Revealing Hidden Interval Graph Structure in STS-Content Data

Eric Harley[1], Anthony Bonner[1] and Nathan Goodman[2]

Abstract

Motivation: STS-content data for genomic mapping contain numerous errors and anomalies resulting in cross-links among distant regions of the genome. Identification of contigs within the data is an important and difficult problem.
Results: This paper introduces a graph algorithm which creates a simplified view of STS-content data. The shape of the resulting *structure* graph provides a quality check — coherent data produce a straight line, while anomalous data produce branches and loops. In the latter case, it is sometimes possible to disentangle the various paths into subsets of the data covering contiguous regions of the genome, i.e., *contigs*. These straight subgraphs can then be analyzed in standard ways to construct a physical map. A theoretical basis for the method is presented along with examples of its application to current STS data from human genome centers.
Availability: Freely available on request.
Contact: `eharley@cs.toronto.edu`

Introduction

Two epic efforts to construct contig maps of the human genome were begun in the early 1990's, one led by Daniel Cohen at the Centre d'Etudes du Polymorphisme Humain and Généthon (CEPH/Généthon), and the second led by Eric Lander at the Whitehead Institute for Biomedical Research/MIT Center for Genome Research (WI/MIT). Both groups adopted a general strategy of combining *bottom-up* methods for constructing contigs across modest regions, with *top-down* methods for ordering the contigs across entire chromosomes. Both groups used STS-content mapping as a key bottom-up method, though the CEPH/Généthon team (which started first) initially used fingerprint mapping for this purpose. For top-down ordering of contigs, both groups planned to use the genetic map (Weissenbach et al., 1992; Gyapay et al., 1994; Dib et al., 1996) generated by a separate team at CEPH, led by Jean Weissenbach. Following successful demonstrations of radiation hybrid (RH) mapping (Cox et al., 1990; Walter et al., 1994), the WI/MIT group undertook extensive RH mapping to augment the Weissenbach genetic map. The groups published several partial maps (Cohen et al., 1993; Chumakov et al., 1995; Hudson et al., 1995), but were never able to produce a near-complete contig map of the human genome. Eventually they joined forces with each other, with several other groups engaged in more focussed mapping efforts, and with several groups engaged in EST sequencing projects, and produced a seminal gene map of the human genome (Schuler et al., 1996); this map stands as one of the major accomplishments of the Human Genome Project to date.

In genome parlance, a *map* is a collection of spatially related *features*. A contig is a portion of the map (or data) that is believed to correspond to a single continuous section of the genome. The contig maps in question involve two principal kinds of features: sequence-tagged sites (STSs) and yeast artificial chromosome clones (YACs). For our purposes, an *STS* is a short stretch of genome, typically a few hundred bases in length, and a *YAC* is a much longer stretch of genome, typically ranging in size from 500 kilobases (kilobase, or kb = 1000 bases) to as much as 2 megabases (megabase, or mb = 1,000,000 bases). The human genome is much larger still, comprising 3×10^9 bases (3,000 mb). The difference in scale between STSs, YACs, and the human genome is so large that we can reasonably think of the genome as a line, and STSs and YACs as points and intervals, respectively, along this line.

Contig mapping has proven to be an extremely difficult task, because of numerous sources of "noise" in the data. Both the CEPH/Généthon and WI/MIT teams were successful in collecting large data sets, but were unable to analyze these data sets to yield comprehensive contig maps. This is somewhat surprising in that the data analysis problem is superficially simple. Bottom-up mapping methods produce data indicating which clones overlap. This information may be viewed as a graph in which nodes correspond to clones, and an edge between two nodes represents overlap between the two clones. In an ideal world, the overlap data conform to a model where the clones are intervals and the STSs are points on the real line. In this case the resulting graph is known as an *interval graph*, and contigs correspond directly to the connected components. One can use elementary graph methods to find the connected components, and then use interval graph methods (Booth and Leuker, 1976; Corneil et al., 1998) to

[1]Department of Computer Science, University of Toronto, Toronto, Ont, Canada M5S 1A4
[2]The Jackson Laboratory, Bar Harbor ME 04609-1500

determine the order of the points and intervals (STSs and clones) contained in each component (contig).

