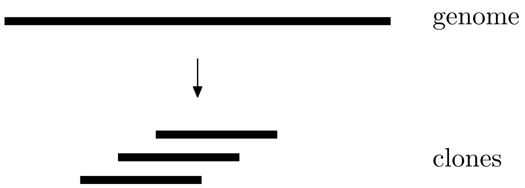
Fingerprint-based physical mapping

Dustin Cartwright (joint with Alexander Gutin)

October 30, 2007

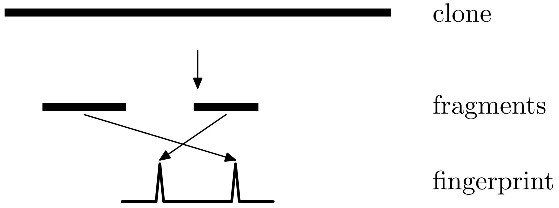
BAC clones

Break the genome into clones (about 100 kbp in length).



Fingerprints

The clones are then digested by restriction enzymes and the lengths of resulting fragments are measured via (gel or capillary) electrophoresis. A fingerprint is the collection of these fragment sizes.



Digression: Fragment “sizes” are not really sizes

With capillary electrophoresis (newer technology):

- Measurements of different fragments of the same size vary by 1–2 bps.
- Measurements of the same fragment vary by about .2 bps.

Digression: Fragment “sizes” are not really sizes

With capillary electrophoresis (newer technology):

- Measurements of different fragments of the same size vary by 1–2 bps.
- Measurements of the same fragment vary by about .2 bps.

Conclude: Fragment “sizes” are an invariant of the fragment, which closely correlates with, but is not identical to, number of base pairs.

Digression: Fragment “sizes” are not really sizes

With capillary electrophoresis (newer technology):

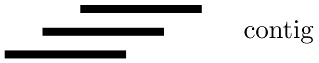
- Measurements of different fragments of the same size vary by 1–2 bps.
- Measurements of the same fragment vary by about .2 bps.

Conclude: Fragment “sizes” are an invariant of the fragment, which closely correlates with, but is not identical to, number of base pairs.

In fact, this makes fingerprints more informative.

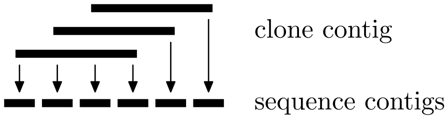
Physical mapping

Goal: Use the fingerprint information to build a physical map, a reconstruction of the (relative) layout of the clones in the genome. Each cluster of overlapping clones is a contig:



Physical mapping in sequencing

It is possible to sequence the ends the BAC clones. These sequences can be used to anchor sequence contigs to the physical map.



Overview of algorithm

Input: Set of clones, and for each clone a set of fragment sizes.
Output: Set of contigs, each of which gives the relative positions of the clones in the contig

- Filter frequent fragments
- Repeat 5 times: Detect pairwise matches (ovelapping clones) and estimate parameters (subset of the data)
- Detect all pairwise matches
- Filter frequently matched fragments
- Filter matches based on graph
- Final assembly

Overview of algorithm

Input: Set of clones, and for each clone a set of fragment sizes.
Output: Set of contigs, each of which gives the relative positions of the clones in the contig

- Filter frequent fragments
- Repeat 5 times: Detect pairwise matches (ovelapping clones) and estimate parameters (subset of the data)
- Detect all pairwise matches
- Filter frequently matched fragments
- Filter matches based on graph
- Final assembly

Detecting pairwise matches

Likelihood-based model for detecting matches between presumptive overlapping clones, with parameters estimated from data:

- Distribution of fragment sizes
- Standard deviation of size measurement procedure (variable across range of fragment lengths)

Detecting pairwise matches

Likelihood-based model for detecting matches between presumptive overlapping clones, with parameters estimated from data:

- Distribution of fragment sizes
- Standard deviation of size measurement procedure (variable across range of fragment lengths)

Output

For every detected match:

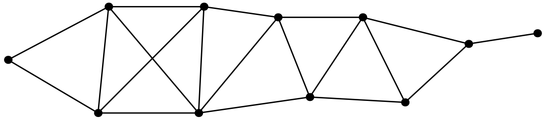
- Likelihood ratio
- Pairings between fragments in the two clones

