Good Maps are Straight[*]

Eric Harley
University of Toronto
Dept. of Computer Science
Toronto, Ont, Canada M5S 3G4
eharley@db.toronto.edu

Anthony J. Bonner
University of Toronto
Dept. of Computer Science
Toronto, Ont, Canada M5S 3G4
bonner@db.toronto.edu

Nathan Goodman
Whitehead Institute/MIT
Center for Genome Research
Cambridge, MA 02139, USA
nat@genome.wi.mit.edu

Abstract

This paper proposes a simplified approach to the assembly of large physical genome maps. The approach focuses on two key problems: (*i*) the integration of diverse forms of data from numerous sources, and (*ii*) the detection and removal of errors and anomalies in the data. The approach simplifies map assembly by dividing it into three phases—*overlap*, *linkage* and *ordering*. In the first phase, all forms of overlap data are integrated into a simple abstract structure, called *clusters*, where each cluster is a set of mutually-overlapping DNA segments. This phase filters out many questionable overlaps in the mapping data. In the second phase, clusters are linked together into a *weighted intersection graph*. False links between widely separated regions of the genome show up as crooked, branching structures in the graph. Removing these false links produces graphs that are straight, reflecting the linear structure of chromosomes. From these straight graphs, the third phase constructs a physical map. Graph algorithms and graph visualization play key roles in implementing the approach. At present, the approach is at an early stage of development: it has been tested on real and simulated mapping data, and the results look promising. This paper describes the first two phases of the approach in detail, and reports on our progress to date.

Introduction

A major goal of the Human Genome Project is to construct detailed physical maps of the human genome and the genomes of other organisms. A physical map gives the position on a genome of numerous small fragments of DNA, called *clones*. Most genomic experiments do not determine clone positions directly. Instead, they determine spatial relationships between them, such as whether two clones overlap. Inferring clone position from the experimental data is called *map assembly*. Assembling complete and detailed maps of large genomes requires the integration of many forms of experimental data from many sources. The resulting maps are said to *integrate* the data.

Algorithms for assembling integrated physical maps face several challenges. First, they must handle the increasingly large volumes of mapping data now being generated. Second, they must be flexible enough to accommodate new forms of data as they become available. Third, they must deal effectively with the complexities of the data, *e.g.*, with anomalies, inconsistencies and imprecision, and with subtle relationships between the different forms of data. Unfortunately, this last point complicates the map-assembly process enormously, limiting the effectiveness and flexibility of many map-assembly algorithms. To address this problem, we propose to simplify map assembly by dividing it into three phases—*overlap*, *linkage* and *ordering*.

The first phase integrates all forms of overlap data into a simple abstract structure, called *clusters*. A cluster is any set of mutually overlapping clones. Because they mutually overlap, the clones in a cluster must have at least one point in common; so the cluster can be thought of as a point on the genome. In addition, clustering filters out many false overlaps, because each overlap in a cluster is corroborated by many others. In this paper, all clusters have at least three clones. The section on overlap describes an algorithm for generating clusters.

After clusters are generated, the second phase links them together into a *weighted intersection graph*. Intersection graphs are a well-studied class of graphs in which each node is a set, and an edge means that two sets intersect. In our case, each node is a cluster, and an edge of weight M means that two clusters have M clones in common. Intuitively, an edge is evidence that two clusters are close together on the genome, and the weight is the strength of that evidence. In this paper, we only consider edges of weight at least 2. This filters out many false links between widely separated regions of the genome (caused by an experimental anomaly

[*]Published in *Proceedings, Fourth International Conference on Intelligent Systems for Molecular Biology (ISMB)*, pages 88-97. AAAI Press. Conference held June 12–15, 1996, Washington University, St. Louis, Missouri. Available on-line at ftp://db.toronto.edu/pub/bonner/papers/genome.mapping/

called chimerism), generalizing the notion of double linkage (Arratia et al. 1991).

If the experimental data were perfect, then the linkage graph would be straight, *i.e.*, long, thin and nearly linear, reflecting the linear structure of chromosomes.[1] However, errors and anomalies in the data distort the graph. We can identify many distortions using algorithms that examine graph structure and by exploiting graph visualization techniques. We can also remove large-scale distortions by exploiting large-scale mapping data, such as genetic maps. The result is a set of straight graphs, each representing a contiguous region of the genome, or *contig*. The remaining distortions are small and local, and are due largely to false negatives.

From these graphs, the ordering phase constructs a physical map. The first step is to determine a genomic order for the nodes in the graphs. As in genetic maps, we assign nodes to an ordered set of "bins." Within a bin, the relative order of nodes is unknown; but with high probability, all nodes in one bin precede all nodes in the next. A novelty of our approach is that if a node cannot be confidently assigned to a single bin, then it is assigned to several bins, to represent the uncertainty in its position. Finally, from these bin assignments, we construct a physical genome map, estimating the position and error of each clone.

Implementing the above approach involves developing numerous graph algorithms. For example, cluster formation is an NP-complete problem that reduces to finding all the *maximal* cliques in a graph. We have adapted the Bron-Kerbosch maximal clique algorithm (Bron & Kerbosch 1971) to exploit the sparsity of the graphs for overlap data. It generates all of the clusters for human genome data—40,000 of them—in 3 minutes. Other algorithms identify large-scale nonlinearities in our graphs, and others extract nearly-linear subgraphs from them. Graph layout and visualization algorithms are also an important part of the approach.

