Detectability of Certain Dark-Genome-Matter Candidates

SUTTA Algorithms for Assembling Genomes Correctly

Bud Mishra

Room 1002, 715 Broadway, Courant Institute, NYU, New York, USA

Jointly with Giuseppe Narzisi (Graduate Student, Courant)
B Mishra  Detectability of Certain Dark-Genome-Matter Candidates
Dude, Where is my genome??

Human Genome Anniversary

with Francis Collins, Eric Lander and Nicholas Wade
in Science & Health
on Monday, July 12, 2010
★★★★★
Discoveries Waiting!

MEDICINE

Revolution Postponed

The Human Genome Project has failed so far to produce the medical miracles that scientists promised. Biologists are now divided over what, if anything, went wrong—and what needs to happen next

By Stephen S. Hall
Genomic Dark Matter

Quality of the human genome sequence remains unassessed!

- Most of the reference sequences are genotypic (i.e., non-haplotypic) and lack long range information (i.e., fail to characterize rearrangements, duplications, inversions and translocations).

- Resequencing reveals about 30% of the resequenced reads not aligning to the reference sequences: **Genomic Dark Matter**.

- **Optical Mapping**, a high-throughput, high-resolution single-molecule system reveals many previously unknown genome structural variants not captured in the reference sequences (Teague et al.)

- **Genome-Wide Association Studies** based on the currently available genomic data have proven inadequate in explaining common diseases.
Characterization of missing human genome sequences and copy-number polymorphic insertions

Jeffrey M Kidd1, Nick Sampson2, Francoca Antonacci1, Tina Graves3, Robert Fulton1, Hillary S Hayden1, Can Alkan1, Maika Malig1, Mario Ventura4, Giuliana Giannuzzi2, Joelle Kallick3, Paige Anderson5, Anya Tsalenko5, N Alice Yamada2, Peter Tsang1, Rajinder Kaul1, Richard K Wilson3, Lauralay Bruhn2 & Evan E Eichler1,5

The extent of human genomic structural variation suggests that there must be portions of the genome yet to be discovered, annotated and characterized at the sequence level. We present a resource and analysis of 2,363 new insertion sequences corresponding to 720 genomic loci. We found that a substantial fraction of these sequences are either missing, fragmented or misassigned when compared to recent de novo sequence assemblies from short-read next-generation sequence data. We determined that 18–37% of these new insertions are copy-number polymorphic, including loci that show extensive population stratification among Europeans, Asians and Africans. Complete sequencing of 156 of these insertions identified new events and conserved noncoding sequences not yet represented in the reference genome. We developed a method to accurately genotype these new insertions by mapping next-generation sequencing datasets to the breakpoint, thereby providing a means to characterize copy-number status for regions previously inaccessible to single-nucleotide polymorphism microarrays.

We recently reported efforts to systematically map and sequence human genome structural variation using a fosmid end-sequence pair mapping approach6–11. We fragmented genomic DNA from nine humans and subcloned 40-kb segments. Using standard capillary sequencing, we generated reads from both ends of each fragment (end-sequence pairs) and mapped clones to the human reference genome. Structural differences (inversions, deletions, insertions and translocations) between the reference genome assembly and the library source were identified on the basis of the mapped location of the end-sequence pairs. As the individual fosmid clones were retained, the procedure allowed simultaneous discovery and complete sequence characterization of a subset of structural variant loci including new insertion sequences common to most individuals but not represented in the human reference genome. Here we present a detailed sequence and copy-number analysis of these segments missing from the human reference genome.
Hooplas, Hypes, Haplotypes

Whole-Genome Human Optical Map (above) constructed by our Gentig algorithm (Anantharaman, Mishra, Schwartz, 1999).
A Mind is a Terrible Thing to Blow!

Eric S. Lander of Broad Institute at M.I.T. calls recent progress “mind-blowing.”
The DNA sequence of an organism is sheared into a large number of small fragments (8-10x coverage), the ends of the fragments are sequenced (∼ 500 bp), then the resulting sequences are joined together using a computer program called *assembler*.
Assume: If two sequence reads (strings of letters produced by the sequencing machine) share the same string of letters, then they must have originated from the same genomic location.
Given a set of sequence fragments the object is to find the shortest common supersequence.

**Algorithm 1: GREEDY - pseudo code**

**Input:** Set of reads  
**Output:** Set of contigs  
1. Calculate pairwise alignments of all fragments;  
2. Choose two fragments with the largest overlap;  
3. Merge chosen fragments;  
4. repeat  
5. step 2. and 3.  
6. until only one fragment is left ;  
7. return Set of contigs;

This is a suboptimal approach, but it’s our best idea!

Two other good ideas: OLC (Overlap-Layout-Consensus) and SBH (Sequencing-By-Hybridization).
<table>
<thead>
<tr>
<th>Name</th>
<th>Algorithm</th>
<th>Author</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimus (AMOS)</td>
<td>OLC</td>
<td>Sommer, D.D. et al.</td>
<td>2007</td>
</tr>
<tr>
<td>Newbler</td>
<td>OLC</td>
<td>454/Roche</td>
<td>2009</td>
</tr>
<tr>
<td>SUTTA</td>
<td>B&amp;B</td>
<td>NYU/Abraxis (unpublished)</td>
<td>2009/2010</td>
</tr>
<tr>
<td>TIGR</td>
<td>Greedy</td>
<td>TIGR</td>
<td>1995 / 2003</td>
</tr>
<tr>
<td>Phusion</td>
<td>Greedy</td>
<td>Mullikin JC, et.al.</td>
<td>2003</td>
</tr>
<tr>
<td>CAP3, PCAP</td>
<td>Greedy</td>
<td>Huang, X. et al.</td>
<td>1999 / 2005</td>
</tr>
<tr>
<td>Euler-SR</td>
<td>SBH</td>
<td>Chaisson, MJ. et al.</td>
<td>2008</td>
</tr>
<tr>
<td>ALLPATHS</td>
<td>SBH</td>
<td>Butler, J. et al.</td>
<td>2008</td>
</tr>
<tr>
<td>SOAPdenovo</td>
<td>SBH</td>
<td>Ruiqiang Li, et al.</td>
<td>2009</td>
</tr>
<tr>
<td>SHARCGS</td>
<td>Prefix-Tree</td>
<td>Dohm et al.</td>
<td>2007</td>
</tr>
<tr>
<td>SSAKE</td>
<td>Prefix-Tree</td>
<td>Warren, R. et al.</td>
<td>2007</td>
</tr>
<tr>
<td>VCAKE</td>
<td>Prefix-Tree</td>
<td>Jeck, W. et al.</td>
<td>2007</td>
</tr>
<tr>
<td>Sequencher</td>
<td>-</td>
<td>Gene Codes Corporation</td>
<td>2007</td>
</tr>
<tr>
<td>SeqMan NGen</td>
<td>-</td>
<td>DNASTAR</td>
<td>2008</td>
</tr>
<tr>
<td>Staden gap4 package</td>
<td>-</td>
<td>Staden et al.</td>
<td>1991 / 2008</td>
</tr>
<tr>
<td>MIRA, miraEST</td>
<td>-</td>
<td>Chevreux, B.</td>
<td>1998 / 2008</td>
</tr>
<tr>
<td>NextGENe</td>
<td>-</td>
<td>Softgenetics</td>
<td>2008</td>
</tr>
<tr>
<td>CLC Genomics Workbench</td>
<td>-</td>
<td>CLC bio</td>
<td>2008 / 2009</td>
</tr>
</tbody>
</table>
Toxic Assembly Recovery Program

