

One Thousand and One Arabian Genomes & Beyond

A Human Journey

Bud Mishra

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

Jointly with *Giuseppe Narzisi* (Graduate Student, Courant)

A Human Journey

The lower valley of the Omo (in southwest Ethiopia) is believed to be the birth place of modern humans.

- Fossils belonging to the genera *Australopithecine* and *Homo* have been found at several archaeological sites here as well as tools made from quartzite, the oldest of which date back to about 2.4 million years ago.
- This valley has the largest diversity of ethnically different groups in the whole of Ethiopia and possibly in Africa. There are more than 16 different languages spoken (excluding dialects), belonging to two important linguistic lines: *Nilo-Saharan* and *Afro-Asiatic*.

Bab-El-Mandeb

From northern Africa, 125,000 years ago, a group travelled across a green Sahara through the open northern gate, up to the Nile to the Levant. The branch that reached the Levant died out by 90,000 years ago during a global freeze-up.

This region was later reoccupied by Neanderthal Man.

Later 85,000 years ago, another group crossed the mouth of the Red Sea — “Bab-El-Mandeb” (“the Gates of Grief”)— prior to travelling as beach-combers along the southern coast of the Arabian Peninsula toward India.

This group is our ancestral population.

Out of Africa

From our genomes, we can reconstruct this history...

But also, our complex traits, genetic diseases, and future of our species...

And how we can/should shape our collective destiny!

- North-East Africa, Arabian Peninsula, Southern and South-East Asia, and Australasia...
- Rift Valley, Yemen, Oman, Emirates, Jharkhand (India), Andaman & Nicobar (India), Srilanka, Melanesia & Australia.

[WGS → GWAS] ; (

THE HOPE

The Logic behind Genome Studies

Much research into the genetic contributions to common diseases has started with the seemingly logical assumption that DNA variants occurring frequently in the human population would be at fault. Some argue, though, that this reasoning is faulty.

The Starting Point
The Human Genome Project identified the location of nucleotide pairs, or DNA building blocks, in the human genome, based on DNA from several volunteers. A single pair consists of a nucleotide (A, C, T, or G) on one strand of the DNA double helix and its complement on the opposite strand (C/G pairs with G/C, and T/A with A/T). Initial work revealed many single-nucleotide polymorphisms, or SNPs—chromosomal locations where a nucleotide pair in one person can differ from that in another person (below)—and it identified “common” SNPs, ones that vary in many people.

The human genome (or complete set of DNA) is contained within 22 autosomal chromosomes plus the X and Y. We inherit one set of 23 chromosomes from each parent.

The Starters and Results
Investigators hoped that they could identify gene variants responsible for major diseases by comparing nucleotides at common SNPs throughout the genomes of people with and without a disease. SNP variants, or “alleles,” and nearby protein-coding genes tend to be inherited together, and so researchers expected that SNP alleles occurring much more frequently in people with a disease would point to common gene variants important to the illness. These genome-wide association (GWA) studies uncovered many SNP alleles related to specific diseases. So far, though, the variations found have typically accounted for only a small fraction of disease risk.

SNP allele associated with disease	Nearby gene variant	Health Status
Yellow	Orange	Diseased
Yellow	Orange	Diseased
Yellow	Orange	Diseased
Blue	Orange	Healthy
Blue	Orange	Healthy

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

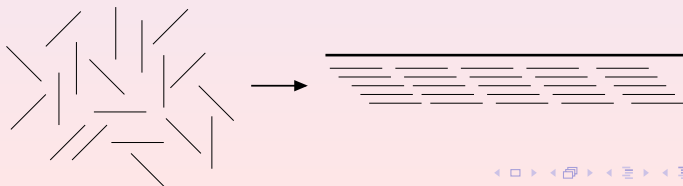
Genomic Dark Matter

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 resquenced reads not aligning to the reference sequences: **Genomic Dark Matter**.
- **Optical Mapping**, a highthroughput, 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.

Shotgun Sequencing

- 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*.
- **Implicit Assumption:** 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.



Greed is Good

Subprime Sequencing

- **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).