At present, the above approach to map assembly is at an early stage of development, and the results are promising. Algorithms developed for the first two phases have been extensively tested on real mapping data covering the entire Human genome and on data for selected genomic regions. Algorithms for the third phase have been tested on simulated data, and on data covering small regions of Human Chromosome 7. This paper reports on our progress, focusing on the first two phases—overlap and linkage, including contig extraction. Due to space limitations, the paper does not elaborate on the third phase—ordering—which will be described in a forthcoming work. The paper includes numerous examples from our on-going project to integrate the mapping data from two large genome centers, namely the Whitehead Institute/MIT Center for Genome Research, and the Centre d'Etude du Polymorphisme Humain (CEPH) and Généthon.

[1]Formally, it would be an *interval graph* (Golumbic 1980). See (Corneil, Olariu & Stewart 1995) for a theory of linear structure in graphs.

Background and Related Work

Constructing a complete physical map of the human genome requires an indexed library of clones of short, contiguous sections of DNA comprising the entire human genome. This library serves as a reservoir of genomic fragments for subsequent study of sequence, function and malfunction of genes. The index, or *physical map*, assigns each clone to its location on the genome. The map together with the library provides direct access to specific locations in the genome, thus facilitating the isolation of disease genes and the assembly of sequence data.

Substantial progress towards this goal has been made. A library of megabase sized genomic fragments cloned in yeast (yeast artificial chromosomes, or YACs) was created and made available to the genome community by CEPH/Genethon (Chumakov et al. 1995). Whitehead/MIT has assembled an STS-based map of the human genome consisting of 15,000 STSs screened against the CEPH YAC library (Hudson et al. 1995). The map is estimated to cover 95% of the genome. A clone based map from CEPH estimated to be about 75% complete (Chumakov et al. 1995) along with detailed "second-generation" physical maps of several human chromosomes by other labs have been published in the Genome Directory (*cf.* Genome Directory 1995).

Despite this progress, the assembly of physical maps remains a complicated process because of a number of biological phenomena and experimental artifacts, such as chimerism, homologous regions, unclonable regions, unstable clones, distribution biases of repeats and genetic markers, non-uniform efficiency of probe screening, and experimental errors (Cohen, Chumakov, & Weissenbach 1993). These problems necessitate the use of several forms of data, such as polymerase chain reaction (PCR)-based amplification of polymorphic sequenced tagged sites (STSs), hybridization to clones of probes composed of inter-Alu sections amplified using the polymerase chain reaction (PCR), restriction enzyme fingerprinting of clones, radiation-hybrid data, genetic mapping of STSs, and fluorescent in-situ hybridization (FISH) of clones to metaphase chromosomes. The first three of these sources of data indicate which pairs of clones overlap, while the last three localize clones or probes to particular regions of the chromosome.

As yet there is no completely automated method that converts the above forms of data into a physical map. The approach taken by CEPH/Genethon consists of several steps, as follows: (1) construct a framework of genetically mapped STSs; (2) use various forms of YAC overlap data to assemble reasonable minimal YAC paths between the mapped markers, where "reasonable" means not using STSs or YACs that belong to other regions of the map; (3) manually inspect paths for consistency and absence of cycles; (4) extend or improve paths by doing specific experiments; and (5) map some YACs that contain genetic markers to the cytogenetic map using FISH. The approach taken by Whitehead/MIT consists of different steps: (1) link together STSs that share at least two YACs, thus forming *doubly-linked contigs*; (2) place the contigs in the map by using genetic and radiation-hybrid data; (3) fill the gaps between contigs with the help of Alu-PCR and fingerprint data and single-linkage STS data; and (4) order the STSs within contigs using a simulated annealing algorithm. Other methods of physical map assembly include Letovsky's method of constraint propagation (CPROP) (Letovsky & Berlyn 1992), and methods based on temporal logic (Lee et al. 1993; Schmeltzer 1995).

Structure of STS Graphs

As described in the Introduction, the essence of our approach is to construct graphs that mimic the maps we are assembling. The benefit is that problems with the map reveal themselves as defects in the graph: bad maps are crooked and branched, while good maps are straight. This section illustrates this idea on a particular form of mapping data—STS content data. We examine this form of data first because it is naturally represented as a graph (Harley & Bonner 1994) and is convenient for introducing some of our graph-based techniques. The main techniques are *graph simplification* and *visualization*. The idea is to abstract or compress the detailed structure of a large and unwieldy graph, to give a much smaller and simpler graph, one that reveals the overall structure of the larger graph. The abstracted graph can be automatically laid out and displayed on a screen, from which we can see large-scale anomalies in the mapping data. We have found this technique to be particularly well-suited to graphs generated from genome mapping data. All graphs in this paper were laid out and displayed using the Hy+ data visualization system (Consens 1994).

An STS, or sequence-tagged site, is a known sequence of about 300 base pairs (bp), whose presence on a clone can be assayed by the polymerase chain reaction (PCR). STS probes serve two purposes: they mark a unique site in the genome, and they indicate which YACs overlap. Current mapping methods rely heavily on STS screening of the YACs in the library. The STS data analyzed in this paper were obtained from the following two sources: Release 8 (Sept. 1995) from the Human Physical Mapping Project at the Whitehead Institute/MIT Genome Center (available by anonymous ftp to `genome.wi.mit.edu`), and the March 1995 Release from CEPH (available by anonymous ftp to `ceph-genethon-map.cephb.fr`). The number of STSs and YACs in the two data sets and some average statistics are summarized in Table 1. The fact that the average number of YACs hit per STS is about 6 implies a 25-40% rate of *false negatives*, since the portion of

	Whitehead	CEPH	Merge
YACs	19,939	18,298	24,448
hits/YAC	3.5 ± 2.6	2.0 ± 1.8	3.6 ± 3.1
STSs	11,990	7,588	14,888
hits/STS	5.9 ± 4.1	4.9 ± 4.0	5.9 ± 4.0

Table 1: Statistical characterization of STS data.