Shortest Common Superstring Problem

Definition (Shortest Superstring Problem)

Given a set of strings \( \{s_1, s_2, \ldots, s_n\} \) find the shortest string \( T \) such that \( \forall i, s_i \) is a substring of \( T \).

- **First issue**: \( \mathcal{NP} \)-completeness! [Gallant et al. 1980]
- **Second issue**: Incorrectlyformulates the assembly problem!!
Screwed!

Unless $P = NP$, this incorrect formulation (SCSP) results in an intractable problem...

“The shortest superstring problem, an elegant but flawed abstraction: [since it defines assembly problem as finding] a shortest string containing a set of given strings as substrings.”

The Role of Algorithmic Research in Computational Genomics, Richard M. Karp

IEEE Computer Society Bioinformatics Conference, August 14, 2003
Intractability
(NP-Completeness)

A class of problems having two properties:

- Any given solution to the problem can be verified quickly (in polynomial time); the set of problems with this property is called NP (nondeterministic polynomial time).
- Any NP problem can be converted into this one by a transformation of the inputs in polynomial time.

There are thousands of important computational problems that represent essentially one problem in many disguises!
A class of problems having two properties:

- Any given solution to the problem can be verified quickly (in polynomial time); the set of problems with this property is called NP (nondeterministic polynomial time).
- Any NP problem can be converted into this one by a transformation of the inputs in polynomial time.

There are thousands of important computational problems that represent essentially one problem in many disguises!
Intractability
(NP-Completeness)

For instance, think about sequence reads as “towns” and overlaps as “roads.” Then Shortest Common Superstring problem is same as visiting all the towns (never more than once) using the roads in a minimum-distance tour.

\[
\text{SCSP} (= \text{ShortestCommonSuperstringProb})
\xrightarrow{\Rightarrow} \text{HAM}(\text{TSP} = \text{TravlingSalesmanProblem})
\xrightarrow{\Rightarrow} \text{SAT}(= \text{Satisfiability})
\xrightarrow{\Rightarrow} \text{LatinSquare}
\xrightarrow{\Rightarrow} \text{Sudoku}
\]
It is easy to verify if a solution of a Sudoku problem is correct.

If you exhaustively try all possible configurations, you can find the correct solution, surely. This takes very long (exponential) time.

Nobody has (yet) a rigorous argument to convince us that there might not be a better/efficient way to solve Sudoku.

Not all instances are hard: they range from easy, medium and hard to devilishly hard.

But if you try to create a Sudoku puzzle at random, with high probability, it will be easy to solve. Pathologically hard instances can be hard to construct.
Tell the biologist to think about easier problems.

Come up with a simpler problem that vaguely looks like the original problem. Solve the easy problem, even if it gives the wrong solution. Tell the biologist to learn to live with incomplete or incorrect solutions.

Work with the biologist to cheat. Design experiments and technologies so that they only generate easy instances of a hard problem. Solve them correctly.

Solve the problem by exhaustive search, but learn to constrain the search space intelligently.

Try all of the above (+ kitchen sink)!
Traveling Salesman Problem

A traveling salesman wishes to visit a given number of cities, returning to the starting point in a tour. Each pair of cities incurs a cost proportional to the distance between these cities. The Traveling Salesman Problem (TSP) is to find a tour with minimum costs.

<table>
<thead>
<tr>
<th>Year</th>
<th>Solvers</th>
<th>Costs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1994</td>
<td>Applegate, Bixby, Chvátal, Cook</td>
<td>7,397</td>
</tr>
<tr>
<td>1998</td>
<td>Applegate, Bixby, Chvátal, Cook</td>
<td>13,509 (USA tour)</td>
</tr>
<tr>
<td>2001</td>
<td>Applegate, Bixby, Chvátal, Cook</td>
<td>15,112 (D tour)</td>
</tr>
<tr>
<td>2004</td>
<td>App., Bixby, Chvátal, Cook, Helsgaun</td>
<td>24,978 (Swe tour)</td>
</tr>
<tr>
<td>2006</td>
<td>App., Bixby, Chvátal, Cook, Helsgaun</td>
<td>85,900</td>
</tr>
</tbody>
</table>

Table: TSP Competition.

These five largest instances were solved by Concorde which is based on branch-and-cut.
Traveling Salesman Problem: World Tour
**SAT Solver**

- Given a Boolean formula (in CNF), determine if, under some truth-assignment, the formula will evaluate to true. This is a classical NP-complete problem.

- A DPLL SAT solver employs a systematic backtracking search procedure to explore the (exponentially-sized) space of variable assignments looking for satisfying assignments. The basic search procedure is based on the Davis-Putnam-Logemann-Loveland algorithm (DPLL). It has a theoretically exponential lower bound.
Modern SAT solvers come in two flavors: “conflict-driven” and “look-ahead.”