Filtering matches

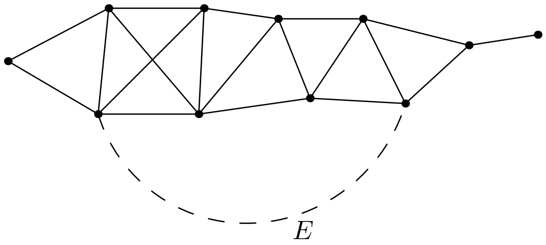
Detect false matches from topology of the match graph:

- Vertices are the clones
- Edges are matches

Add edges in order of decreasing likelihood ratio and throw out those which cause the graph to deviate from the ideal "tube-like" topology:



Acyclic filtering



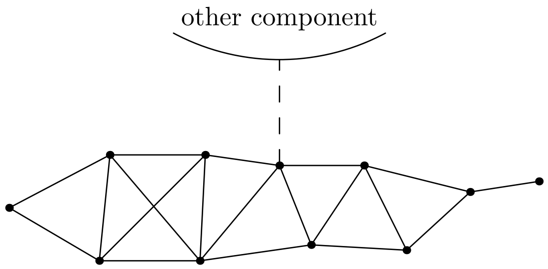
For each new edge E

- Let X be the maximal 2-simplex on E together with all previous edges.
- Let Y be the maximal 2-simplex on the whole graph.

Keep E if and only if we have

$$\begin{aligned} H_1(X, \mathbb{Z}) &\to H_1(Y, \mathbb{Z}) \\ [E] &\mapsto 0 \end{aligned}$$

Linear graph filtering



When adding an edge E joining two components:

- Let D_1, D_2 be the diameters of the components.
- Define an endpoint of component i to be a vertex which is a distance D_i away from another vertex in the component.
- Keep E only if its vertices are within 2 steps of endpoints of their respective components.

Final assembly

- Work with one component of the match graph (cluster) at a time.
- Group paired fragments into consensus fragments.
- Group consensus fragments which come from same set of clones into bins. A bin is represented by:
  - A set of clones
  - Number of consensus fragments.

Consecutive ones problem

Input: Matrix of 0s and 1s:

$$\begin{matrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{matrix}$$

Output: Permutation of columns such that within each row, all 1s are consecutive or failure there is no such permutation:

$$\begin{matrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{matrix}$$

Consecutive ones problem

Input: Matrix of 0s and 1s:

$$\begin{array}{ccccc} 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{array}$$

Output: Permutation of columns such that within each row, all 1s are consecutive or failure there is no such permutation:

$$\begin{array}{ccccc} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{array}$$

Analogy

rows = clones, columns = bins.

Algorithms for the consecutive ones problem

- Booth-Lueker (1976): Iterate over rows and build up tree represent constraints on the column orders. Linear in number of 1s.

Algorithms for the consecutive ones problem

- Booth-Lueker (1976): Iterate over rows and build up tree represent constraints on the column orders. Linear in number of 1s.
- Depth-first search over column orderings with lots of pruning.

Using consecutive ones problem to build contigs

Input: List of bins, integer C.
Output: Subset of bins, ordered as in consecutive ones problem, or failure.

- Loop until $> C$ bins have been removed or the remaining bins are orderable:
  - Use consecutive ones algorithm on bins.
  - If failure, discard bin with fewest consensus fragments.
- If $> C$ consensus fragments have been removed, return failure.
- Otherwise, loop over discarded bins in reverse order of discarding:
  - Temporarily add back discarded bin and use consecutive ones algorithm.
  - On success, keep bin. On failure, discard permanently.

Using consecutive ones problem to build contigs

Input: List of bins, integer C.

Output: Subset of bins, ordered as in consecutive ones problem, or failure.

- Loop until $> C$ bins have been removed or the remaining bins are orderable:
  - Use consecutive ones algorithm on bins.
  - If failure, discard bin with fewest consensus fragments.
- If $> C$ consensus fragments have been removed, return failure.
- Otherwise, loop over discarded bins in reverse order of discarding:
  - Temporarily add back discarded bin and use consecutive ones algorithm.
  - On success, keep bin. On failure, discard permanently.

Remark: The resulting subset of bins is a maximal orderable subset in a certain sense.

