```
% Paper on Drosophila version of unitigger
% For submission to conference or journal
% $Id: DrosUnitigger.tex,v 1.1.1.1 2004/04/14 13:42:55 catmandew Exp $
%\documentstyle[12pt]{article}
\documentclass{article}
\markright{Celera Confidential}
\usepackage{graphicx}
\usepackage{vmargin}

\newcommand{\Curl}{d}
\newcommand{\Homotopy}[1]{{H}_{#1}}
\newcommand{\Su}{{\Uparrow}} % Stencil up
\newcommand{\Sd}{{\Downarrow}} % Stencil down
\newcommand{\Ip}{{\cal I}} % interior product
\newcommand{\Ex}{{\cal E}} % extension
\newcommand{\Lie}{{\L}}
\newcommand{\Dual}{{*}}
\newcommand{\inner}[2]{ {\langle {#1} \mid {#2}\rangle} }

\begin{document}

\setpapersize{USletter}
\setmarginsrb{1in}{1in}{1in}{1in}{0pt}{0mm}{0pt}{0mm}

\title{Fragment Overlap Graph Transformations for Large Whole Genome Shotgun
Assembly of Inbred Strains}
\author{
Clark ~Mobarry\thanks{Celera Genomics, Rockville, MD USA}
\and
Ian Dew
\and
Gene Myers
\and
Granger Sutton
\and
Others
}
\date{September 2001}
\maketitle



\begin{abstract}

Assembly of large shotgun datasets requires powerful and efficient
algorithms to remove redundant overlaps from the fragment overlap
graph and identify contigs. The double-barreled genome shotgun
approach further requires that paired end reads be reliably anchored
in contigs that represent unique genomic sequence to order and orient
contigs. This paper presents the Unitigger module of the Celera
Assembler, which performs graph reduction and unique contig (unitig)
identification in three phases - fragment localization, transitive
overlap reduction, and chunking and classifying overlapping chains of
fragments into unique and repetitive contigs.

\end{abstract}

\section{Intro: Context of Celera WGS Assembler}
The Celera Assembler is the first of a new breed of genome assembly
programs designed for large and very large double-barreled whole
genome shotgun datasets. This system extends the conventional
```

overlap-layout-consensus paradigm with the additional step of ordering, orienting, and spacing contigs into scaffolds (see figure 1). The unitigger module described in this paper performs the layout function with the additional requirement of providing a suitable substrate for scaffolding. The version of the unitigger described here was developed for assembling an inbred strain of \emph{Drosophila melanogaster}.

\section{Definitions}

\subsection{Fragment overlap}
A $\mathnormal{\ge}$ 96\% identity matching sequence interval $\mathnormal{\ge}$ 40bp between two fragments that is terminated in each direction by the end of a fragment.

\subsection{Dovetail overlap}
An overlap between two fragments in which only one end of each fragment is in the overlapping interval. The 5' end of a fragment is its prefix, the 3' end its suffix. Given two fragments A and B, a normal dovetail is an overlap between a proper suffix of A and a proper prefix of the reverse complement of B. Definitions of innie, outtie, and antinormal dovetails follow. We have adopted the convention of expressing antinormal dovetails as normal dovetails by reverse-complementing and swapping fragments A and B. See figure n.

\subsection{Containment overlap}
An overlap between two fragments in which the sequence of one fragment is wholly contained in that of the other fragment. Containment overlaps are expressed in two forms, forward and reverse. The A fragment is always in forward orientation and contains either B (forward) or the reverse-complement of B (reverse).

\subsection{Overlap representation}
Overlaps are represented in terms of A fragment, B fragment, orientation, type (dovetail or containment) - which is redundant, a-hang, b-hang, minimum offset, and maximum offset. The a-hang is the number of basepairs in fragment A not in a given dovetail overlap, and the number of prefix basepairs in fragment A not in a given containment overlap. A-hang values are always non-negative. The b-hang is the number of basepairs in fragment B not in a dovetail overlap, and the number of basepairs in the suffix of fragment A not in a containment overlap. Minimum and maximum offsets specify a range of a-hang values in the event a pair of fragments have multiple overlaps of the same type and orientation. See figure q. The a-hang/b-hang notation is used to represent overlaps because it is more precise than using the number of basepairs in the overlap. Given a gapped consensus sequence of an overlap alignment, a single number may not equal the number of basepairs in the overlap in either fragment. In contrast, the a-hang and b-hang identify the exact basepair in each fragment where the overlap begins and ends.

\subsection{Fragment overlap graph}
A graph where the vertices represent fragments and edges represent overlaps. Each vertex has two ports, representing the prefix and suffix of the fragment.

\subsection{Reduced fragment overlap graph}
A subgraph of the fragment graph in which redundant, repetitive, non-useful, and spurious overlap edges have been removed. A correctly reduced fragment overlap graph contains all possible correct fragment

assemblies.

\subsection{Dovetail run}
  A simple path in the reduced fragment overlap graph, terminated by a fragment on either end, consisting of fragments and dovetail overlaps where each fragment end in the run has a dovetail overlap to zero or one other fragment end.