- Conflict-driven solvers augment the DPLL search algorithm with efficient conflict analysis, clause learning, non-chronological backtracking, “two-watched-literals” unit propagation, adaptive branching, and random restarts.
- Look-ahead solvers have especially strengthened reductions (going beyond unit-clause propagation).

**SAT Competitions:** The conflict-driven MiniSAT (the 2005 SAT competition) only has about 600 lines of code. The look-ahead solver march_dl won a prize at the 2007 SAT competition.
**Figure 4** Backbone fractions as a function of $\alpha$ for 2-SAT and 3-SAT. The data were extracted from 3,000–15,000 samples for the 2-SAT cases (on the left) with $N$ values of 20 (red), 30 (blue), 45 (green), 100 (black), 200 (light blue), and 500 (orange). The results for $N < 45$ were obtained by exhaustive enumeration, examining all assignments, while those for $N > 100$ used a modified Davis-Putnam search procedure. The 3-SAT cases were studied by exhaustive enumeration in 7,500–30,000 samples for $N$ values of 16 (red), 20 (blue), 24 (green), and 28 (black). The vertical lines mark the observed SAT/UNSAT thresholds in the limit $N \to \infty$. For 2-SAT, data obtained from larger sizes show that the backbone fraction at the threshold tends towards zero.
Fragments:
- A set of fragments/reads $F = \{ r_1, r_2, \ldots, r_N \}$, s.t. $r_i \in \{ A, C, G, T \}^*$.
- Each read is represented as pairs of integers $(s_i, e_i)$, $i \in [1, |F|]$ where $1 \leq s_i, e_i \leq |R|$, and $R$ is the reconstructed string (the order of $s_i$ and $e_i$ encodes the orientation of the read).

Overlaps:
- Use Smith-Waterman algorithm to compute the best alignment between a pair of strings.
Layout Representation

Let us define the layout $L$ associated to a set of reads $F = \{r_1, r_2, \ldots, r_N\}$ as follows:

$$L = r_1 \overset{\pi_1}{\Rightarrow} r_2 \overset{\pi_2}{\Rightarrow} r_3 \overset{\pi_3}{\Rightarrow} \cdots \overset{\pi_{N-1}}{\Rightarrow} r_N \quad (1)$$

where there are no containments (contained reads can be initially removed and then added later after the layout has been created).

**Definition (Consistency Property)**

*A layout $L$ is consistent if the following property holds for $i = 2, \ldots, N - 1$:

$$\pi_{i-1} \overset{\pi_i}{\Rightarrow} r_i \overset{\pi_i}{\Rightarrow} \text{ iff } \text{ suffix}_{\pi_{i-1}}(r_i) \neq \text{ suffix}_{\pi_i}(r_i) \quad (2)$$

The estimated start positions for each fragment are given by:

$$sp_1 = 1, \quad sp_i = sp_{i-1} + \pi_{i-1} \cdot \text{hang}_{r_{i-1}} \quad \text{if } i > 1 \quad (3)$$
Layout for a set of fragments $F = \{A, B, C, D, E, F, G\}$ with a sequence of overlaps $\pi^N_{(A,B)}, \pi^I_{(B,C)}, \pi^N_{(C,D)}, \pi^I_{(D,E)}, \pi^N_{(E,F)}, \pi^N_{(F,G)}$
Definition (Sequence Assembly Problem)

Given a collection of fragment/reads $F = \{r_i\}_{i=1}^N$ and a tolerance level (error rate) $\epsilon$, find a reconstruction $R$ whose layout $L$ is $\epsilon$-valid, consistent and such that the following set of properties (oracles) are satisfied:

- **Overlap-Constraint (O):** The cumulative overlap score of the layout is optimized.

NP-completeness of this formulation becomes a serious issue.
If we look for a reconstruction of minimum length, the reconstructed string can have many errors due to *repeats*. 

**Correct Assembly**

```
A       R_1       B       R_2       C

```

**Mis-assembly**

```
A       R_1 + R_2       C

```

```
B
```

Detectability of Certain Dark-Genome-Matter Candidates
Greedy Strategy

- Pick the highest scoring overlap.
- Merge the two fragments (add this new sequence to the pool of sequences).
- Heuristically correct regions of the overlay in some plausible manner (whenever possible).
- Regions that do not yield to these error-correction heuristics are abandoned as irrecoverable and shown as gaps.
- Repeat until no more merges can be done.
**Idea:** Construct a graph in which nodes represent reads and edges indicate overlaps.

**Goal:** Need to solve for a Hamiltonian path!

**Strategy:**
- Remove contained and transitivity edges.
- Collapse "unique connector" overlaps (chordal subgraph with no conflicting edges).
- Use mate-pairs to connect and order the contigs.
Sequencing by Hybridization
(EULER 2001, Velvet 2008)

- **Idea**: Break the reads into overlapping $n$-mers (an $n$-mer is a substring of length $n$). Build a DeBruijn graph in which each edge is an $n$-mer and the source and destination nodes are respectively the $n-1$ prefix and $n-1$ suffix of the corresponding $n$-mer.

- **Goal**: find a path that uses all the edges (an Eulerian path) → linear time algorithm (however actual performance similar to the overlap-layout-consensus approach).