Real mapping data deviates from the ideal in several respects: (1) most egregiously, many YACs are *chimeric*, meaning they contain DNA found in disparate regions of the genome (Green et al., 1991; Haldi et al., 1994); (2) many STSs are *repetitive*, tagging more than one site in the genome; and (3) the mapping procedures generate many false positive and false negative errors. This noise obscures the underlying interval nature of the problem, and gives rise to false contigs that are far too large, and that erroneously connect widely separated parts of the genome. Hence the need for the top-down methods which serve to sort and anchor features of the map along the genome, thus providing guidelines for subdividing components into true contigs.

There is at present no completely automated way to construct large maps from STS-content data, but there are many software packages which do much of the analysis. These systems typically present preliminary maps in visually meaningful ways, and the researcher can then interact with the system, adjusting parameters, throwing out inconsistent data and including new data. Mott and co-workers, mapping the yeast genome, developed several programs to order probes and clones and to display maps (Mott et al., 1993). The program **probeorder** uses a maximum-likelihood measure of separation between pairs of probes and a simulated annealing algorithm to minimize the path through the probes. Two other programs, **barr** and **costig**, order probes using noise-reducing heuristics and tree-search techniques. The heuristics are designed to break up forks in their maps caused by coligated clones, repetitive probes and random false positives. First, probes are flagged as possibly repetitive if they are connected to too many other probes, and the user can interactively make a decision regarding their deletion. Second, two probes are considered neighbors only if they hit more than one clone in common.

The approach taken by CEPH/Généthon for mapping the human genome data is embedded in a compact database and navigation tool called QUICKMAP, developed by P. Rigault and E. Poullier. It 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 (Fluorescent In Situ Hybridization) data (Chumakov et al., 1995).

The WI/MIT group have developed physical mapping software called CONTIGMAKER. Their approach to analysis of the human genome data consists of different steps: (1) link together STSs that share at least two YACs, thus forming *double-linked contigs*; (2) place the contigs in the map by using genetic and radiation-hybrid data; (3) use single linkage where it joins contigs which are genetically close; (4) order the STSs within contigs using a simulated annealing algorithm; (5) add in incomplete addresses that can now be disambiguated; (6) reorder, manually inspect and refine; and (7) fill remaining gaps between contigs with the help of Alu-PCR and fingerprint data (Hudson et al., 1995).

A number of systems for analysis of probe-clone data within a contig have been developed by groups working on smaller genomes or (portions of) single human chromosomes. Examples of these systems are: ODS: a program for quickly ordering random clones into a physical map (Cuticchia et al., 1993); ODS_BOOTSTRAP: a program for quickly ordering clones and for applying a statistical tool to assess reliability of the ordering (Wang et al., 1994); SAM: a system for iteratively building marker maps (Soderlund and Dunham, 1995); SEGMAP: an interactive graphical tool for analyzing and displaying physical mapping data (Green and Green, 1991; Magness and Green, 1996); CONTIG EXPLORER: a package for interactive assembly of STS-content maps (Nadkarni et al., 1995).

The primary goal of these systems is to find a most likely order of the probes and a corresponding placement of the clones. The data are sometimes converted to a weighted intersection graph, where nodes correspond to STSs, edges correspond to clones connecting STSs, and edge weights count the number of clones contributing to each edge. In some of these systems, an initial estimate of probe order is obtained by calculating a maximal spanning tree (MST) of the weighted intersection graph. The initial order is then refined by a simulated annealing algorithm. Other methods of physical map assembly use techniques from artificial intelligence, such as constraint propagation (CPROP) (Letovsky and Berlyn, 1992), and methods based on temporal logic (Lee et al., 1993; Schmeltzer, 1995). Karp and co-workers have developed physical mapping algorithms based on the maximum-likelihood method and on the Hamming Distance Traveling- Salesman Problem (Alizadeh et al., 1993).