Name	Algorithm	Author	Year
Arachne WGA	OLC	Batzoglou, S. et al.	2002 / 2003
Celera WGA Assembler / CABOG	OLC	Myers, G. et al.; Miller G. et al.	2004 / 2008
Minimus (AMOS)	OLC	Sommer, D.D. et al.	2007
Newbler	OLC	454/Roche	2009
Edena	OLC	Hernandez D., et al.	2008
SUTTA	B&B	NYU/Abraxis (unpublished)	2009/2010
TIGR	Greedy	TIGR	1995 / 2003
Phusion	Greedy	Mullikin JC, et.al.	2003
Phrap	Greedy	Green, P.	2002 / 2003 / 2008
CAP3, PCAP	Greedy	Huang, X. et al.	1999 / 2005
Euler	SBH	Pevzner, P. et al.	2001 / 2006
Euler-SR	SBH	Chaisson, MJ. et al.	2008
Velvet	SBH	Zerbino, D. et al.	2007 / 2009
ALLPATHS	SBH	Butler, J. et al.	2008
ABYSS	SBH	Simpson, J. et al.	2008 / 2009
SOAPdenovo	SBH	Ruiqiang Li, et al.	2009
SHARCGS	Prefix-Tree	Dohm et al.	2007
SSAKE	Prefix-Tree	Warren, R. et al.	2007
VCAKE	Prefix-Tree	Jeck, W. et al.	2007
QSRA	Prefix-Tree	Douglas W. et al.	2009
Sequencher	-	Gene Codes Corporation	2007
SeqMan NGen	-	DNASTAR	2008
Staden gap4 package	-	Staden et al.	1991 / 2008
MIRA, miraEST	-	Chevreur, B.	1998 / 2008
NextGENe	-	Softgenetics	2008
CLC Genomics Workbench	-	CLC bio	2008 / 2009
CodonCode Aligner	-	CodonCode Corporation	2003 / 2009

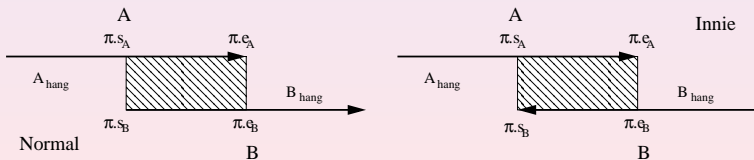
Fragments and Overlaps

Fragments:

- A set of fragments/reads $F = \{r_1, r_2, \dots, r_N\}$, 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 $F = \{r_1, r_2, \dots, r_N\}$ as follows:

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

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 $i = 2, \dots, N - 1$:

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

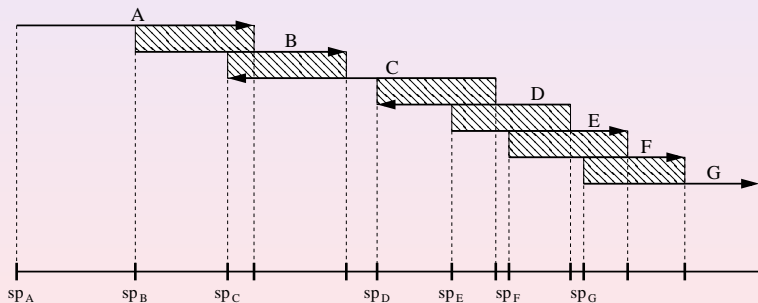
- The estimated start positions for each fragment are given by:

$$sp_1 = 1, \quad sp_i = sp_{i-1} + \pi_{i-1} \cdot \text{hang}_{r_{i-1}} \quad \text{if } i > 1 \quad (3)$$

Layout Representation

(Example)

Layout for a set of fragments $F = \{A, B, C, D, E, F, G\}$ with a sequence of overlaps $\pi_{(A,B)}^N, \pi_{(B,C)}^I, \pi_{(C,D)}^N, \pi_{(D,E)}^I, \pi_{(E,F)}^N, \pi_{(F,G)}^N$



Toxic Assembly Recovery Program

Shortest Common Superstring Problem

Definition (Sequence Assembly Problem)

Given a collection of fragment/reads $F = \{r_i\}_{i=1}^N$ and a tolerance level (error rate) ϵ , find a reconstruction R whose layout L is ϵ -**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.

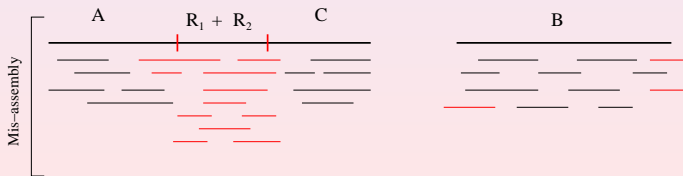
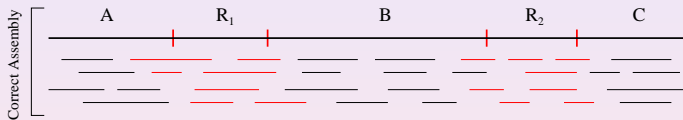
Closely Related to a formulation of assembly in terms of **SCSP**:
Shortest Common Superstring Problem...