This table characterizes the STS data from Whitehead Release 7; CEPH release Mar, 1995; and the merge of the two databases. We include both definite and disambiguated addresses in the Whitehead data. The YAC count includes only YACs positive for at least one STS. Average YAC length was 1000 ± 500 *kb.*

the library screened by Whitehead has 8-fold coverage (Hudson et al. 1995), and the portion screened in the merged data set has 10-fold coverage (Chumakov et al. 1995).

STS Intersection Graphs

One way to represent STS data is with a weighted intersection graph (Harley & Bonner 1994). In this graph, each node is an STS (or equivalently, the set of YACs hit by the STS). There is an edge of weight $M > 0$ between two nodes if the corresponding STSs hit M YACs in common. Figure 1 shows a schematic representation of noise-free STS-YAC data and the resulting STS intersection graph. The weight of each edge is given by the number of YACs bridging the two endpoint STSs. For example, the weight of the edge between STSs a and b is two, and that between STSs b and c is one.

STSs which are adjacent in the intersection graph are expected to be close together in the genome. For this to be true, we must suppress edges with weight $M = 1$, since there is a good chance that such edges are artifacts caused by chimeras or random false positives in the assay. For example, in the intersection graph for the merged Whitehead and CEPH STS data, 53% of the edges connect pairs of STSs for which the chromosome location of each endpoint is known. Of these, 67% of the unit weight edges ($M = 1$) and only 6% of the non-unit weight edges ($M > 1$) connect STSs that are on different chromosomes. Thus, we will only consider intersection graphs for which $M \geq 2$, which we call *double linkage* intersection graphs.

Double linkage is a good method for eliminating spurious connections caused by chimeras, since it is very unlikely that two chimeric YACs will happen to link the same unrelated STSs (Arratia et al. 1991). However, it provides no defense against spurious links caused by repeat regions. It appears that most of the non-unit weight edges that link STSs on different chromosomes are caused by non-unique STSs, i.e., STSs that hit repeat regions. This appears likely for

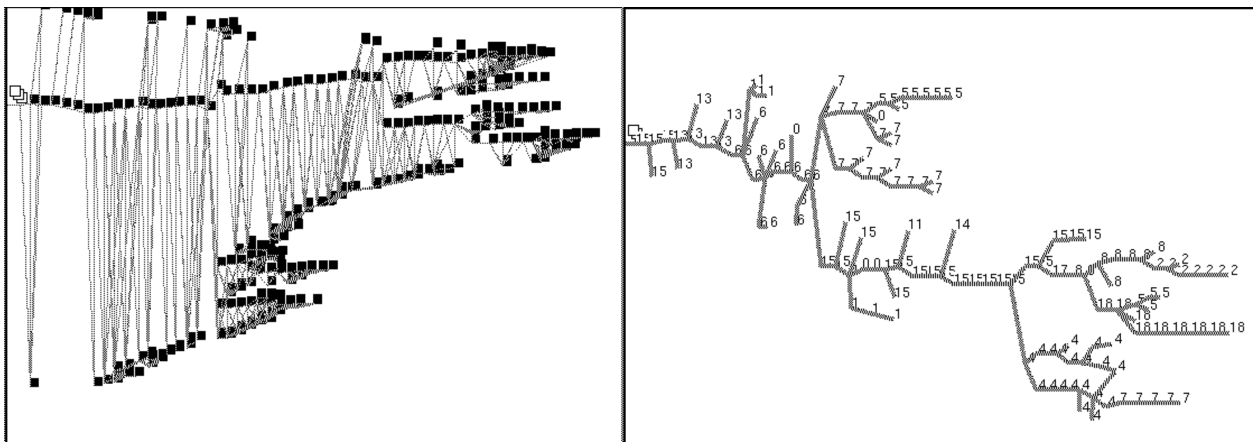

Figure 2: STS intersection graph and structure graph of the human genome.

(a) On the left is shown one component (441 nodes, 1,540 edges) of the STS double-linkage intersection graph for the whole genome. (b) On the right is the corresponding structure graph (144 nodes, 145 edges). Icons in the structure graph represent the chromosome assigned to the majority of STSs within each blob.

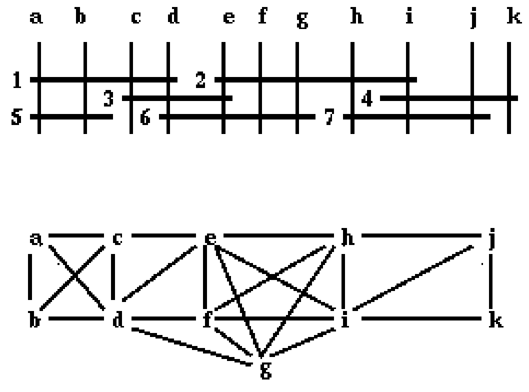

Figure 1: Schematic and graph of STS-YAC data

In the drawing at the top, YACs are represented as horizontal lines. The presence of an STS on a YAC (a hit) is indicated by a vertical bar crossing the YAC. The corresponding STS intersection graph is shown below.

two reasons. First, the number of such edges (905) is at least ten-fold higher than one would expect on the basis of chimerism alone. Second, the number of YACs involved (the weight of the edge) ranges from 2 to 10, whereas there should be virtually no chimeric links with weight greater than 2.