In this paper we describe an algorithm which can augment these systems: it provides a preliminary view of the data that helps in the identification of contigs

```
Procedure Structure-Graph(G) ;
Input: graph G = (V,E) ;
Output: Structure graph G' = (V',E') ;
{
  Step 0: b = 0 ;
  Step 1: BFS(G,u), for some u in V ;
          let x denote some vertex in the last layer
  Step 2: S = Make-Blob(x) ; V' = { blob(x) } ;
  Step 3: BFS(G,S) ;
  Step 4: foreach v in V
          { if v is not in a blob { Make-Blob(v) ; V' = V' ∪ {blob(v)} } }
  Step 5: foreach (u,v) in E where blob(u) ≠ blob(v)
          { E' = E' ∪ { (blob(u),blob(v)) } }
  Step 6: return ( G' = (V',E') ) ;

  Function Make-Blob(v)
  Input: Vertex v in V
  Output: a set S_b subset of V
  {
    b = b + 1 ; S_b = { v } ; blob(v) = b ; L = layer(v) ; T = { v } ;
    while T is not empty do {
        Remove a node x from T ;
        foreach neighbour y of x {
            if y is not yet in a blob and layer(y) is L
              { T = T ∪ { y } ; S_b = S_b ∪ { y } ; blob(y) = b ; }
        }
    }
    return ( S_b ) ;
  }
}
```

Figure 1: Structure-graph algorithm

even in the absence of top-down information. This view, called a structure graph, produces an outline of the connected components by compressing local detail in the intersection graph. If the data are free of cross-links among contigs, then connected components will appear as simple, chordless paths, i.e., straight lines. On the other hand, a connected component which is rendered as a branching structure very likely contains several contigs spuriously interconnected. Thus, the structure graph provides a quality check which can reduce the chance of mistaking a complex connected component for a contig. Without this check, one may unwittingly supply an STS ordering system with data for a component composed of interlinked contigs or a contig with spurious self-links. The result would typically be an incorrect order for the STSs. Besides flagging complex connected components, structure graph analysis can also aid in their subdivision into contigs, especially when there is top-down information such as genetic or radiation hybrid positioning of the markers.

System and methods

The STS data were obtained from two sources: Release 10 (May, 1996) from WI/MIT, available by anonymous ftp (genome.wi.mit.edu, directory /pub/human_STS_releases/may96) and the March 1995 Release from CEPH/ Généthon, also available by anonymous ftp (ceph-genethon-map.cephb.fr, /pub/ceph-genethon-map/STS/29MAR95.DAT). We combined WI/MIT data (12,527 STSs and 21,051 YACs) with CEPH/Généthon data (7,026 STSs and 18,298 YACs) to produce a merged data set of 15,136 STSs and 24,855 YACs. The average number of hybridizations ('hits') per STS in the merged data is 6.2 (standard deviation 4.0).

STS-YAC data can be represented as a weighted intersection graph in which each node is an STS and an edge of weight $M > 0$ between two nodes indicates that the corresponding STSs hit M YACs in common. The weighted intersection graph for the merged STS data consists of 665 connected components, one of which contains 94% of all the STS nodes. This anomalous aggregation of most of the nodes into a single component is caused by edges which link STSs belonging to different chromosomes, and it is a well known effect of chimeric YACs. In fact 69% of the unit weight edges link STSs belonging to different chromosomes, and thus do not reflect proximity in the genome. This has motivated the *double-linkage* strategy used by WI/MIT and discussed in (Arratia et al., 1991), where one considers only subgraphs for which $M \geq 2$.

Double-linkage intersection graphs, however, also suffer from cross-linkage of STSs from disparate parts of the genome. As much as 6% of the 38,142 double-linkage edges cross-link chromosomes. Assuming a random chimeric process, we would expect virtually no chimeric edges with weight greater than two, whereas the weights of the cross-linking edges can be as high as $M = 15$, and about one third of them have weight $M > 2$. We suspect that most of the non-unit weight edges that link STSs on different chromosomes result from non-unique STSs, i.e., STSs that hit repeat regions in the genome. As a result of these edges, the double-linkage intersection graph is dominated by one anomalously huge connected component containing one third of all STSs and representing every chromosome.