- **Problem**: Errors in the data can introduce many erroneous edges!
DeBruijn graph for the list $L = \{AAA, AAC, ACA, CAC, CAA, CGC, GCG\}$. The Euler path is:
$AC \rightarrow CA \rightarrow AC \rightarrow CG \rightarrow GC \rightarrow CA \rightarrow AA \rightarrow AC$
Parameters under control: Read Length, Resolution, Accuracy, Throughput, and Latency

- Dideoxy chain termination to measure lengths (base positions): Accuracy in length requires large number of clonal copies. 
  *Problems*: Distribution of PCR clones, measurements by electrophoresis, volume, Joule heating and throughput... (SANGER)

- Small number of clones, Synchronized chemical reactions (pyrosequencing, pH measurements, etc.).
  *Problems*: Homopolymers, Phasing (fading, leading and lagging), etc. (Illumina, Ion Torrent, 454,...)
Variety of Sequencing Technologies

- Single Molecule...
  - Immobile Molecules: Problems: surface chemistry, sample preparation, sensor size and speed... (Optical Sequencing, SMASH, Helicos, Life Technology, PacificBio)
  - Mobile Molecules: Problems: nanopore detection, processivity, sample preparation, molecule size and speed... (Oxford Nanopore)
Next Generation

- Greedy algorithm is competitive (within a factor of four).
- Let $G$ be a weighted directed graph induced by the nodes representing the reads and edges representing the overlaps (with their weights determined by overhangs).

$$\text{CYC}(G) \leq \text{MWHC}(G) \leq \text{Opt}(S),$$

An optimal Cycle Cover has a smaller weight than that of a minimal weight Hamiltonian Cycle of the graph.

- Furthermore, if $T$ is the solution of the greedy algorithm then

$$\text{Opt}(S) \leq |T| \leq 3\text{CYC}(G) + \text{Opt}(S) \leq 4\text{Opt}(S).$$

Note: if the genome is completely random, then

$$|T| = (1 + o(1))\text{Opt}(S).$$

- Try to create better Cycle Cover. . . As few contigs as one can.
Higher Coverage ... Shorter Reads

Massively parallel sequencing platforms such as:
- Illumina, Inc. Genome Analyzer,
- Applied Biosystems SOLiD System, and
- 454 Life Sciences (Roche) GS FLX
- IonTorrent Sequencer

Features:
- Typical read size **35-500 bps**. Poorer quality base-calls
- Very **high coverage** (up to 200X).
- Need to assemble **millions of reads**!
Raw Coverage vs. Effective Coverage

- Raw coverage depth $c = \frac{LN}{G}$ where
  - $G =$ Genome length (in bp).
  - $L =$ Average length of a fragment.
  - $N =$ Number of fragments.
  - $K =$ # base pairs two fragments must have in common to ensure their overlap (overlap parameter).

- Effective coverage
  
  $c_e = \frac{N(L - K)}{G}$

*S. aureus*

- Raw coverage $c = \frac{LN}{G} = 48X$
- Effective coverage $c_e = \frac{N(L - K)}{G} = 14X$
Higher Effective Coverage

- As the read-length increases, the needed overlap ratio parameter gets smaller.
- Even better: shorter repeats and haplotype ambiguities become less of a problem.
- However, in order to get better quality in base calling, it becomes necessary to develop single-molecule methods that can detect chemistry at single base resolution.
- Cost of sample preparation and high resolution sensing!
- How about low-cost low resolution approaches:
  - Optical Mapping (Restn. or Nicking enz. or probes)
  - Mate-pairs (with multiple-length clones)
  - Strobed sequencing
  - Dilution
Optical Maps

- **Whole-Genome Optical Map**: Ordered Restriction Map; Markers are restriction sites; usually represented by an ordered sequence of restriction fragment length.

- Statistical algorithms (e.g., **Gentig**[AMS-1999] and **Haptig**[AM-2005]) construct accurate consensus map, even if the raw input suffers from many corrupting error processes (e.g., sizing, partial digestion, desorption, false-cuts, etc.)

- Gentig and Haptig integrate nicely with SUTTA assembler in a *technologically agnostic* manner.
Optical Maps

Step 1: *E. coli* microbial cells

Step 2 (left): Genomic DNA, captured as single DNA molecules produced by random breakage of intact chromosomes.
(right): Digestion reveals cleavage sites as "gaps."

Step 3: Fluorescent intensity is measured to determine fragment sizes while fragment order is maintained.

Overlapping single molecular maps are assembled to produce a highly accurate whole genome restriction map.
Optical Maps

Model Parameters

- \( k \) = # Cuts
- \( m \) = # Symmetric Cuts
- \( L \) = Length of the Clone in bps
- \( \Delta \) = Length of the Discretized Subintervals in bps
- \( \rho_t \) = Partial Digestion Rate
- \( \lambda_f \) = Spurious Cut Poisson Parameter
- \( \rho_e \) = Cutting Rate of the Enzyme

Theorem

Assume that the sizing error \( \sigma = 0 \).

Let \( \epsilon \) be a positive constant and \( c \geq 1 \) be so chosen that

\[
1 - e^{-32\sigma^2m^2} = \epsilon.
\]

Then for

\[
n \geq \frac{18}{\rho_c} \max \left( c + 2 \ln(k + m), \frac{c + \ln m}{\rho_c}, \frac{1}{\rho_c}, \frac{\ln \left( \frac{k}{k - \ln k - c} \right)}{\rho_c}, \frac{c + 2 \ln k}{\rho_c} \right),
\]

\( k > c + \ln k, m \geq 1, L > 2\Delta \) and \( \lambda_f < \rho_c L/5\Delta \), with probability at least \( 1 - \epsilon \), the correct ordered restriction map can be computed in \( O(nk^2) \) time.

When

\[
n < \max \left[ \ln(k + m), \frac{1}{\rho_c(1 + \rho_e)}, \frac{\ln m}{\rho_c}, \frac{\ln \frac{k}{k - 1}}{\rho_c}, \frac{\ln \left( \frac{L}{\Delta} \right)}{\rho_c} \right]
\]

\( k > 1, m > 1, L > \Delta \) and \( 0 < \rho_e < 0.69 \), no algorithm can compute the correct ordered restriction map with probability better than half.
Transcriptomics

- MMC (Molecular Morse Codes): Single Molecule Single Cell AFM-Based Transcriptome profiling
Wicked Problem

- The Sequence Assembly problem is an \( \mathcal{NP} \)-hard combinatorial optimization problem.
- The Sequence Assembly problem is claimed to have been successfully (but approximately) solved using greedy and heuristic methods; the greedy approaches exhibit many limitations and low flexibility.
- “Fast” Brute-Force global optimization of the sequence assembly problem is possible!
- SUTTA outperforms many assembly algorithms on bacterial genomes.
- SUTTA has the potential to assemble haplotypic whole-genome sequences.
- SUTTA is technology-agnostic: if the sequencing technology changes, just change the score function.
Regroup, Reformulate and Attack

Definition (Sequence Assembly Problem)

Given a collection of fragment/reads $F = \{r_i\}_{i=1}^N$ and a tolerance level (error rate) $\epsilon$, find a reconstruction $R$ whose layout $L$ is $\epsilon$-valid, consistent and such that the following set of properties (oracles) are satisfied:

- **Overlap-Constraint (O):** The cumulative overlap score of the layout is optimized.
- **Local-Read-Distribution-Constraint (R):** The observed distribution of fragment reads start point, $D_{obs}$, has the minimum deviation from the source distribution $D_{src}$.
- **Mate-Pair-Constraint (M):** The distance between mate-pairs is consistent.
- **Optical-Map-Constraint (OM):** The observed distribution of restriction enzyme sites, $C_{obs}$ is consistent with the distribution of experimental optical map data $C_{src}$.