One of the larger components of the double-linkage intersection graph for the merged STS data of the whole genome is shown in Figure 2(a). Chromosome assignments of the 441 STSs in this component are spread over 16 chromosomes, including 78 assignments to Chromosome 7, 64 to Chromosome 15, 57 to Chromosome 4, etc. Thus, edges caused by nonunique STSs and double chimeras connect widely-separated regions of the genome in this component. The next section describes a method that makes the presence of these "cross-links" easily visible in many cases, whether they link different chromosomes or distant sites on the same chromosome.

Graph Simplification

As the amount of mapping data increases, the intersection graphs rapidly become large, unwieldy and impossible to visualize. This is already apparent in Figure 2(a), which is a relatively small graph. However, much of the complexity arises from small-scale details. In contrast, the large-scale structure of the graph is relatively simple, and reveals large-scale problems with the developing genome map. To elucidate this structure, we have developed several algorithms for graph abstraction. This section describes one of them.

We define a simplification of a graph G, called a *structure graph* G'. The simplification is achieved by coalescing nodes of G into sets called *blobs*. Intuitively, a blob is a set of nodes from a small, localized region of the graph. We have experimented with different methods of defining these localized regions, and the method we found to be most insensitive to noise in the data while preserving underlying structure is based on breadth first search, a classical graph traversal algorithm.

Breadth first search (BFS) divides a graph G into *layers*, where the nodes in layer L_i are a minimal distance of i edges from a given starting node or from the nearest node in a set of starting nodes S. A *blob* is defined to be a connected component within a BFS layer; *i.e.*, a set of nodes that belong to the same layer and that are connected by a path using only nodes in this layer. We form a structure graph from an STS intersection graph G by the following five steps:

Step 1 Do a BFS traversal of G starting at an arbitrary node. This serves to define BFS layers relative to the starting node.

Step 2 Choose any node x in the last BFS layer L_n and form a blob S containing that node; i.e., S contains x and the neighbours of x that are in layer L_n and the neighbours of those neighbours that are in L_n, and so on.

Step 3 Do a second BFS traversal of G starting from set S. This defines a new BFS layering of G to be used when constructing the rest of the blobs.

Step 4 Create the rest of the blobs of the structure graph as follows: for each node of G not yet in a blob, construct a blob (as was done for x in Step 2).

Step 5 Create the edges of the structure graph as follows: for each edge $\langle v_1, v_2 \rangle$ in the original graph G, where $v_1 \in blob_1$, $v_2 \in blob_2$ and $blob_1 \neq blob_2$, construct edge $\langle blob_1, blob_2 \rangle$ of the structure graph (unless that edge is already made).

As an example, we will construct the structure graph for the hypothetical data of Figure 1. The starting node in Step 1 is, say, h. The BFS layers are then $L_0 = \{h\}$, $L_1 = \{i, j, k, g, f, e\}$, $L_2 = \{d, c\}$, and $L_3 = \{a, b\}$. A node most distant from h is a, from which the first blob $b_0 = S = \{a, b\}$ is formed in Step 2. The second layering of Step 3 produces $L_0 = \{a, b\}$, $L_1 = \{c, d\}$, $L_2 = \{e, f, g\}$, $L_3 = \{h, i\}$, and $L_4 = \{j, k\}$. Each of these layers happens to be internally connected, so the blobs formed in Step 4 are the same as the layers of Step 3, i.e., $b_i = L_i, i = 1, \ldots, 4$. Step five connects blobs which contain adjacent nodes. For example $\langle b_0, b_1 \rangle$ is an edge in the structure graph since $\langle a, c \rangle$ is an edge in G and $a \in b_0$ and $c \in b_1$. Figure 3 shows two representations of the resulting structure graph.

.

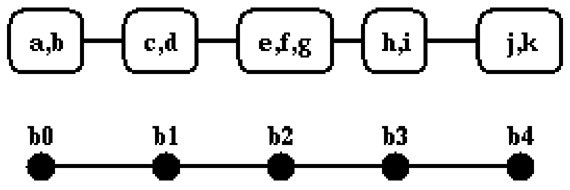

Figure 3: Structure graph

The structure graph corresponding to the intersection graph of Figure 1. The top view shows the contents of the blobs; the lower view hides blob contents. See text for details of construction.

The structure graph is simpler than the original graph since it has fewer nodes and edges. More importantly, it captures the underlying linear structure of intersection graphs derived from STS data. The following theorem states that the structure graph is a simple linear sequence of blobs for any intersection graph of noise-free STS-YAC data. The proof is straightforward and therefore omitted.

Theorem 1 *Let $G = (V, E)$ be a connected intersection graph for a set V of STSs such that the underlying STS-YAC data is perfect, i.e., noise-free. Then the structure graph G' derived from G by the structure graph algorithm is a chordless path.*

According to this theorem, deviations in a structure graph from a simple path similar to that of Figure 3 reflect deviations from ideal data. The simple path structure is robust to the extent that it is preserved by false negatives which do not interrupt connectivity within a BFS layer. More disruptive false negatives create "hair" on the structure graph, i.e., small branches typically of length 1 or 2. Non-unique STSs and chimeras can produce long branches, while the branches themselves remain simple or "hairy" paths.

The structure graph corresponding to the STS graph in Figure 2(a) is shown in Figure 2(b). The structure graph has ten-fold fewer edges than the original graph (145 compared to 1,540), and about three-fold fewer nodes (144 compared to 441). It is tree-like, with many hairy branches. The blobs are labeled with chromosome numbers to show that much of the branching occurs at the junctures of chromosomes. The icon for each node represents the chromosome to which the majority of the STSs in the blob are assigned. The icon "0" indicates that there is no majority chromosome assignment for that blob, either because most STSs in the blob are unassigned or because they are assigned to a variety of chromosomes. Blobs for a single chromosome tend to form simple or hairy paths, but not always—those for Chromosome 4 form a loop.