Thus, although edges with weight $M \geq 2$ are generally reliable, many of these strong edges are misleading. Chimeric addresses, non-unique STSs and false positives imply incorrect proximities in a genomic map. False links between chromosomes (when indicated by the assignment of the STSs involved) are easily detected and removed. False links within a single chromosome are more troublesome. These unwanted edges artificially cross-link widely separated regions of the genome. The next section describes a method that

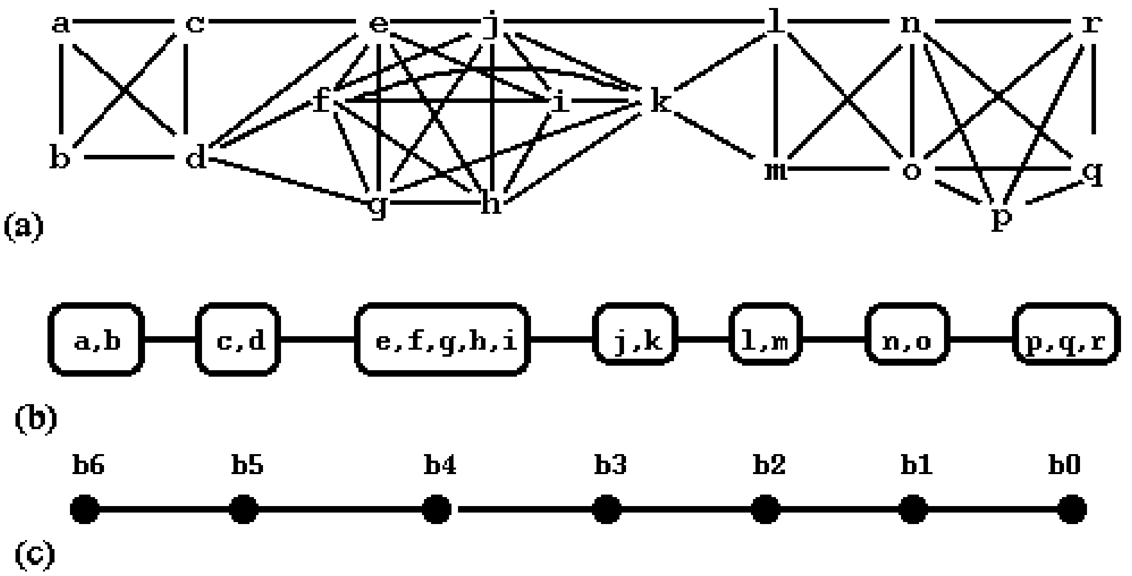

Figure 2: (a) Example of intersection graph and (b),(c) two forms of structure graph.

makes the presence of these cross-links easily visible in many cases, whether they link different chromosomes or distant sites on the same chromosome.

Algorithm

As the amount of mapping data increases, the intersection graphs rapidly become large, unwieldy and extremely complicated. However, much of the complexity arises from small-scale details. In contrast, the large-scale structure of the graph should be relatively simple, in fact, linear, since the chromosome has a linear arrangement. The presence of strong ($M > 1$) spurious edges, however, creates links and loops among what would otherwise be separate components of the intersection graph. As a result, the intersection graph becomes a complex web of components that are difficult to separate and map. We describe here an algorithm for graph abstraction which elucidates this large scale structure.

The simplification is achieved by coalescing nodes of the double-linkage STS intersection graph 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. The method we find to be most insensitive to noise in the data while preserving underlying structure is based on breadth first search (BFS), a classical graph traversal algorithm which partitions nodes into layers $L_0, L_1, \ldots, L_i, \ldots$ according to their distance i from a given starting node (cf. a graph theory text, e.g., (Golumbic, 1980)). We define a *blob* 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. A *structure graph* consists of nodes corresponding to blobs in the intersection graph, with an edge (b_i, b_j) between blobs b_i and b_j if there is an edge in the intersection graph between an STS in blob b_i and an STS in blob b_j.