**Constrained Optimization:** We suggest an approach that can combine and use all the oracles while searching the optimal layout.
Outline

1 Methods
   - SUTTA: Scoring-and-Unfolding Trimmed Tree Assembler
   - Algorithmic Improvements

2 Results
   - Assembly comparison

3 Conclusions and Discussions

Outline

Whole-Genome Shotgun Sequence Assembly
A Phylogeny of Assemblers
Assembly Paradigms

SUTTA: Scoring-and-Unfolding Trimmed Tree Assembler
Algorithmic Improvements
SUTTA’s approach

1. Could potentially lead to an exhaustive search over all possible overlays;
2. Tames the computational complexity through a constrained search (Branch-and-Bound) by identifying implausible overlays quickly;
3. Uses a score-function (*oracle*) combining different structural properties (e.g., transitivity, coverage, physical maps, etc).
First generate LEFT and RIGHT trees for the start read.

Next, the best LEFT path is concatenated with the root and the best RIGHT path to create a globally optimal contig.
Algorithm 2: SUTTA - pseudo code

Input: Set of $N$ reads
Output: Set of contigs

1. $\mathcal{F} := \emptyset$; /* Forest of D-trees */
2. $\mathcal{C} := \emptyset$; /* Set of contigs */
3. $B := \bigcup_{i=1}^{N} \{ r_i \}$; /* All the available reads */

while ($B \neq \emptyset$) do
1. $r := B$.getNextRead();
2. if $(!\text{isUsed}(r) \land !\text{isContained}(r))$ then
3. $\mathcal{D} := \text{create_double_tree}(r)$;
4. $\mathcal{F} := \mathcal{F} \cup \{ \mathcal{D} \}$;
5. $\text{Contig CTG} := \text{create_contig}(\mathcal{D})$;
6. $\mathcal{C} := \mathcal{C} \cup \{ \text{CTG} \}$;
7. $\text{CTG.layout}();$ /* Compute contig layout */
8. $B := B \setminus \{ \text{CTG.reads} \}$; /* Remove used reads */
else
9. /* jump to next available read */
end

end

return $\mathcal{C}$;
Node expansion
(High-level Description)

1. Start with a random read (It will be the root of a tree; Use only the read that has not been "used" in a contig yet, or that is not "contained").

2. Create RIGHT Tree: Start with an unexplored leaf node (a read) with the best score-value; Choose all its non-contained "right"-overlapping reads and expand the node by making them its children; Compute their scores. (Add the "contained" nodes along the way, while including them in the computed scores; Check that no read occurs repeatedly along any path of the tree). STOP when the tree cannot be expanded any further.

3. Create LEFT Tree: Symmetric to previous step.
Node expansion (Branch-and-Bound)

Algorithm 3: Node expansion

Input: Start read $r_0$, max queue size $K$, percentage $T$ of top ranking solutions
Output: Best scoring leaf

1. $T := \emptyset$; /* Set of leaves */
2. $L := \{(r_0, g(r_0))\}$; /* Live nodes (priority queue) */
3. while ($L \neq \emptyset$) do
4.   $L := \text{Prune}(L, K, T)$; /* Prune the queue */
5.   $r_i := L$.getNext(); /* Get the best scoring node */
6.   $L := L \setminus \{r_i\}$;
7.   if (no reads align with $r_i$) then
8.     $T := T \cup \{r_i\}$; /* $r_i$ is a leaf */
9.   else
10.    Add contained reads to $r_i$;
11.     for ($j=1$ to $M$) do
12.       $L := L \cup \{(r_i, g(r_i))\}$;
13.     end
14.   end
15. end
16. return $\max_{r_i \in T} \{g(r_i)\}$;
Overlap Score
(Weighted transitivity)

**Idea**: if read $A$ overlaps read $B$, and read $B$ overlaps read $C$, we will score those overlaps strongly if in addition $A$ and $C$ also overlap. This implicitly assumes that the coverage is higher than 3.

$$\text{if}(\pi(A,B) \land \pi(B,C)) \text{then} \{S_{\pi(A,B,C)} = S_{\pi(A,B)} + S_{\pi(B,C)} + (\pi(A,C) \cdot S_{\pi(A,C)} : 0) \}$$ (4)

A simple generalization for higher coverage is obvious.

This score cannot resolve repeats or haplotype variations. Solution: augment the score with information for mate-pairs distances or optical map alignment to put an appropriate reward/penalty term.
Strategy for selecting next sub-problem

**Problem:** trade-off between keeping the number of explored nodes in the search tree low and staying within the memory capacity.

- **Best First Search (BeFS):** always select among the live subproblems the one with best score.
  1. Memory problems can arise since this strategy behaves similarly to a Breadth First Search (BFS).
  2. Checking repeated nodes in a branch is computationally expensive (linear time).
  3. Theoretically superior: whenever a node is chosen for expansion, a best-score path to that node has been found.

- **Depth First Search (DFS):** always select among the live subproblems the one with largest level in the search tree.
  1. Memory requirements are bounded by $depth \times branching$.
  2. Use depth-first search interval schemes to check if a read occurs repeatedly along a path (constant time).

