair $(\overline{r_1}, \overline{r_2})$ if some distance constraints are satisfied. Define $b(r_1, r_2) = |\overline{r_1}| + |\overline{r_2}|$ as a lower bound for the distance between reads r_1 and r_2 in the final sequence. If $b(r_1, r_2) \leq l(r_1, r_2)$ then the mate-pair (r_1, r_2) *supports* the gap edge e with the length equal to the estimated *gap length* $l(r_1, r_2) - b(r_1, r_2)$. In practice we allow gap estimates to take small negative values to account for variance in the estimate of clone length. As a result, the support condition changes into $l(r_1, r_2) - d(r_1, r_2) \geq -\Delta$. EULER-SF transforms mate-pairs into extended mate-pairs and adds gap edges into the de Bruijn graph to form scaffolds.

In practice, EULER-SF uses a more flexible notion of support to generate gap edges. First, the reads r_1 and r_2 do not necessarily have to map into terminal and initial edges. In this case, a mate-pair (r_1, r_2) may imply a few fictitious edges, each with its own estimate of gap length. In this case, the mate-pair (r_1, r_2) *supports* an edge e if e is the only such edge with the gap estimate that is within acceptable limit among these gap edges ($l(r_1, r_2) - d(r_1, r_2) \geq -\Delta$). In practice, we filter the list of gap edges by deleting “competing” gap edges, i.e. the multiple gap edges with the same initial (terminal) vertices. To reduce the influence of errors in DB data we order the gap edges according to the number of mate-pairs that support them. Adding the gap edges in the graph in this order connected different parts of the graph and produced reliable scaffolds in our experiments with bacterial sequencing data.

COMPUTING REPEAT COPY NUMBERS

The Eulerian path approach reduces the Copy Number Problem to finding multiplicities of edges in the de Bruijn graph. Fig. 7, left presents the de Bruijn graph corresponding to a contig from the *Mycoplasma genitalium* genome (simulated reads) with three short tangles. However, it is not obvious which edges in this graph correspond to tangles and what are their multiplicities. Although the edge $(2, 3)$ is used four times in the Eulerian path in this

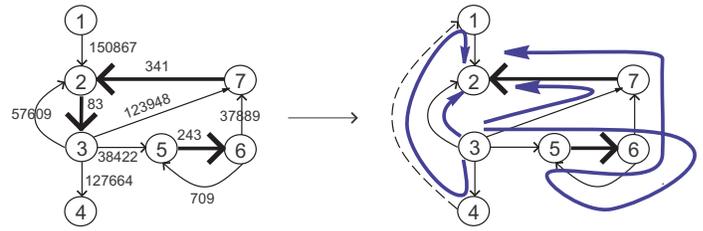


Fig. 7. Minimum flow (shown by four paths) reconstructs the multiplicity of 83 bp tangle in the *Mycoplasma genitalium* genome (simulated reads). The tangles $(2,3)$, $(5,6)$, and $(7,2)$ have multiplicities 4, 2, and 2, correspondingly.

graph (its multiplicity is 4), our construction of the de Bruijn graph generates a graph with unit multiplicities and does not reveal this information.

Our approach to the Copy Number Problem is very different from the previous statistical approaches that attempted to solve this problem by considering excessive read coverage in repeated regions. The existing implementation of EULER-CN uses a simple labeling procedure that assigns the copy numbers in a greedy fashion by iteratively checking the flow balance condition (the Kirkhoff law) in every vertex and modifying multiplicities of some edges. For example, since there are 4 edges leaving vertex 3 and only one edge $(2,3)$ entering vertex 3, the edge $(2,3)$ is a tangle with multiplicity at least 4. Similar analysis implies that the edges $(5,6)$ and $(7,2)$ are tangles with multiplicity at least 2. Assigning these multiplicities to the tangles $(2,3)$, $(5,6)$ and $(7,2)$ produces a balanced graph satisfying the Kirkhoff law.

Although this procedure always led to correct solutions for all tangles in the studied bacterial genomes, it has to be complemented by a more rigorous flows in networks procedure for larger genomes. Let’s assume for simplicity that $G(V, E, w)$ is a graph with a single source s and sink t and unit edge multiplicities ($w(e) = 1$ for all $e \in E$). For a vertex $v \in V$ define $div(v)$ as the sum of multiplicities of edges entering v minus the sum of multiplicities of edges leaving v . A graph is called *balanced* if $div(v) = 0$ for every vertex $v \in V$, except for the source and the sink. The graph in Fig. 7 is not balanced and we are interested in a way to increase the multiplicities of its edges to make a balanced graph $G(V, E, f)$ with the same edge set and $w(e) \leq f(e)$ for all $e \in E$. The balanced graph $G(V, E, f)$ is called a *flow* from s to t and $div(t)$ is called the *value* of flow f .

Copy Number Problem. For an edge e in a graph $G(V, E, w)$, find a flow $G(V, E, f)$ minimizing the multiplicity $f(e)$ of the edge e .

The Copy Number Problem can be solved by finding

a minimal flow in a network with lower capacity bounds (Grotschel et al., 1993). To make this reduction, we add an artificial edge from the sink to the source in G and assume the unit lower capacities on all edges in G . A flow with the minimum multiplicity of the edge $e = (v, w)$ in this graph (Fig. 7, right) corresponds to a solution of the *Minimum Flow Problem* from w to v satisfying the lower capacity constraints (w and v are treated as new source and sink, correspondingly). The minimum flows can be found by an application of the Min-Flow Max-Cut theorem:

Min-Flow Max-Cut Theorem. For a directed acyclic graph $G(V, E, w)$ with lower capacity bounds, the minimum flow from w to v equals to the capacity of the maximum directed cut separating w from v .