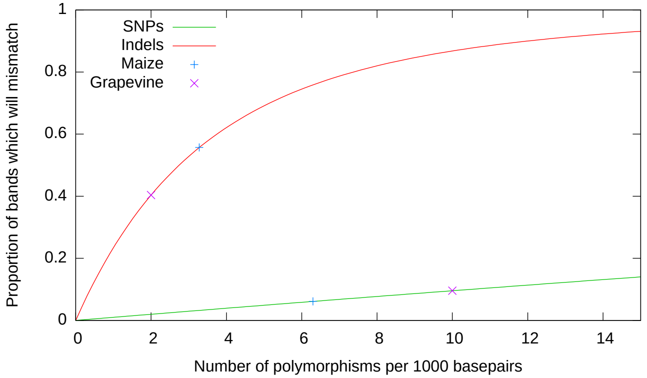
Incrementally adding clones

Rather than apply the previous algorithm on the totality of each cluster, we want to build contigs incrementally. This allows us to detect matches which cause problems.

- Initialize with no contigs
- For each clone in decreasing quality score (determined from trace)
    - For each contig which clone is connected to:
        - Try to add clone to contig using all matches between the two.
        - If successful, continue with merged contig in place of clone.

Heterozygous genomes

Newer, capillary-based fingerprinting has sufficient accuracy to detect insertion/deletion heterozygosity in the genome.



Heterozygous version of consecutive ones problem

Input: Matrix of 0s and 1s:

$$\begin{array}{ccccccc} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{array}$$

Output: A permutation of columns, and labelling of the columns by A, B, AB and the rows by A, B such that we have: for every row labeled by A (resp. B), all 1s are in columns labeled with A (resp. B) or AB and are consecutive within the subset of columns with those labels.

$$\begin{array}{cccccccc} & \text{AB} & \text{AB} & \text{AB} & \text{A} & \text{B} & \text{AB} & \text{AB} \\ \text{A} & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ \text{B} & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ \text{A} & 0 & 0 & 1 & 1 & 0 & 1 & 1 \end{array}$$

Heterozygous version of consecutive ones problem

Analogy

- rows = clones
- columns = bins
- row labels = chromosomal origin of clone
- column labels = chromosomal origin of consensus fragments (AB = common to both).

$$\begin{array}{cccccccc}
 & AB & AB & AB & A & B & AB & AB \\
A & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
B & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\
A & 0 & 0 & 1 & 1 & 0 & 1 & 1
\end{array}$$

Heterozygous version of final assembly

Heterozygous assembly works similarly except that:

- Consecutive ones algorithm generalized to heterozygous problem (Booth-Lueker does not seem to generalize).
- Clone labels are preserved, and at each step only a subset are allowed to vary.

Simulation

Three programs:

- FPC: standard physical mapping software
- ASFP
- ASFP-heterozygous: heterozygous version of ASFP

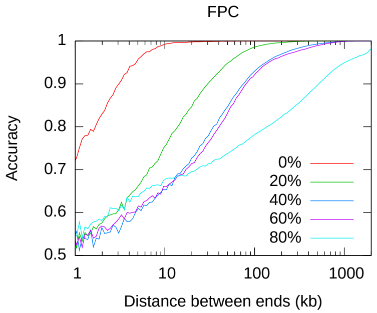
Simulation

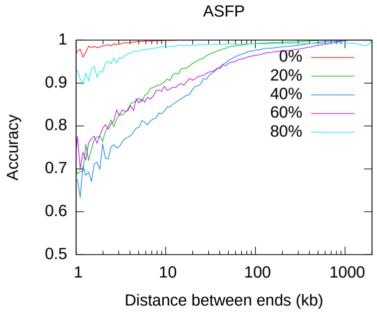
Three programs:

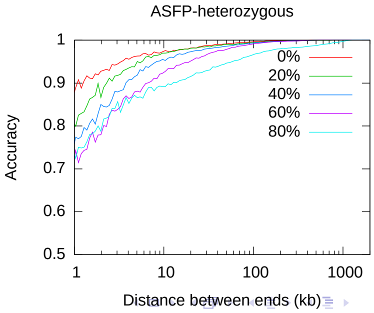
- FPC: standard physical mapping software
- ASFP
- ASFP-heterozygous: heterozygous version of ASFP

In the simulation, ASFP and ASFP-heterozygous had all filtering steps turned off.

Simulation results


FPC
Accuracy
Distance between ends (kb)


ASFP
Accuracy
Distance between ends (kb)


ASFP-heterozygous
Accuracy
Distance between ends (kb)