NP-completeness of this formulation becomes a serious Issue.

Problems Related to Genome Structure

Repeats

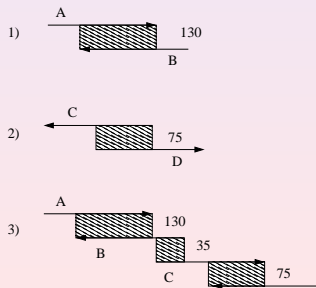
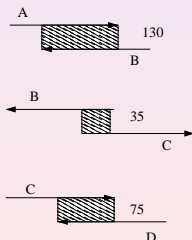
- If we look for a reconstruction of minimum length, the reconstructed string can have many errors due to *repeats*.



Greedy Strategy

(TIGR 1995, Phrap 1996, CAP3 1999)

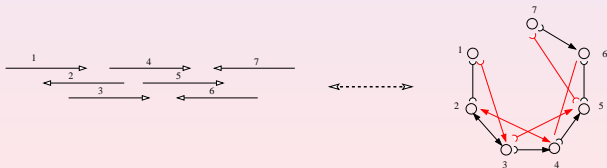
- Pick the highest scoring overlap.
- Merge the two fragments (add this new sequence to the pool of sequences).
- Heuristically correct regions of the overlay in some plausible manner (whenever possible).
- Regions that do not yield to these error-correction heuristics are abandoned as irrecoverable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



Overlap-Layout-Consensus

(CELERA 2000, Minimus 2007)

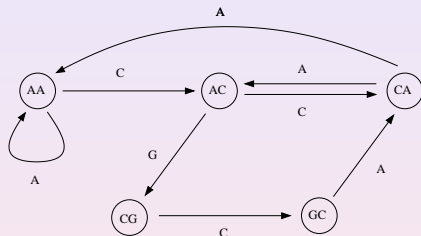
- **Idea:** Construct a graph in which nodes represent reads and edges indicate overlaps.
- **Goal:** Need to solve for a Hamiltonian path !
- **Strategy:**
 - Remove contained and transitivity edges.
 - Collapse "unique connector" overlaps (chordal subgraph with no conflicting edges).
 - Use mate-pairs 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) \rightarrow 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:
 $AC \rightarrow CA \rightarrow AC \rightarrow CG \rightarrow GC \rightarrow CA \rightarrow AA \rightarrow AC$

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!**

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.
- 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

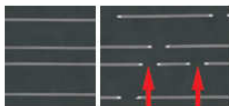
Optical Maps

- **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

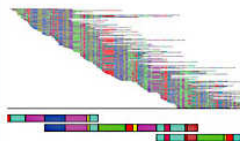


Step 2 (left): Genomic DNA, captured as single DNA molecules produced by random breakage of intact chromosomes.

(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.

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!

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!*

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.

SCSP(= ShortestCommonSuperstringProb)

- ↳ HAM(TSP = TravlingSalesmanProblem)
- ↳ SAT(= Satisfiability)
- ↳ LatinSquare
- ↳ Sudoku

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.

How to Cope with NP-Completeness

- 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

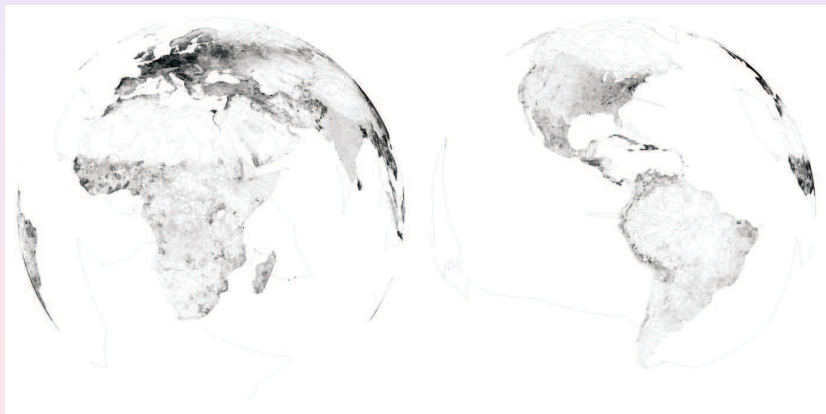
A traveling salesman wishes to visit a given number of cities, returning to the starting point in a tour. Each pair of cities incurs a cost proportional to the distance between these cities. The Traveling Salesman Problem (TSP) is to find a tour with minimum costs.