The structure graph of Figure 2(b) makes evident the composite but underlying piecewise-linear nature of the graph in Figure 2(a). Clearly this connected component consists of a collection of contigs from a number of chromosomes tied together by repeats or double chimeras. Restricting attention to a single chromosome, however, does not guarantee good contigs, as the loop for Chromosome 4 demonstrates. We will discuss extraction of contigs from structure graphs in the section on linkage, after we generalize the above notions to clusters.

Data Integration Using Clusters

This section extends the graph-based approach developed above for STS data to other forms of mapping data. We illustrate the approach on three common forms of physical mapping data—STS, fingerprint, and Alu-PCR data. These three forms of data are typically examined separately, as in the map-assembly methods described for CEPH and Whitehead. In contrast, the approach described here integrates the different forms of data by treating each as a source of YAC overlap information.

The main idea is to convert all forms of overlap data to a common denominator, called *clusters*, which at once unify the data and weed out some false positives. Each cluster is a maximal set of mutually overlapping YACs. The *center* of a cluster is the area on the genome which is shared by all YACs in the cluster. If an STS were at the center of a cluster, then the probe would (barring false negatives) hit all of the YACs. In this sense, a cluster is analogous to an STS. We form cluster intersection graphs and structure graphs just as we did for STS data. The main difference is that in cluster graphs, we use more data and should therefore obtain larger contigs than from STS data alone. As described in the Introduction, the approach is divided into three phases—overlap, linkage and ordering. This section describes the first two phases in detail, and comments briefly on the third phase.

Overlap

This phase extracts overlap information from the experimental data, and converts it into clusters. For the examples in this section, we obtained Alu-PCR data and fingerprint overlap information from the CEPH releases of March 1995 and May 1994, respectively. The CEPH file `RELATIONS` provides overlaps for 31,392 YACs based on fingerprint data. The Alu-PCR screening defines overlaps for 8,785 Alu-PCR YAC probes against 24,576 YAC targets. We merged 198,157 overlap pairs implicit in the STS data of Section 2 with the Alu-PCR and fingerprint overlaps, to obtain a total of 255,615 different pairs of overlapping YACs.

Converting this overlap data into clusters corresponds to the classical NP-complete problem of finding all maximal cliques in a graph. In this case, the nodes of the graph are YACs, and there is an edge between two nodes if the corresponding YACs overlap.[2] If a YAC Y overlaps with n other YACs, then there are 2^n subsets of these neighbors, any or all of which could form a clique with Y, depending on which neighbors overlap with each other. In general each of these subsets must be examined. The merged data contains overlap data for 30,598 YACs, and the largest number of neighbors is 125. Since 2^{125} is an astronomically large number, and since we cannot expect to find a better-than-exponential algorithm for an NP-complete problem, we need heuristics to speed up the search.

The branch and bound heuristic of (Bron & Kerbosch 1971) is an effective method to reduce the search space for maximal cliques. We modified the Bron-Kerbosch algorithm in the following way. Since our graphs are large and sparse, we represented edges by adjacency lists rather than as a two-dimensional array. An initial step in the Bron-Kerbosch algorithm requires time quadratic in the number of nodes. This step is exorbitant for a graph of 30,000 nodes. By adapting the algorithm to work on one small subgraph at a time, where each subgraph is that induced by a YAC and its neighbours, the running time is reduced from hours to minutes. The result is that all maximal cliques for 30,441 YACs are generated in just three minutes. Details of this algorithm will be described elsewhere.

[2]This YAC-overlap graph should not be confused with the intersection graphs developed in previous sections.

The total number of maximal clusters generated for the above data is 40,060, including 27,455 of size 3 or more. The sizes of clusters range from 2 to 30 YACs. The frequency of clusters of a given size decreases exponentially as a function of size, size 2 being the most frequent (12,605 clusters), then size 3 (5,003), size 4 (4,092), etc.

Finally, we assigned clusters to particular chromosomes based on known assignments of STSs to chromosomes. A cluster was assigned to a chromosome if an STS assigned to that chromosome hits at least two YACs in the cluster. This method implies tight linkage between a cluster and its assigned chromosome(s). Ignoring clusters of size 2 (for reasons mentioned below), only 13% of the clusters were left unassigned by this method; 74% of the clusters were uniquely assigned, 11% were doubly-assigned, and the remaining 2% were assigned to 3-7 chromosomes.

Linkage

This phase of map assembly links clusters into contigs. Here, the term "contig" means a graph representing a contiguous region of the genome. In these graphs, each node is a cluster, and an edge means that two clusters are close together on the genome. These graphs should be straight, reflecting the linear structure of chromosomes. Unfortunately, because of errors and anomalies in the data, it is easy to get graphs that are crooked and branched. Moreover, because of the huge volumes of data involved, the graphs are large, unwieldy and impossible to visualize. To solve these problems, we divide this phase into several steps. First, we build an intersection graph from the clusters. Then, we simplify the graph so that its overall structure is apparent. Finally, we identify false links in the graph and extract contigs from it.