The algorithm for forming a structure graph $G' = (V', E')$ from an STS double-linkage intersection graph $G = (V, E)$ is shown in Figure 1. It consists of a main procedure and a function, *Make-Blob*. The main procedure invokes the breadth-first search algorithm BFS twice — the first BFS determines an extremity of the intersection graph, and the second BFS defines new BFS layers relative to this extremity. We assume that the number of the layer to which a node $y \in V$ belongs according to the most recent BFS is given by $layer(y)$. The main procedure also calls the function *Make-Blob* once for each blob formed, and constructs the edges of the structure graph. The function Make-Blob(v) determines which vertices $w \in V$ belong to the same blob as v, based on the most recent BFS. It returns the blob as a set, S, and increments a counter b which identifies the corresponding node in the structure graph. Note also that the Make-Blob function defines the association $blob(w) = b$ for each vertex w in the blob, i.e., for each $w \in S$.

As an example, we construct the structure graph for the simple intersection graph shown in Figure 2(a).

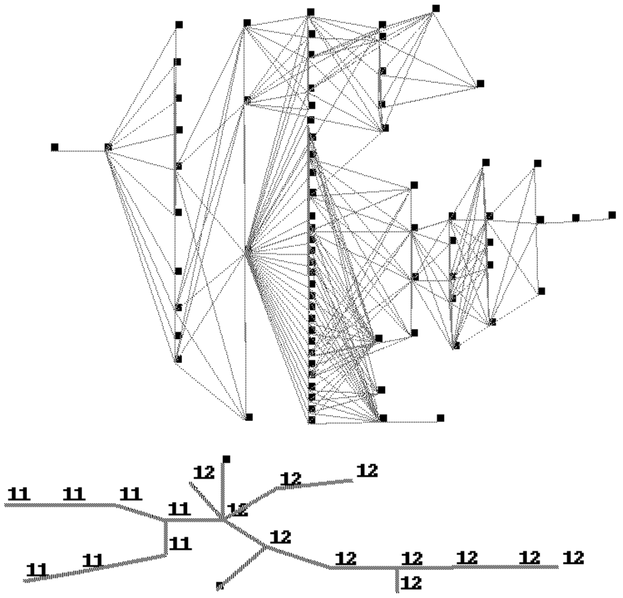

Figure 3: A small component of the merged data STS intersection graph (*top*) and its corresponding structure graph (*bottom*).

Arbitrarily choose starting node $u = h$ in Step 1. The BFS layers are then $L_0 = \{h\}$, $L_1 = \{e, f, g, i, j, k\}$, $L_2 = \{c, d, l, m\}$, $L_3 = \{a, b, n, o\}$, and $L_4 = \{p, q, r\}$. The last node visited in the BFS could be any of p, q, or r; let it be $x = r$. Step 2 forms the first blob as $S = blob_0 = \{p, q, r\}$. The second BFS layering performed in Step 3, BFS(G,S), starting at the set $\{p, q, r\}$ produces $L_0 = \{p, q, r\}$, $L_1 = \{n, o\}$, $L_2 = \{l, m\}$, $L_3 = \{j, k\}$, $L_4 = \{e, f, g, h, i\}$, $L_5 = \{c, d\}$, and $L_6 = \{a, b\}$. Each of these layers is internally connected, so the blobs formed in Step 4 are the same as the layers of Step 3, i.e., $blob_i = L_i, i = 0, 1, \ldots, 6$. Step 5 forms the edges of the structure graph by connecting blobs which contain adjacent nodes. For example, $(blob_0, blob_1)$ is an edge in the structure graph since (r, o) is an edge in G, and $r \in blob_0$ and $o \in blob_1$. Figure 2(b) shows the resulting structure graph, representing the blobs as groups of explicitly listed STSs. Figure 2(c) shows another representation of structure graph which displays blobs as simple nodes, while hiding the contents. The latter representation is used in other figures of this paper.

Comparing the graphs in Figure 2, one can see that 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 by compressing local complexity. The theorem below states that the structure graph is a simple linear sequence of blobs for any intersection graph of noise-free STS-YAC data.