1994	Applegate, Bixby, Chvátal, Cook	7,397
1998	Applegate, Bixby, Chvátal, Cook	13,509 (USA tour)
2001	Applegate, Bixby, Chvátal, Cook	15,112 (D tour)
2004	App., Bixby, Chvátal, Cook, Helsgaun	24,978 (Swe tour)
2006	App., Bixby, Chvátal, Cook, Helsgaun	85,900

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

- Davis-Putnam-Logemann-Loveland algorithm (DPLL)
- 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.

k -SAT

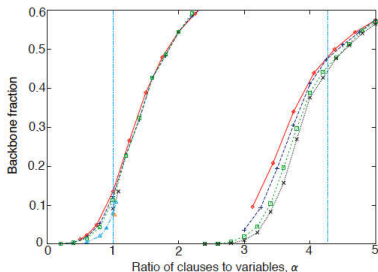


Figure 4 Backbone fractions as a function of α for 2-SAT and 3-SAT. The data were extracted from 3,000–15,000 samples for the 2-SAT cases (on the left) with N values of 20 (red), 30 (blue), 45 (green), 100 (black), 200 (light blue), and 500 (orange). The results for $N \leq 45$ were obtained by exhaustive 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 $N \rightarrow \infty$. For 2-SAT, data obtained from larger sizes show that the backbone fraction at the threshold tends towards zero.

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) ϵ , find a reconstruction R whose layout L is ϵ -**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

- Whole-Genome Shotgun Sequence Assembly
- An Explosion Assemblers
- Assembly Paradigms

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

- 2 **Results**
 - Assembly comparison

- 3 **Conclusions and Discussions**

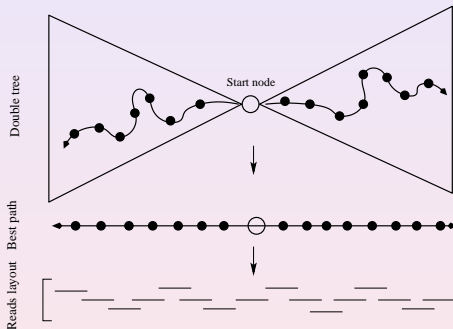
NP-Easy

De Novo Genome Assembly

- 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).

SUTTA

Illustration



- 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.

SUTTA

Pseudocode

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  $\mathcal{B} := \bigcup_i^N \{r_i\};$                     /* All the available reads */
4 while ( $\mathcal{B} \neq \emptyset$ ) do
5      $r := \mathcal{B}.\text{getNextRead}();$ 
6     if ( $\neg \text{isUsed}(r) \ \&\& \ \neg \text{isContained}(r)$ ) then
7          $DT := \text{create\_double\_tree}(r);$ 
8          $\mathcal{F} := \mathcal{F} \cup \{DT\};$ 
9          $\text{Contig } CTG := \text{create\_contig}(DT);$ 
10         $\mathcal{C} := \mathcal{C} \cup \{CTG\};$ 
11         $CTG.\text{layout}();$  /* Compute contig layout */
12         $\mathcal{B} := \mathcal{B} \setminus \{CTG.\text{reads}\};$  /* Remove used reads */
13    else
14        /* jump to next available read */
15    end
16 end
17 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  $\mathcal{L} := \{(r_0, g(r_0))\};$  /* Live nodes (priority queue) */
3 while ( $\mathcal{L} \neq \emptyset$ ) do
4    $\mathcal{L} := \text{Prune}(\mathcal{L}, K, T);$            /* Prune the queue */
5    $r_i := \mathcal{L}.\text{getNext}();$  /* Get the best scoring node */
6    $\mathcal{L} := \mathcal{L} \setminus \{r_i\};$ 
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    /* Branch on  $r_i$  generating  $r_{i_1}, r_{i_2}, \dots, r_{i_M}$  */
12    for ( $j=1$  to  $M$ ) do
13       $\mathcal{L} := \mathcal{L} \cup \{(r_{ij}, g(r_{ij}))\};$ 
14    end
15  end
16 return  $\max_{r_i \in T} \{g(r_i)\};$ 