\subsection{Dovetail path framework}
  A non-cyclic subgraph of a fragment overlap graph that satisfies properties in (reference Myers, J. Comp. Bio. vol.2 1995)

\subsection{Chunk}
  A maximal subgraph of the reduced fragment overlap graph that consists of only non-contained fragments and dovetail overlap edges, where each overlap edge is the only dovetail overlap edge adjacent to each fragment end in the overlap.

\subsection{Unitig}
  A dovetail path framework in the reduced fragment overlap graph where the dovetail path is a chunk. This differs from a chunk by including contained fragments and dovetail edges to them. This extends the definition of Myers to include contained fragments.

\subsection{U-Unitig}
  A unitig that has certain statistical properties indicating it is highly likely to represent uniquely occurring genomic sequence.

\section{\emph{Drosophila} genome and data sets}
The \emph{Drosohpila} genome was, at the time, the largest genome sequenced using the whole genome shotgun approach. It is ~180Mbp in size, with ~120Mbp in clonable euchromatin. Celera, Berkeley \emph{Drosophila} Genome Project (BDGP), and European \emph{Drosophila} Genome Project (EDGP) sequencing laboratories produced 3.156 million reads from X BAC and P1 libraries generated from an isogenic \emph{y; cn bw sp} strain (Adams, 2000). After trimming to 98\% accuracy, the mean read length was 551bp and the observed mean accuracy was 99.5\%. This represents approximately 14.5-fold coverage, exceeding the 10-fold coverage requested as based on Lander-Waterman statistics and results of assemblies using simulated data sets.

\section{Phases of assembly}
The Celera Assembler consists of five modules as shown in Figure x. The screener masks fragment intervals that have high-fidelity local alignment matches to any of a curated set of repeats. Screening is performed to speed up the overlap computation and reduce the number of repetitive overlaps generated by it. The overlapper performs the all-against-all comparison of fragment sequences to identify high-fidelity ($\mathnormal{\ge}$96\%) fragment overlaps of $\mathnormal{\ge}$ 40bp. The output of the overlapper is an implicit fragment overlap graph. The unitigger localizes, reduces, and chunks the fragment overlap graph to produce unique contigs (unitigs) as a basis for scaffolding. The scaffolder generates contigs as single or overlapping sets of unitigs and orders and orients these into scaffolds using the mate pair data generated by the double-barreled shotgun sequencing. Finally, the consensus module generates a consensus sequence for each contig in each scaffold.

```
\section{The unitigger}
The purpose of the unitigger is to produce unitigs as a basis for
scaffolding. The intent is for the vast majority of unitigs to be
U-unitigs, with the rest comprising over-compressed repeats. The
scaffolding algorithm operates initially and primarily on
U-unitigs. To associate a pair of U-unitigs in an overlapping contig
or a gapped scaffold there must be at least two pairs of end sequenced
reads anchored in the U-unitigs where the mean and standard deviations
of the distance estimates between the reads (based on clone libraries)
agree via a $\mathnormal{\chi}^2$ test. The fraction of pairs of
fragments where both are in U-unitigs is approximately the square of
the fraction of fragments in U-unitigs. For example, if 75\% of
fragments are in U-unitigs, then approximately 56\% of end read pairs
will both be in U-unitigs. Thus it is critical to incorporate as many
fragments into U-unitigs as possible.

%**** Text here is based on AS\_PRIV\_DROS\_ASSEMBLER tag of code ****

\subsection{Input and Localization}
%***** INPUT *****
%default containment rule was AS\_CGB\_CONTAINMENT\_RULE\_4
%no evidence that this was over-ridden on the command line

%Read fragments into frags array
%- track UID, IID, type, length
%- set label as AS\_CGB\_SPANNING\_FRAG - assume it spans a chunk
%- set contained flag to false
%Read overlaps into edges array
%- a-hang, b-hang, min\_offset, max\_offset, quality
%- compute approximate bmin, bmax
%- set flags iasx and ibsx for suffixes in overlap
%- edge is either to-contained (containment) or interchunk (dovetail)

%***** LOCALIZE - fragmentOverlapGraphBuild1 *****
%\#undef AHG\_FIRST
%\#undef SORT\_PRE\_CONDITIONER

%The edges of each fragment are sorted by fragment index \& suffix flag

%\#define IN\_PLACE\_PERMUTATION

%edges for each fragment are permuted in place (ScatterInPlace\_VA) by
%first assigning a rank to each edge

%\#define QUICK\_SORT2
%use quicksort to sort edges in fragment end assemblies by ABS(ahg)

%duplicate edges are deleted

%then the graph is reordered via a breadth-first search of connected
%comonents, assigning ranks to fragments

%number of weakly connected components is determined circular chunks
%are identified

%fragments are reordered using an in-place permutation
%edges of each fragment are then reordered? why?

%Perhaps this should go in the results section.
The overlapper module produced 212 million overlaps with a mean of
```

33.7 overlaps per fragment end. This data actually approximates a Poisson distribution with mean 13.7, and the tail extending up to 4000.