Theorem 1 *The structure graph derived from a connected component of an STS intersection graph for perfect STS-YAC data by the Structure Graph Algorithm is a simple chordless path.*

Proof: We only sketch the proof of this theorem, since the result follows almost directly from the theory of unit interval graphs (Corneil et al., 1995). First, a graph is a *unit interval graph* if each vertex can be identified with a unit interval along the real line such that there is an edge between two vertices if and only if the corresponding intervals overlap. A theorem of Roberts (Roberts, 1968) cited in (Corneil et al., 1995) states that a graph G is a unit interval graph iff there is an order $<$ on the vertices such that for each vertex v, the set comprising v and its neighbors in G is consecutive with respect to this order. This clearly holds for a perfect STS intersection graph if we let $<$ be the genomic order of the STSs. Therefore, the STS intersection graph for ideal data is a unit interval graph. Second, because it is a unit interval graph, Proposition 2.1(2) and Theorem 2.3 of (Corneil et al., 1995) (with minor modifications) imply that each BFS layer produced in Step 3 of the Structure Graph Algorithm applied to the intersection graph for perfect data is connected (in fact, each layer is a clique). Third, Theorem 1 follows easily now, since in this case each BFS layer forms a single blob, and the BFS layers by definition are joined in sequence.

According to this theorem, deviations in a structure graph from a simple path similar to that of Figure 2(c) 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 small branches typically of length 1 or 2, whereas non-unique STSs and pairs of chimeric YACs or false positives linking unrelated STSs can produce long branches. Thus, the main advantage of representing data as a structure graph is to verify whether or not the data are reasonably 'clean' and free from improper cross-links among contigs or within a contig.

Results

Figure 3 shows a small connected component from the double-linkage STS intersection graph for the whole genome and its corresponding structure graph. The graphs are displayed using the Hy+ data visualization system (Consens, 1994). This example is typical in

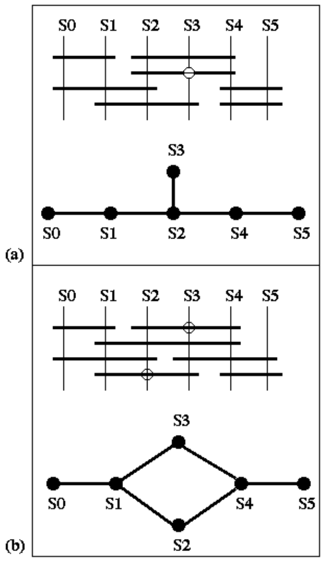

Figure 4: Effect of false negatives on structure graph.

that the structure graph has on the order of twenty-fold fewer edges (20 vs. 534) than the original graph and three or four-fold fewer nodes (21 vs. 75). This condensing of nodes and edges simplifies the representation of the graph, but more importantly, the shape of the structure graph says something about the coherence of the data. When the shape is branching, as in Figure 3, this suggests that the connected component does not represent a single region of a chromosome, but rather two or more regions that are artificially joined. To confirm this inference we have labeled the blobs according to the chromosome assignment of the majority of STSs in each blob. An unlabeled square indicates a blob for which there is no majority chromosome assignment. This connected component appears to be composed of a segment of chromosome 11 cross-linked with a segment of chromosome 12. Larger components have many branches and loops, while most paths between branch points are homogeneous in chromosome content. Thus, double-linkage connected components often appear to be composed of a number of artifactually linked contigs. In the absence of obvious warning signs like changes in chromosome assignment, a branching or looping structure-graph is the main indication of cross-links in the data.

Structure graphs can be further simplified by removal of branches of unit length and small cycles. The rationale for this simplification is that these deviations from linearity are most likely caused by false negatives, which are not relevant to structure. Moreover, some graph layout algorithms exaggerate the length of edges in short branches or cycles, making them visually distracting. This simplification is illustrated in Figure 4. Figure 4(a) presents a model of STS-YAC data (vertical bars represent STSs; horizontal bars represent YACs; and uncircled intersections of bars represent hits) which includes a false negative (the open circle) for STS S3. This gives rise to a branch of length one in the structure graph shown under the model. Figure 4(b) presents a model which includes false negatives for STS probes S2 and S3. The two false negatives cause a small cycle in the corresponding structure graph. (In these two examples, the structure graphs are isomorphic with the corresponding double-linkage intersection graphs, so each blob contains a single STS.) A simple algorithm (not shown) can be used to *trim* away the branches of length one. The structure algorithm can be modified slightly so that nodes S2 and S3 form a single blob, thus compressing these small cycles.