**Solution:** combined strategy. DFS + BeFS.
Transitivity pruning

- **Observation**: do not waste time expanding nodes that create suffix-paths of a previously created path.
- **Idea**: delay expansion of the "last" node/read involved in a transitivity relation.

The expansion of nodes $B_2, B_3, \ldots, B_n$ can be delayed because their overlap with read $A$ is enforced by read $B_1$ ($h_1 \leq h_2 \leq \cdots \leq h_n$).
Lookahead
Using Long Range Information

- **Scenario**: A potential repeat boundary between reads A, B and C. Read A overlaps both reads B and C, but B and C do not overlap each other.
- **Observation**: No decision can be made at this point on which read to keep/prune.
- **Idea**: Chose between reads A and B based on how well the mate-pairs in their subtree satisfy the length constraints.

![Diagram showing the relationship between reads A, B, and C, with a potential repeat boundary between A and B/C.](attachment:image.png)
**Staphylococcus Epidermidis** - 2,616,530 bp

(SUTTA DotPlot)

- (left plot) no lookahead; (right plot) with lookahead.

Num. of reads: 60, 761; Avg read length: 900.2; Coverage: 19.9X
Hard instances (SUTTA DotPlot)

- Chromosome 22 - 5Mbp (simulated reads)

Num. of reads: 62,542; Avg read length: 799.5; Coverage: 10X
The Sequence Assembly problem is an $\mathcal{NP}$-hard combinatorial optimization problem.

The Sequence Assembly problem is claimed to have been successfully (but approximately) solved using greedy and heuristic methods; the greedy approaches exhibit many limitations and low flexibility.

“Fast” Brute-Force global optimization of the sequence assembly problem is possible!

SUTTA outperforms many assembly algorithms on bacterial genomes.

SUTTA has the potential to assemble haplotypic whole-genome sequences.

SUTTA is technology-agnostic: if the sequencing technology changes, just change the score function.
Outline

- Whole-Genome Shotgun Sequence Assembly
- A Phylogeny of Assemblers
- Assembly Paradigms

1 Methods
- SUTTA: Scoring-and-Unfolding Trimmed Tree Assembler
- Algorithmic Improvements

2 Results
- Assembly comparison

3 Conclusions and Discussions
Benchmark data

<table>
<thead>
<tr>
<th>Genome</th>
<th>Length (bp)</th>
<th># reads</th>
<th>avg. read length (bp)</th>
<th>Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brucella S.</td>
<td>3,315,173</td>
<td>36,276</td>
<td>895.8</td>
<td>9.8</td>
</tr>
<tr>
<td>Wolbachia Sp.</td>
<td>1,267,782</td>
<td>26,817</td>
<td>981.9</td>
<td>20.7</td>
</tr>
<tr>
<td>Staphylococcus E.</td>
<td>2,616,530</td>
<td>60,761</td>
<td>900.2</td>
<td>19.9</td>
</tr>
</tbody>
</table>

Table: Bacteria benchmark data.

These bacteria have been sequenced and fully finished at TIGR, and all the sequencing reads generated for these projects are publicly available at both the NCBI Trace Archive, and from the CBCB website\(^1\).

\(^1\)www.cbcb.umd.edu/research/benchmark.shtml
## Assembly comparison

<table>
<thead>
<tr>
<th>Genome</th>
<th>Assembler</th>
<th># contigs</th>
<th># big contigs (&gt;10 kbp)</th>
<th>Max contig size (kbp)</th>
<th>Mean big contig size (kbp)</th>
<th>N50(^2) (kbp)</th>
<th>Big contig coverage (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brucella Suis</td>
<td>Minimus</td>
<td>203</td>
<td>101</td>
<td>89</td>
<td>30</td>
<td>32</td>
<td>93.1</td>
</tr>
<tr>
<td></td>
<td>TIGR</td>
<td>108</td>
<td>67</td>
<td>182</td>
<td>48</td>
<td>57</td>
<td>98.8</td>
</tr>
<tr>
<td></td>
<td>CAP3</td>
<td>1321</td>
<td>35</td>
<td>18</td>
<td>12</td>
<td>4</td>
<td>12.7</td>
</tr>
<tr>
<td></td>
<td>Euler</td>
<td>238</td>
<td>108</td>
<td>82</td>
<td>25</td>
<td>25</td>
<td>82.2</td>
</tr>
<tr>
<td></td>
<td>Phrap</td>
<td>54</td>
<td>23</td>
<td>434</td>
<td>126</td>
<td>199</td>
<td>103.2</td>
</tr>
<tr>
<td></td>
<td>SUTTA(^c)</td>
<td>73</td>
<td>53</td>
<td>268</td>
<td>62</td>
<td>79</td>
<td>99.2</td>
</tr>
<tr>
<td></td>
<td>SUTTA(^a)</td>
<td>73</td>
<td>45</td>
<td>396</td>
<td>72</td>
<td>98</td>
<td>98.4</td>
</tr>
<tr>
<td>Brucella Suis</td>
<td>Wolbachia Sp.</td>
<td>Minimus</td>
<td>1545</td>
<td>37</td>
<td>16</td>
<td>13</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>TIGR</td>
<td>1080</td>
<td>46</td>
<td>46</td>
<td>20</td>
<td>5</td>
<td>73.6</td>
</tr>
<tr>
<td></td>
<td>CAP3</td>
<td>1661</td>
<td>1</td>
<td>10</td>
<td>10</td>
<td>2</td>
<td>0.7</td>
</tr>
<tr>
<td></td>
<td>Euler</td>
<td>615</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Phrap</td>
<td>2253</td>
<td>55</td>
<td>64</td>
<td>22</td>
<td>1.8</td>
<td>98.5</td>
</tr>
<tr>
<td></td>
<td>SUTTA(^c)</td>
<td>1089</td>
<td>39</td>
<td>87</td>
<td>26</td>
<td>6</td>
<td>80.8</td>
</tr>
<tr>
<td></td>
<td>SUTTA(^a)</td>
<td>1068</td>
<td>27</td>
<td>181</td>
<td>39</td>
<td>6</td>
<td>83.5</td>
</tr>
<tr>
<td>Wolbachia Sp.</td>
<td>Minimus</td>
<td>425</td>
<td>86</td>
<td>119</td>
<td>10</td>
<td>19</td>
<td>80.7</td>
</tr>
<tr>
<td></td>
<td>TIGR</td>
<td>94</td>
<td>38</td>
<td>230</td>
<td>68</td>
<td>100</td>
<td>99.8</td>
</tr>
<tr>
<td></td>
<td>CAP3</td>
<td>1219</td>
<td>39</td>
<td>21</td>
<td>13</td>
<td>5</td>
<td>20.2</td>
</tr>
<tr>
<td></td>
<td>Euler</td>
<td>116</td>
<td>54</td>
<td>149</td>
<td>44</td>
<td>55</td>
<td>91.5</td>
</tr>
<tr>
<td></td>
<td>Phrap</td>
<td>86</td>
<td>22</td>
<td>357</td>
<td>123</td>
<td>183</td>
<td>103.9</td>
</tr>
<tr>
<td></td>
<td>SUTTA(^c)</td>
<td>65</td>
<td>33</td>
<td>268</td>
<td>78</td>
<td>98</td>
<td>98.7</td>
</tr>
<tr>
<td></td>
<td>SUTTA(^a)</td>
<td>64</td>
<td>24</td>
<td>756</td>
<td>108</td>
<td>148</td>
<td>99.1</td>
</tr>
</tbody>
</table>