In Fig. 7, right, the capacity of the maximum directed cut separating vertex 3 from all other vertices is 4, exactly the multiplicity of the tangle (2,3) and the multiplicity of the corresponding 83 bp repeat in the *Mycoplasma genitalium* genome. In a more general formulation, one may want to minimize the sum of the flows over all edges of the graph. This generalization may be addressed by an application of the Minimum-Cost Circulation Problem that also can be solved in polynomial time (Grotschel et al., 1993).

CONCLUSIONS

Difficulties in resolving repeats led to the introduction of the double-barreled DNA sequencing and the breakthrough genome sequencing efforts at Celera (Myers et al., 2000). The Celera assembler is a two-stage procedure that includes masking repeats at the overlap-layout-consensus stage with further ordering of contigs via the double-barreled information. The DB step of the Celera assembler is influenced by the previous “overlap-layout-consensus” step. All “overlap-layout-consensus” algorithms are “afraid” of repeats and the Celera assembler has no choice but to mask the repeats. EULER does not require masking the repeats but instead provides a clear view of repeats (tangles) in the genome. These tangles may be resolved either by double-barreled information (EULER-DB) or by additional PCR experiments (EULER-PCR, work in progress).

ACKNOWLEDGEMENTS

We are indebted to David Harper and Julian Parkhill for their advice and feedback, as well as for providing sequencing data from different bacterial sequencing projects. We are grateful to Michael Fonstein, Eugene Goltsman, Zufar Mulyukov, Julian Parkhill, Alexei Sorokine, and Michael Waterman for useful discussions and providing sequencing data. Equipment used in this research was supported in part by the UCSD Active Web Project, NSF Research Infrastructure Grant Number 9802219.

REFERENCES

- Bonfield, J. K., K. F. Smith, and R. Staden (1995). A new DNA sequence assembly program. *Nucleic Acids Research* 23, 4992–4999.
- Drmanac, R., I. Labat, I. Brukner, and R. Crkvenjakov (1989). Sequencing of megabase plus DNA by hybridization: theory of the method. *Genomics* 4, 114–128.
- Fleischmann, R. and et al. (1995). Whole-genome random sequencing and assembly of haemophilus influenzae. *Science* 269, 496–512.
- Gordon, D., C. Abajian, and P. Green (1998). Consed: a graphical tool for sequence finishing. *Genome Research* 8, 195–202.
- Green, P. (1994). Documentation for phrap. <http://www.genome.washington.edu/UWGC/analysis-tools/phrap.htm>.
- Grotschel, M., L. Lovasz, and A. Schrijver (1993). *Geometric Algorithms and Combinatorial Optimization*. Springer-Verlag.
- Huang, X. and A. Madan (1999). CAP3: A DNA sequence assembly program. *Genome Research* 9, 868–877.
- Kececioğlu, J. and E. Myers (1995). Combinatorial algorithms for DNA sequence assembly. *Algorithmica* 13, 7–51.
- Kent, W. and D. Haussler (2001). Assembly of the working draft of the human genome with gigassembler. (*submitted*).
- Lysov, Y., V. Florent'ev, A. Khorlin, K. Khrapko, V. Shik, and A. Mirzabekov (1988). DNA sequencing by hybridization with oligonucleotides. *Doklady Academy Nauk USSR* 303, 1508–1511.
- Myers, E. M. (1995). Toward simplifying and accurately formulating fragment assembly. *Journal of Computational Biology* 2, 275–290.
- Myers, E. W., G. G. Sutton, A. L. Delcher, I. M. Dew, D. P. Fasulo, M. J. Flanigan, S. A. Kravitz, C. M. Mobarry, K. H. Reinert, K. A. Remington, E. L. Anson, R. A. Bolanos, H. H. Chou, C. M. Jordan, A. L. Halpern, S. Lonardi, E. M. Beasley, R. C. Brandon, L. Chen, P. J. Dunn, Z. Lai, Y. Liang, D. R. Nusser, M. Zhan, Q. Zhang, X. Zheng, G. M. Rubin, M. D. Adams, and J. C. Venter (2000). A whole-genome assembly of *Drosophila*. *Science* 287, 2196–2204.
- Peltola, H., H. Soderlund, and E. Ukkonen (1984). SEQAID: a DNA sequence assembling program based on a mathematical model. *Nucleic Acids Research* 12, 307–321.
- Pevzner, P. (1989). l -tuple DNA sequencing: computer analysis. *Journal of Biomolecular Structure and Dynamics* 7, 63–73.
- Pevzner, P., H. Tang, and M. Waterman (April 2001). A new approach to fragment assembly in DNA sequencing. In *Proceedings of the Fifth Annual International Conference on Computational Molecular Biology (RECOMB-01)*, Montreal, Canada, pp. 256–267. ACM Press.
- Pevzner, P. A. (2000). *Computational Molecular Biology: An Algorithmic Approach*. The MIT Press.
- Roach, J., C. Boysen, K. Wang, and L. Hood (1995). Pairwise end sequencing: a unified approach to genomic mapping and sequencing. *Genomics* 26, 345–353.
- Sutton, G., O. White, M. Adams, and A. Kerlavage (1995). TIGR assembler: A new tool for assembling large shotgun sequencing projects. *Genome Science & Technology* 1, 9–19.
- Weber, J. and G. Myers (1997). Whole genome shotgun sequencing. *Genome Research* 7, 401–409.