Cluster Intersection Graphs: These are a generalization of the STS intersection graphs. In a cluster intersection graph, each node is a cluster, and there is an edge of weight M between two nodes if the corresponding clusters have $M > 0$ YACs in common. As with STS graphs, we only consider double linkage ($M \geq 2$). We also ignore clusters of size two since they cannot contribute to double linkage graphs—if a cluster of size two shares its two YACs with another cluster, then it is not maximal, which is a contradiction. Moreover, in the case of Alu-PCR and fingerprint data, where each overlap is independent of other overlaps, a size two

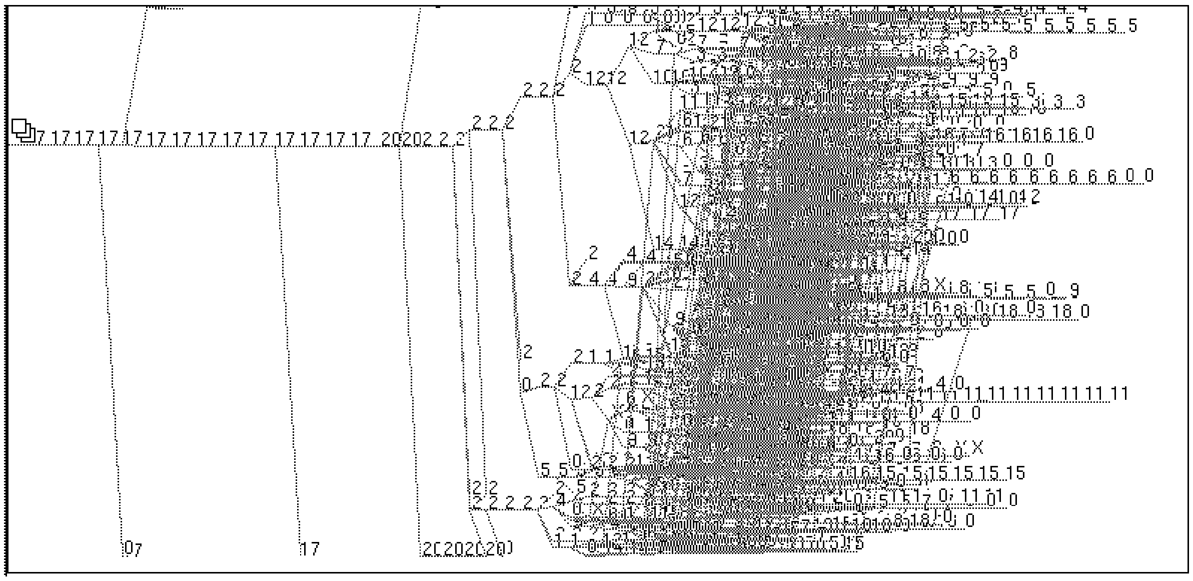

Figure 4: Cluster structure graph of the human genome

The structure graph of the largest component of the double-linkage cluster intersection graph for the whole human genome. The source graph has 24,304 nodes, 266,444 edges; the structure graph: 4,621 nodes, 5,001 edges.

cluster is not as reliable as a larger cluster, because it lacks the mutual corroboration of overlap evidence inherent in larger clusters.

The double-linkage cluster intersection graph for the whole genome is almost entirely one component of 24,304 nodes (out of the 27,379 total) and 266,444 undirected edges. The size of this graph prohibited its display, because of time and memory limitations. Even if displayed, the detailed intersection graph would not be intelligible as it would blacken the screen with edges.

Graph Simplification: The structure-graph algorithm described for STS intersection graphs can be applied to any graph, including cluster intersection graphs. The algorithm simplifies the cluster graph of the human genome to the extent that some piecewise-linearity is apparent, as shown in Figure 4. There are about 5,000 nodes and 5,000 edges in this structure graph, a reduction of five-fold and fifty-fold, respectively, from the cluster graph. Nevertheless, even the structure graph is extremely complex and interconnected. In this graph, the node labels represent assignments of blobs to particular chromosomes. A blob has label C if the majority of clusters in the blob are assigned to Chromosome C. The label is “0” if there is no majority assignment or if the majority of the clusters in the blob are unassigned. From these assignments, we can see that the graph represents numerous chromosomal fragments tied together by false links.

We can simplify the cluster intersection graph even further by the following three steps: first, extract a subgraph for a particular chromosome; then, apply the structure-graph algorithm; finally, trim off any remaining small-scale details. To illustrate the process, we selected a subset of clusters which might contribute to contigs for Chromosome 7, namely clusters assigned to Chromosome 7 and clusters not assigned to any chromosome. This filter produces a subset of 5,079 clusters. The double-linkage intersection graph for this set of clusters has 1,570 connected components, most of which (1,410) contain less than 5 clusters (*i.e.*, nodes). The three largest components contain 121, 223 and 645 clusters. The largest component is shown in Figure 5(a), along with its “trimmed” structure graph in Figure 5(b) and (c). The trim algorithm simplifies the structure graph by removing branches of length 1 and compressing cycles of length four, since such “hair” is not relevant to the overall structure. The trimmed structure graph is almost two orders of magnitude simpler than the cluster graph in Figure 5(a), since the number of edges is reduced from 6,736 to 72.

Contig Extraction: From the structure graph of Figure 5(b), it is immediately obvious that the cluster graph is not a single contig, since it has a number of piecewise linear sections and a cycle. To extract individual contigs from it, we use positional information in addition to the structural information. To this end, we assigned map positions to clusters and blobs based on known positions of STSs, taken from genetic and/or radiation-hybrid maps. We first assigned a position p to a cluster if the cluster contains all the YACs hit by