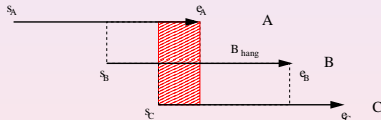
```

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.

$$\text{if } (\pi_{(A,B)} \wedge \pi_{(B,C)}) \text{ then } \{ S_{\pi_{(A,B,C)}} = S_{\pi_{(A,B)}} + S_{\pi_{(B,C)}} + (\pi_{(A,C)} ? S_{\pi_{(A,C)}} : 0) \} \quad (4)$$



- A simple generalization for higher coverage is obvious.
- This score cannot resolve repeats or haplotypic 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

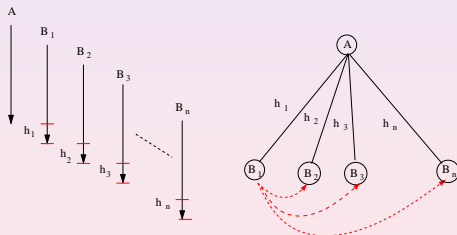
Problem: trade-off between keeping the number of explored nodes in the search tree low and staying within the memory capacity.

- **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.

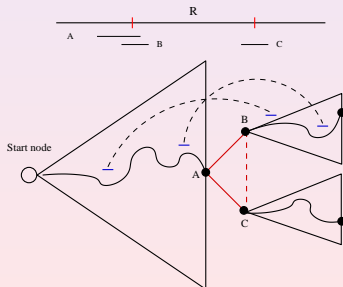


The expansion of nodes B_2, B_3, \dots, B_n can be delayed because their overlap with read A is enforced by read B_1 ($h_1 \leq h_2 \leq \dots \leq h_n$).

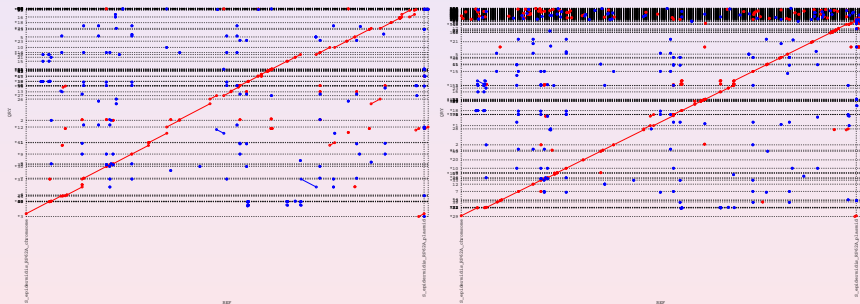
Lookahead

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.



Staphylococcus Epidermidis - 2,616,530 bp (SUTTA DotPlot)



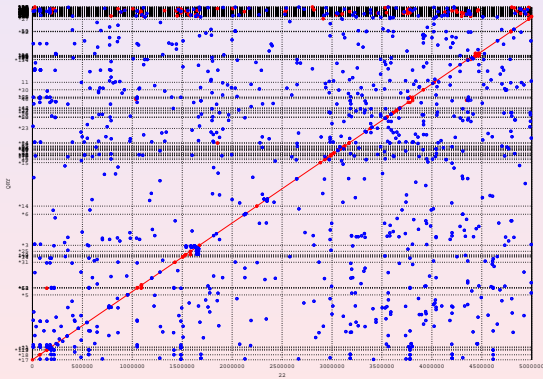
- (left plot) no lookahead; (right plot) with lookahead.

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

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.

Outline

- Whole-Genome Shotgun Sequence Assembly
- An Explosion Assemblers
- **Assembly Paradigms**

1 Methods

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

2 Results

- Assembly comparison

3 Conclusions and Discussions

Benchmark data

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

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 website¹.

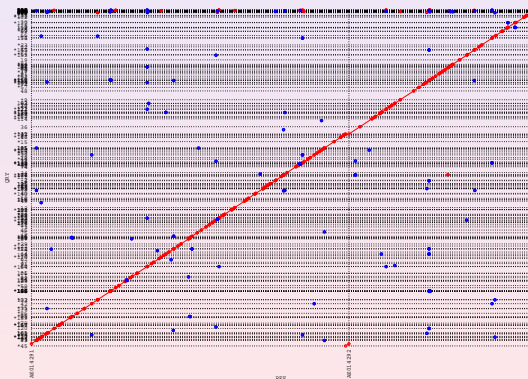
¹www.cbcb.umd.edu/research/benchmark.shtml

Genome	Assembler	# contigs	# big contigs (> 10 kbp)	Max contig size (kbp)	Mean big contig size (kbp)	N50 ² (kbp)	Big contig coverage (%)
<i>Brucella Suis</i>	Minimus	203	101	89	30	32	93.1
	TIGR	108	67	182	48	57	98.8
	CAP3	1321	35	18	12	4	12.7
	Euler	238	108	82	25	25	82.2
	Phrap	54	23	434	126	199	103.2
	SUTTA ^c	73	53	268	62	79	99.2
	SUTTA ^a	73	45	396	72	98	98.4
<i>Wolbachia Sp.</i>	Minimus	1545	37	16	13	2	40.7
	TIGR	1080	46	46	20	5	73.6
	CAP3	1661	1	10	10	2	0.7
	Euler	615	0	6	0	1	0
	Phrap	2253	55	64	22	1.8	98.5
	SUTTA ^c	1089	39	87	26	6	80.8