\(^2\)N50 = length \(L_c\) of the largest contig such that the sum of contigs of equal length or longer is at least 50% of the total length of all contigs.

---

B Mishra

Detectability of Certain Dark-Genome-Matter Candidates
Brucella Suis - 2 chromosomes of 2,107,792 and 1,207,381 bp
(Minimus DotPlot)

Minimus’s conservative strategy fails to create long contigs.

Num. of reads: 36,276; Avg read length: 895.8; Coverage: 9.8X
Brucella Suis - 2 chromosomes of 2,107,792 and 1,207,381 bp (Phrap DotPlot)

Phrap’s aggressive strategy creates many mis-assemblies.

Num. of reads: 36,276; Avg read length: 895.8; Coverage: 9.8X
Brucella Suis - 2 chromosomes of 2,107,792 and 1,207,381 bp
(SUTTA DotPlot)

Num. of reads: 36,276; Avg read length: 895.8; Coverage: 9.8X
Staphylococcus Epidermidis - 2,616,530 bp
(TIGR DotPlot)

TIGR’s greedy strategy fails to join some of the contigs and produces few mis-assemblies.

Num. of reads: 60,761; Avg read length: 900.2; Coverage: 19.9X
Staphylococcus Epidermidis - 2,616,530 bp
(Phrap DotPlot)

Phrap's greedy strategy fails to join some of the contigs and produces many mis-assemblies.

Num. of reads: 60,761; Avg read length: 900.2; Coverage: 19.9X
Staphylococcus Epidermidis - 2,616,530 bp
(SU TT A DotPlot)

Num. of reads: 60,761; Avg read length: 900.2; Coverage: 19.9X
Too often assemblies have been judged only by contig size, with larger contigs preferred without regard to quality. A new and more reliable metric needs to be devised.

Inspired by the standard receiver operating characteristic (ROC) curve, the Feature-Response curve characterizes the sensitivity (coverage) of the sequence assembler as a function of its discrimination threshold (number of features/errors).

Features include:

- \((M)\) mate-pair orientations and separations,
- \((K)\) repeat content by \(k\)-mer analysis,
- \((C)\) depth-of-coverage,
- \((P)\) correlated polymorphism in the read alignments, and
- \((B)\) read alignment breakpoints to identify structurally suspicious regions of the assembly.
For a fixed feature threshold $\phi$, the contigs are sorted by size and, starting from the longest, only those contigs are tallied, if their sum of features is $\leq \phi$.

For this set of contigs, the corresponding genome coverage is computed, leading to a single point of the Feature-Response curve.
The Sequence Assembly problem is an \(NP\)-hard combinatorial optimization problem. The Sequence Assembly problem is claimed to have been successfully solved using greedy and heuristic methods; the greedy approaches exhibit many limitations and low flexibility.

“Fast” Brute-Force global optimization of the sequence assembly problem is possible!

SUTTA outperforms many assembly algorithms on bacterial genomes.

SUTTA has the potential to assemble haplotypic whole-genome sequences.

SUTTA is technology-agnostic: if the sequencing technology changes, just change the score function.
Short-Read Overlapper

Idea: use exact matching
- allowing approximate matching (using dynamic programming) would significantly increase the number of nonspecific spurious overlaps (sequencing errors)
- drastically faster than approximate matching

Implementation: Trie data structure (prefix-tree)
- index the non-redundant read data set by a prefix-tree (both forward and reverse complement).
- find overlaps by simple in-order traversal of the tree.
Short-Read Overlapper

Set of overlapping reads

Prefix-Tree

A

i

j

K

B

B Mishra

Detectability of Certain Dark-Genome-Matter Candidates
Comparison of assemblies
short read data

- Only contigs ≥ 100 bp.
- Correct contig (S. aureus): aligned along its whole length with at least 98% base similarity.
- Correct contig (E. coli): fewer than 5 consecutive base mismatches at the termini and at least 95% base similarity.