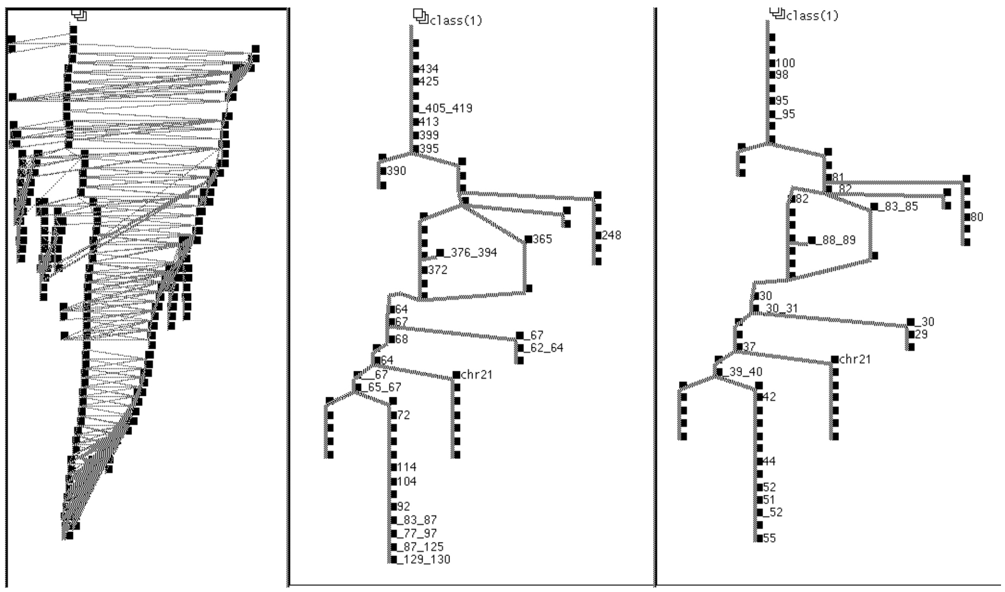

Figure 5: Cluster intersection graph and structure graphs of Human Chromosome 7.

(a) On the left the largest component of the cluster intersection graph for Chromosome 7 (645 nodes, 6,736 edges) is shown. The center (b) and right (c) panels contain the corresponding structure graph (72 nodes, 72 edges). Blobs in the center graph are labeled with radiation-hybrid positions (centiRays), and those in the right graph are labeled with genetic positions (centiMorgans) where known.

an STS at position p. A blob inherits all the positions of its component clusters. Figures 5(b) and 5(c) show two versions of the structure graph corresponding to Figure 5(a). In Figure 5(b), blobs are labelled with radiation-hybrid positions in centiRays (cR), while in Figure 5(c), they are labelled with genetic positions in centiMorgans (cM).[3] Blobs that have more than one map position are labeled with the positions connected by an underscore. For example, the lowermost blob in Figure 5(b) is labeled "129_130" since it contains a cluster at 129 cR and one at 130 cR. Only positions relevant to Chromosome 7 are shown. Observe that only a portion of the blobs are assigned map positions.

To extract a contig, we look for a simple path with a reasonable sequence of map positions. The longest such path of blobs includes the following sequence of radiation-hybrid positions: (62_64, 67, 67, 68, 64, 67, 65_67, 72, 114, 104, 92, 83_87, 77_97, 87_125, 129_130). and the following sequence of genetic positions: (29, 30, 30_31, 37 ,39_40, 42, 44, 52, 51, 52, 55), and This proposed contig is shown in the top of Figure 6. At the bottom of Figure 6 is a subgraph of the cluster intersection graph corresponding to the shortest paths from one extremum to the other of this contig. Genetic positions of the blobs and clusters are shown where known.

[3]The approximate correspondence between these measures and megabases is 1 megabase $\approx$ 5 cR $\approx$ 1 cM (*cf.* Whitehead Release 8).

Other, shorter contigs could be extracted based on structural and positional information in Figure 5. However, it is difficult to select the sequence of blobs that form a contig in the region of the cycle, since here both the positional and the structural information are ambiguous. Additional information or a search for "suspicious" clusters may resolve the ambiguity.

This example clearly illustrates two features of cluster/structure graphs: (*i*) they provide a visual aid in identifying errors and anomalies in physical mapping data, and (*ii*) they help in extracting contigs from the much-larger and more-complex cluster intersection graphs. These contigs are the result of integrating many forms of physical mapping data and of removing large-scale errors and anomalies. In this way, cluster/structure graphs greatly simplify the construction of physical maps, and can improve their completeness and accuracy.

Ordering

In the last phase of our approach to map assembly, we generate a physical map from the contig graphs produced in the linkage phase. A detailed discussion of the ordering algorithms we are developing for this phase is beyond the scope of this paper. However, some preliminary results are worth mentioning.

To estimate the accuracy of our approach, we constructed a map from the contig in Figure 6, and compared it to the current Whitehead map of Chromo-

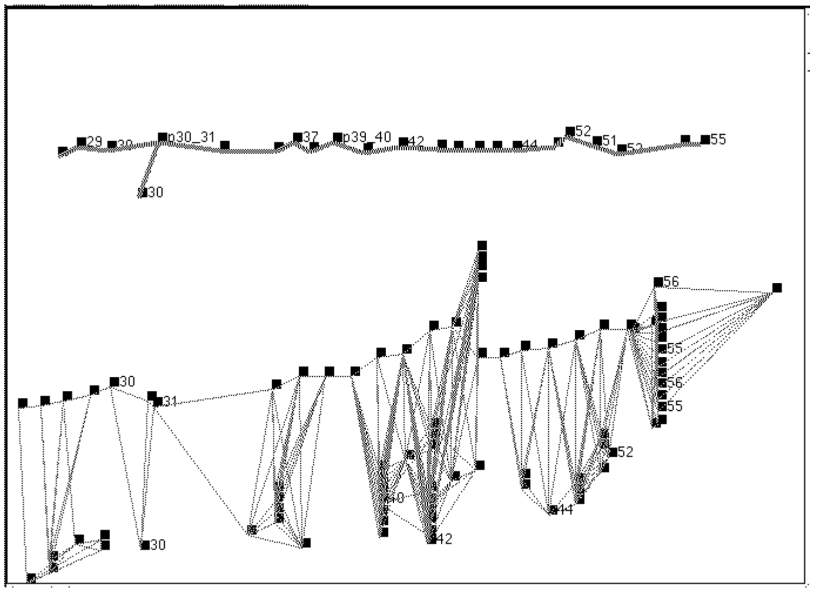

Figure 6: A contig from Human Chromosome 7.