	SUTTA ^a	1068	27	181	39	6	83.5
<i>Staphylococcus Epidermidis</i>	Minimus	425	86	119	10	19	80.7
	TIGR	94	38	230	68	100	99.8
	CAP3	1219	39	21	13	5	20.2
	Euler	116	54	149	44	55	91.5
	Phrap	86	22	357	123	183	103.9
	SUTTA ^c	65	33	268	78	98	98.7
	SUTTA ^a	64	24	756	108	148	99.1

²N50 = 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.

Brucella Suis - 2 chromosomes of 2,107,792 and 1,207,381 bp (Minimus DotPlot)

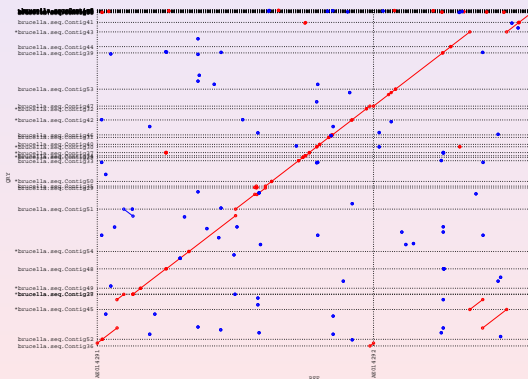
Minimus's conservative strategy fails to create long contigs.



Num. of reads: 36, 276; Avg read length: 895.8; Coverage: 9.8X

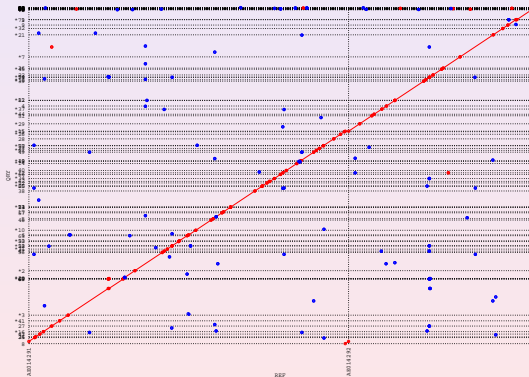
Brucella Suis - 2 chromosomes of 2,107,792 and 1,207,381 bp (Phrap DotPlot)

Phrap's aggressive strategy creates many mis-assemblies.



Num. of reads: 36, 276; Avg read length: 895.8; Coverage: 9.8X

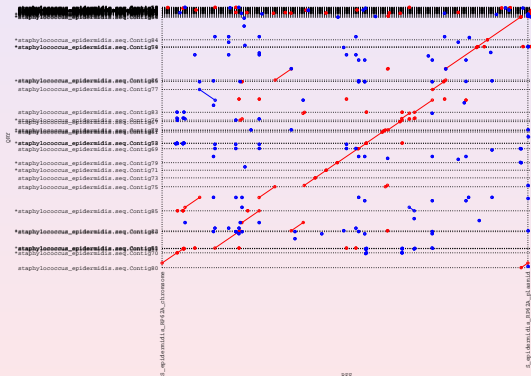
Brucella Suis - 2 chromosomes of 2,107,792 and 1,207,381 bp (SUTTA DotPlot)



Num. of reads: 36,276; Avg read length: 895.8; Coverage: 9.8X

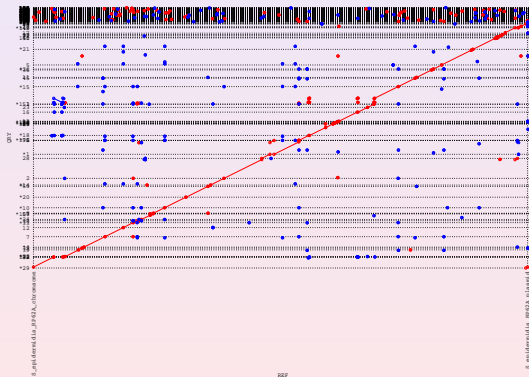
Staphylococcus Epidermidis - 2,616,530 bp (Phrap DotPlot)

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 (SUTTA DotPlot)



Num. of reads: 60,761; Avg read length: 900.2; Coverage: 19.9X



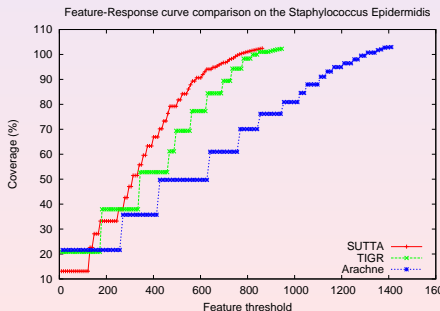
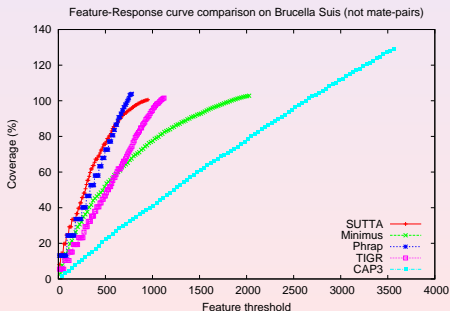
Feature-Response curve

(motivation)

- 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.

Feature-Response Curve (computation)

- For a fixed feature threshold ϕ , 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.



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.

Short-Read Overlapper

Idea: use **exact matching**

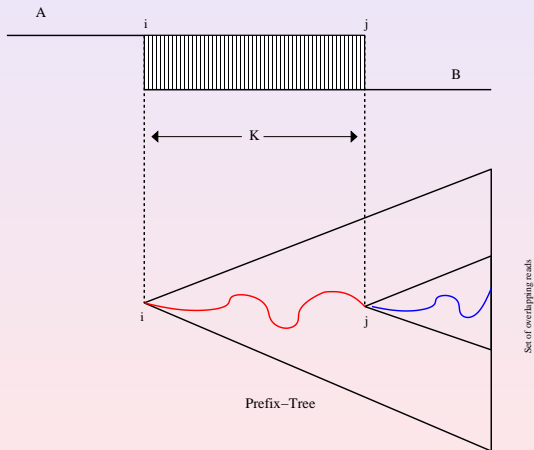
- allowing approximate matching (using dynamic programming) would significantly increase the number of nonspecific spurious overlaps (sequencing errors)
- drastically faster than approximate matching

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

illustration



Comparison of assemblies

short read data

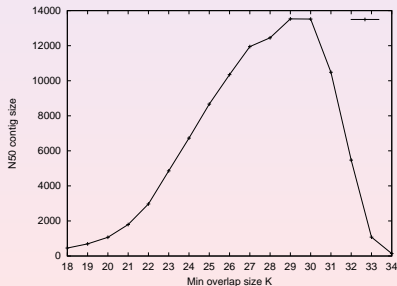
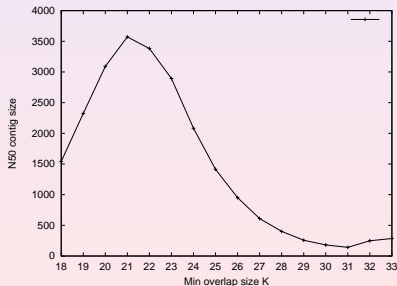
- Only contigs ≥ 100 bp.
- Correct contig (*S. aureus*): aligned along its whole length with at least 98% base similarity.