<table>
<thead>
<tr>
<th>Genome</th>
<th>Assembler</th>
<th># correct</th>
<th># misassembled</th>
<th>N50</th>
<th>Mean</th>
<th>Max</th>
<th>Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>(mean)</td>
<td>(kbp)</td>
<td>(kbp)</td>
<td>(kbp)</td>
<td>(%)</td>
</tr>
<tr>
<td><strong>S. aureus</strong></td>
<td>SUTTA</td>
<td>998</td>
<td>11</td>
<td>6.0</td>
<td>2.6</td>
<td>22.8</td>
<td>97</td>
</tr>
<tr>
<td></td>
<td>Edena (strict)</td>
<td>1122</td>
<td>0</td>
<td>6.0</td>
<td>2.6</td>
<td>25.7</td>
<td>98</td>
</tr>
<tr>
<td></td>
<td>Edena (nonstrict)</td>
<td>733</td>
<td>14</td>
<td>9.4</td>
<td>3.7</td>
<td>51.8</td>
<td>97</td>
</tr>
<tr>
<td></td>
<td>Velvet</td>
<td>1093</td>
<td>2</td>
<td>5.4</td>
<td>2.5</td>
<td>22.9</td>
<td>98</td>
</tr>
<tr>
<td></td>
<td>SSAKE</td>
<td>2334</td>
<td>99</td>
<td>2.0</td>
<td>1.2</td>
<td>12.6</td>
<td>97</td>
</tr>
<tr>
<td></td>
<td>SHARCGS</td>
<td>3632</td>
<td>3</td>
<td>1.2</td>
<td>0.76</td>
<td>8.6</td>
<td>97</td>
</tr>
<tr>
<td><strong>E. coli</strong></td>
<td>SUTTA</td>
<td>423</td>
<td>7 (18.8)</td>
<td>22.7</td>
<td>10.2</td>
<td>84.5</td>
<td>98</td>
</tr>
<tr>
<td>(K12 MG1655)</td>
<td>ABySS</td>
<td>220</td>
<td>13 (33.2)</td>
<td>45.3</td>
<td>20.2</td>
<td>173.8</td>
<td>99</td>
</tr>
<tr>
<td></td>
<td>Edena (strict)</td>
<td>674</td>
<td>6 (13.2)</td>
<td>16.4</td>
<td>6.6</td>
<td>67.1</td>
<td>99</td>
</tr>
<tr>
<td></td>
<td>Velvet</td>
<td>277</td>
<td>9 (52.3)</td>
<td>54.3</td>
<td>15.9</td>
<td>164.2</td>
<td>98</td>
</tr>
<tr>
<td></td>
<td>SSAKE</td>
<td>893</td>
<td>38 (5.8)</td>
<td>11.4</td>
<td>4.9</td>
<td>50.6</td>
<td>99</td>
</tr>
<tr>
<td></td>
<td>EULER-SR</td>
<td>190</td>
<td>26 (37.8)</td>
<td>57.4</td>
<td>21.1</td>
<td>174.0</td>
<td>99</td>
</tr>
<tr>
<td></td>
<td>SOAPdenovo</td>
<td>182</td>
<td>5 (n.a.)</td>
<td>89.0</td>
<td>25.0</td>
<td>n.a.</td>
<td>n.a.</td>
</tr>
</tbody>
</table>

N50 = the largest number $L$ such that the combined length of all contigs of length $\geq L$ is at least 50% of the total length of all contigs.
N50 vs. min overlap size $k$

- The min overlap length $k$ is a determinant parameter and its optimal setting strongly depends on the data (coverage).
- Trade-off between number of spurious overlaps and lack of overlaps.

![Graphs showing N50 vs. min overlap size $k$](image)
Outline

- Whole-Genome Shotgun Sequence Assembly
- A Phylogeny of Assemblers
- Assembly Paradigms

1. Methods
   - SUTTA: Scoring-and-Unfolding Trimmed Tree Assembler
   - Algorithmic Improvements

2. Results
   - Assembly comparison

3. Conclusions and Discussions
Wicked Problem

- The Sequence Assembly problem is an $\mathcal{NP}$-hard combinatorial optimization problem.
- The Sequence Assembly problem is claimed to have been successfully solved using greedy and heuristic methods; the greedy approaches exhibit many limitations and low flexibility.
- “Fast” Brute-Force global optimization of the sequence assembly problem is possible!
- SUTTA outperforms many assembly algorithms on bacterial genomes.
- SUTTA has the potential to assemble haplotypic whole-genome sequences.
- SUTTA is technology-agnostic: if the sequencing technology changes, just change the score function.
Genomics Redux

1. Whole-Genome Haplotypic Sequence Assembly:
   - **Cost:** High-Throughput Short Reads and Low-Resolution Single Molecule Maps
   - Methylation-Sensitive Restriction Maps
   - Epigenetics, Rare variants, de novo mutations, structural variants, and much more... (with haplotype phasing)

2. 4,000 – 10,000 Haplotypic References from a Well-sampled Population
   - Distribution of de novo mutations
   - Out-of-Africa (North-West Africa, Arabian Peninsula, Southern and South-East Asia, Australasia)
   - Indian Subcontinent

3. Characterization of Genomic and de novo Variants, Selective Sweeps and Population Dynamics

4. **Phenotyping!!** [Causality Analysis...]

B Mishra

Detectability of Certain Dark-Genome-Matter Candidates
Puzzle: A shotgun assembly of few words.

WordList: “assembled,” “completely,” “correct,” “genome,” “human,” “in-,” “is,” “only,” “sequence,” and “the.”

WordAssembly: “the human genome sequence is correct, only incompletely assembled;”

Other Solutions: “the only assembled human genome sequence is completely incorrect;”

Other Solutions: “only, correct the assembled sequence; genome is completely inhuman.”
Puzzle: A shotgun assembly of few words.

**WordList**: “assembled,” “completely,” “correct,” “genome,” “human,” “in-,” “is,” “only,” “sequence,” and “the.”

**WordAssembly**: “the human genome sequence is correct, only incompletely assembled;”

**Other Solutions**: “the only assembled human genome sequence is completely incorrect;”

**Other Solutions**: “only, correct the assembled sequence; genome is completely inhuman.”
Puzzle: A shotgun assembly of few words.

**WordList**: “assembled,” “completely,” “correct,” “genome,” “human,” “in-,” “is,” “only,” “sequence,” and “the.”

**WordAssembly**: “the human genome sequence is correct, only incompletely assembled;”

**Other Solutions**: “the only assembled human genome sequence is completely incorrect;”

**Other Solutions**: “only, correct the assembled sequence; genome is completely inhuman.”
Puzzle: A shotgun assembly of few words.

WordList: “assembled,” “completely,” “correct,” “genome,” “human,” “in-,” “is,” “only,” “sequence,” and “the.”

WordAssembly: “the human genome sequence is correct, only incompletely assembled;”

Other Solutions: “the only assembled human genome sequence is completely incorrect;”

Other Solutions: “only, correct the assembled sequence; genome is completely inhuman.”