The set of fragments and overlaps is an implicit (chordal? - what about missed overlaps?) directed graph of fragments and overlaps. We chose to represent fragments as dual-ported vertices and overlaps as edges. Alternative formulations were considered, including representing fragments as edges. However these did not offer any clear advantages. The unitigger represents this as an adjacency list in which each directed edge is replaced by a pair of symmetric undirected edges. In the directed graph, dovetail overlaps are represented as single directed edges. Containment overlaps are represented as a pair of single directed edges, as this type of overlap involves both ends of the contained fragment.

The fragments and corresponding edges are localized in machine memory to speed up subsequent graph traversals. This is done by rank ordering fragments in each connected component via a breadth-first search followed by an in-place permutation. The edges indicident on each fragment end are sorted by the absolute value of the a-hang, effectively from 'thickest' to 'thinnest', via quicksort.

```
\subsection{Graph Reduction}
%NOTE: 4 CGB passes are specified in the script
%
%  The raw fragment overlap graph contains numerous redundant
%overlaps. Treatment of containments
%
%***** fragment\_overlap\_graph\_build\_2 *****
%contained fragments are identified and marked
%the median fragment end degree is computed - just for informational purposes
%edges beyond the adjacency degree threshold are marked for removal
%  only edges beyond the threshold for both fragment ends
%  default is 100 - was this over-ridden?
%edges are actually deleted
%
%***** fragment\_overlap\_graph\_build\_3 *****
%all edges to contained fragments are marked as such
%transitively inferrable edges are marked
%  iterate over each fragment
%    iterate over each end
%

%I didn't examine how containment overlaps are checked & marked for removal...

%transitive overlap inference slop is:
%abs(lenMiddleFrag + lenOppositeOverlap - lenRightOverlap - lenLeftOverlap) <= 20 +
.07 * (bpInOverlap)
%OR
%20 + .07*(lenAFrag - lenAAhang + lenBFrag - lenBBhang)
%Transitively inferrable edges are marked
%local containment rules
%
```

NOTE: This text describes how things are done. Obviously, this is where the description of overlap calculus goes along with the theoretical basis for it.

The fragment overlap graph is reduced to eliminate redundant overlap edges. This reduces the graph complexity and makes identification of unitigs straightforward.

Whereas contained fragments have been set aside in previous formulations, we chose to keep contained fragments in the graph and eliminate redundant edges to contained fragments. Globally contained fragments (i.e., fragments conained by one or more other fragments in the graph) are identified as are all edges (containment or dovetail) incident on them.

We chose to keep only the 100 thickest overlap edges incident to each fragment end.

Edges marked as transitively inferrable via the following overlap calculus.

\subsection{Chunking}

```
%***** chunk\_classification\_dvt *****
%mark fragments - deleted, contained, solo, hanging, or spanning
%
%chunk classification of overlaps - intrachunk or interchunk - by \# of
%other edges incident on each end of each fragment, but edges to
%contained fragments cannot be intrachunk
%
%chunk classification of fragments -
%  if it was hangingChunk or hanging
%    hangingChunk if 1 edge off 1 end, 0 off other
%    hanging if no edges?
%  if it was spanning or intrachunk or interchunk
%    spanning has no overlaps
%    interchunk has >1 overlaps off at least one end
%    intrachunk has exactly 1 overlap off each end
%
%***** CGB *****
%count guides - so as to exclude from arrival rate
%
%chunkGraphBuild
%  makeTheChunks 2 passes
%    first makes light chunks - no contained
%    second makes heavy chunks - includes contained
%    compute coverage statistic of chunk
%  findBranchPoints - determines where off end of chunk sequence diverges
%  edgeTrimmer
%  countChimeras
%
%
%
%  Alu smashing
%  Discriminator statistic
```

\section{Results}
\subsection{Output}

Clark has generated more stats on more characteristics of unitigs than one would put in a book on unitigging... Select a subset for here.

Nature of output - unitigs, a-stat, unitig overlaps
54,000 unitigs with 2+ fragments, 3.104 million overlaps between unitigs
9413 U-unitigs, mean 12.2kbp, 115.4Mbp total
After alu smashing?:
8389 U-unitigs, mean 13.9kbp, 116.3Mbp total
As input to scaffolder
U-unitigs provided 20,000 bundles of 10.6 mate pairs per bundle average

\subsection{Performance}

```
  sizes of arrays? total memory? time for input, localization,
  reduction, chunking? Machine specs?

\begin{thebibliography}{plain}
  \bibitem{dros1} Mark D. Adams, et al., {\it The Genome Sequence of
  Drosophila melanogaster}, Science 287, 2185 (2000).
  \bibitem{dros2} Eugene W. Myers, et al., {\it A Whole-Genome Assembly of
  Drosophila}, Science 287, 2196 (2000).
\end{thebibliography}

\end{document}
```