- Correct contig (*E. coli*): fewer than 5 consecutive base mismatches at the termini and at least 95% base similarity.

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

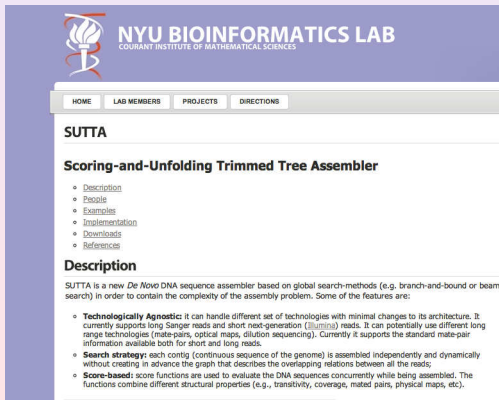
N50 = the largest number L such that the combined length of all contigs of length $\geq L$ is at least 50% of the total length of *all* contigs.

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**.



<http://bioinformatics.nyu.edu/wordpress/projects/sutta/>



The screenshot shows the NYU Bioinformatics Lab website. At the top left is the lab logo, a stylized DNA double helix with a flame-like top. To its right is the text "NYU BIOINFORMATICS LAB" and "COURANT INSTITUTE OF MATHEMATICAL SCIENCES". Below the logo is a navigation menu with four items: "HOME", "LAB MEMBERS", "PROJECTS", and "DIRECTIONS". The main content area is titled "SUTTA" and features a sub-header "Scoring-and-Unfolding Trimmed Tree Assembler". Underneath are several links: "Description", "People", "Examples", "Implementation", "Downloads", and "References". A "Description" section follows, starting with a paragraph about SUTTA being a new *De Novo* DNA sequence assembler. It then lists three key features: Technologically Agnostic, Search strategy, and Score-based.

NYU BIOINFORMATICS LAB
COURANT INSTITUTE OF MATHEMATICAL SCIENCES

HOME LAB MEMBERS PROJECTS DIRECTIONS

SUTTA

Scoring-and-Unfolding Trimmed Tree Assembler

- [Description](#)
- [People](#)
- [Examples](#)
- [Implementation](#)
- [Downloads](#)
- [References](#)

Description

SUTTA is a new *De Novo* DNA sequence assembler based on global search-methods (e.g. branch-and-bound or beam search) in order to contain the complexity of the assembly problem. Some of the features are:

- Technologically Agnostic:** it can handle different set of technologies with minimal changes to its architecture. It currently supports long Sanger reads and short next-generation (Illumina) reads. It can potentially use different long range technologies (mate-pairs, optical maps, dilution sequencing). Currently it supports the standard mate-pair information available both for short and long reads.
- Search strategy:** each contig (continuous sequence of the genome) is assembled independently and dynamically without creating in advance the graph that describes the overlapping relations between all the reads;