On the top is shown a path extracted from the structure graph of Figure 5. The approximately sequential genetic positions indicated by the labels of the blobs suggest that this path represents a contig. The lower graph shows clusters contained in these blobs and shortest paths between the extrema in the corresponding cluster graph.

some 7 (Hudson et al. 1995). To construct this map, we ordered the clusters, and then identified clusters with STSs. A cluster is identified with an STS if it contains all the YACs hit by the STS. The resulting map shows that the doubly-linked contig of Figure 6 corresponds to a concatenation of four doubly-linked contigs in the Whitehead map (contigs WC-234, WC-427, WC-633, and WC-146). Whitehead also connects these contigs, and more, but by single linkage, in contig WC7.3. The cluster/STS order in our map is very similar to the STS order in WC7.3, and differences appear to be within the resolution of the Whitehead map. This result suggests that integration of data using clusters improves the coverage of double-linkage maps, and that contig extraction using structure graphs and positional information is viable.

Conclusion

This paper discusses a promising approach for the assembly of integrated genome maps. The essence of the approach is to devise discrete structures (*i.e.*, graphs) that nicely mimic the maps we are constructing. The benefit is that problems with the map reveal themselves as defects in the structure: bad maps are crooked and branched, while good maps are straight. The trick, of course, is to find a good balance between statistical and discrete reasoning, *e.g.*, to use statistics in forming clusters and links, then use discrete reasoning to find defects in the graph and to propose corrections, then return to statistics to choose the best correction. We are at an early stage in this process, and have only a few steps worked out. However, the examples and theory discussed in this paper provide strong evidence for the potential of the approach. Since the real test lies in map construction, our next step will be to generate maps using the cluster/structure method, and to determine whether they are more complete and accurate than maps prepared by other methods. We will also examine the possibility of automating the entire process.

Acknowledgements: We gratefully acknowledge the assistance and expert advice of the following people: Lincoln Stein at the Whitehead Institute/MIT Center for Genome Research (*re.* interpretation of genomic data); Derek Corneil at the University of Toronto (*re.* linear structure in graphs); Alberto Mendelzon, Dimitra Vista at the University of Toronto, and Mariano Consens at the University of Waterloo (*re.* the Hy+ data visualization system); and Gloria Kissin at the University of Toronto (*re.* critical comments).

This work was supported in part by a research grant from the Natural Sciences and Engineering Research Council of Canada (NSERC), and by funds from the U.S. National Institutes of Health, National Center for Human Genome Research, grant number P50 HG00098, and from the U.S. Department of Energy under contract DE-FG02-95ER62101.

References

Arratia, R.; Lander, E. S.; Tavare, S.; and Waterman, M. S. 1991. Genomic mapping by anchoring random clones: A mathematical analysis. *Genomics* 11: 806–827.

Bron, C.; and Kerbosch, J. 1971. Finding All Cliques of an Undirected Graph. *Communications of the ACM* 16: 575–577.

Chumakov, I. M.; Rigault, P.; Le Gall, I.; *et al.* 1995. A YAC contig map of the human genome. *Nature* 377 Suppl.: 175–298.

Cohen, D.; Chumakov, I.; and Weissenbach, J. 1993. A First Generation Physical Map Of The Human Genome. *Nature* 336: 698–701.

Consens, M. 1994. *Creating and Filtering Structural Data Visualizations using Hygraph Patterns.* Ph.D. Dissertation, Department of Computer Science, University of Toronto, 10 King's College Rd, Toronto, Ont, Canada.

Corneil, G.G.; Olariu, S.; and Stewart, L. 1995. Linear Time Algorithms for Dominating Pairs in Asteroidal Triple-Free Graphs *Technical Report 294/95*, University of Toronto, Dept. of Computer Science, Toronto, ON, M5S 3H5.

M.C. Golumbic. 1980. *Algorithmic Graph Theory and Perfect Graphs.* Academic Press.

Harley, E.; Bonner, A.J. 1994. A Flexible Approach to Genome Map Assembly. In *Proceedings of the Second International Conference on Intelligent Systems for Molecular Biology (ISMB)*, AAAI Press, 161–169.

Hudson, T.J.; Stein, L.D.; Gerety, S.S.; *et al.* 1995. An STS-Based Map of the Human Genome. *Science* 270: 1945–1954.

Lee, A.; Rundensteiner, E.; Thomas, S.; and Lafortune, S. 1993. An information model for genome map representation and assembly. *Technical Report* SDE-TR-163-93, University of Michigan, Dept of Electrical Engineering and Computer Science, Ann Arbor, MI 48109-2122.

Letovsky, S. and Berlyn, M. B. 1992. CPROP: A Rule-Based Program for Constructing Genetic Maps. *Genomics* 12: 435–446.

Schmeltzer, O. 1995. Using Temporal Reasoning for Genome Map Assembly. In *Proceedings of the Third International Conference on Intelligent Systems for Molecular Biology (ISMB)*, AAAI Press, 332–340.

Genome Directory. 1995. *Nature* 377 Suppl.