- Score-based:** score functions are used to evaluate the DNA sequences concurrently while being assembled. The functions combine different structural properties (e.g., transitivity, coverage, mated pairs, physical maps, etc).

Outline

- Whole-Genome Shotgun Sequence Assembly
- An Explosion 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...]

Scalability

**A single human genome requires 100 GB of raw data —
We need extremely scalable systems and algorithms.**

- **Cloud Computing:** using *MapReduce* ... Google's parallel distributed framework Data and computations are spread over thousands of computers... 946,460 TB processed in May 2010 (Jeff Dean @ Stanford, Nov 10, 2010)
- **Hadoop** is the leading open source implementation ... GATK is an alternative implementation specifically for NGS
- **Hadoop MapReduce:** Scalable, Efficient, Reliable and Easy to Program; BUT requires Redesigning / Retooling applications; Everything must be in MapReduce (Not Condor, Nor MPI)...

Strategy: Similar to Crossbow

- **Map:** Total ReCaller
 - Call bases while aligning each read to a genotype reference
 - Align single-molecule maps
 - Emit (chromosome region, alignment for reads and maps)
- **Shuffle:** Hadoop
 - Group and sort reads and maps into overlapping regions
- **Reduce:** SUTTA
 - Make haplotype assemblies for each region
 - Combine into whole-genome assembly (while integrating unaligned reads and optical maps)

Cost and Time Estimates

1 Arabian Individual Genome

Data Loading	3.3 B reads	106.5 GB	\$10.65	AED 39.12
Data Transfer	1h :15m	40 cores	\$3.40	AED 12.49
Setup	0h : 15m	320 cores	\$13.94	AED 51.20
Alignment	1h : 30m	320 cores	\$41.82	AED 153.61
Assembly	2h : 00m	320 cores	\$55.76	AED 204.81
End-to-end	5h : 00m		\$125.57	AED 461.23

1001 Arabian Individual Genomes = AED 461,691.23.
(Based on an estimation by M. Schatz of CSHL.)

[End of Talk]

- **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.”

[End of Talk]

- **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.”

[End of Talk]

- **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.”

[End of Talk]

- **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.”