Figure 5 provides an example of trimming unit length branches and compressing small cycles. From left to right the graphs in this figure are: (*i*) the largest connected component in the double-linkage intersection graph for STSs assigned to Chromosome 21 or unassigned (263 nodes, 1593 edges); (*ii*) the corresponding structure graph (49 nodes, 49 edges); (*iii*) the result of trimming branches of length one (38 nodes, 39 edges); (*iv*) the result of compressing small cycles (36 nodes, 35 edges). This sequence of graphs illustrates a progressive reduction of detail leading to a graph which clearly reveals whether the structure is branching, complex, or straight. In this case the final graph is a simple path, consistent with the component being a single contig. (When possible, a linear component like this should also be checked using positional information as discussed below.) Small cycles are less common than unit length branches, so unless otherwise specified, we use the unmodified structure algorithm of Figure 1, and trimming refers only to removal of unit length branches.

As an example of a *complex* connected component from a *single* chromosome, we show the largest connected component for Chromosome 5 in Figure 6. The double-linkage intersection graph (134 nodes and 523 edges) for the component appears in top part of this figure, and the structure graph (26 nodes and 26 edges) with branches of length one removed is shown below. Comparison of these two graphs illustrates the usefulness of this technique of graph simplification: there is a five-fold reduction in the number of nodes, a twenty-fold reduction in the number of edges, and the branched and looped structure clearly reveals the presence of several contigs.

To confirm this inference we have annotated blobs

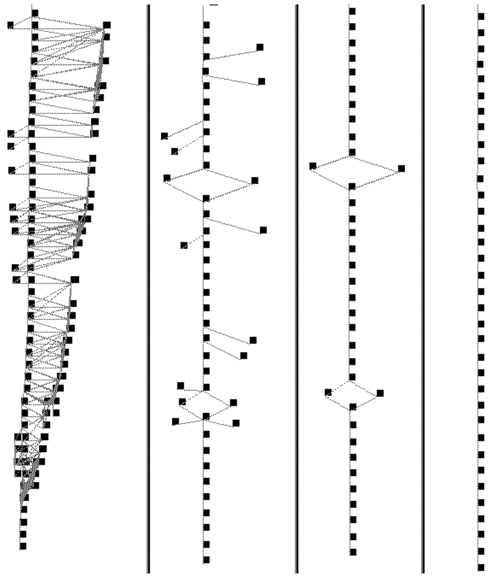

Figure 5: A component of Chromosome 21: STS intersection graph, structure graph, trimmed structure graph, trimmed structure graph with small cycles removed

with average genetic positions according to mapped STSs within blobs. (Average radiation hybrid positions are omitted for clarity, but support the conclusions we derive from genetic positions.) Matching branches with similar positions, three paths stand out as potential contigs: P1 = A-B-C-J2-J1-D-E-F-G-H-I-K; P2 = S-T-U-V-J2-W-X-Y; and P3 = L-M-N-O-J1-J2-P-Q-R. The path P1 is outlined in bold in the structure graph.

To decide if a path in a structure graph corresponds to a contig, one must have sufficient positional information to assess (1) whether there is a relatively smooth and reasonable rate of change of position along the path, and (2) that there are no other paths (in this or other components) with positions that fall inside the range of the path in question. Regarding point (1), consider the rightmost STS in one blob and the rightmost STS in an adjacent blob, where 'rightmost' is defined by position along a chromosome. The separation of these two STSs is at most the length of a clone. Thus, for the YAC clones in this data, the average separation of blobs should be no more than approximately 1 mb. A length of one megabase corresponds roughly to a genetic map separation of about 1 centiMorgan (cM) or a radiation hybrid map separation of about 4 centiRays (cR) (*cf.* WI/MIT Release 10 README.html), although the relationship among these measures varies throughout the genome. Omitting details of the analysis, path P1 (defined above) meets these criteria for

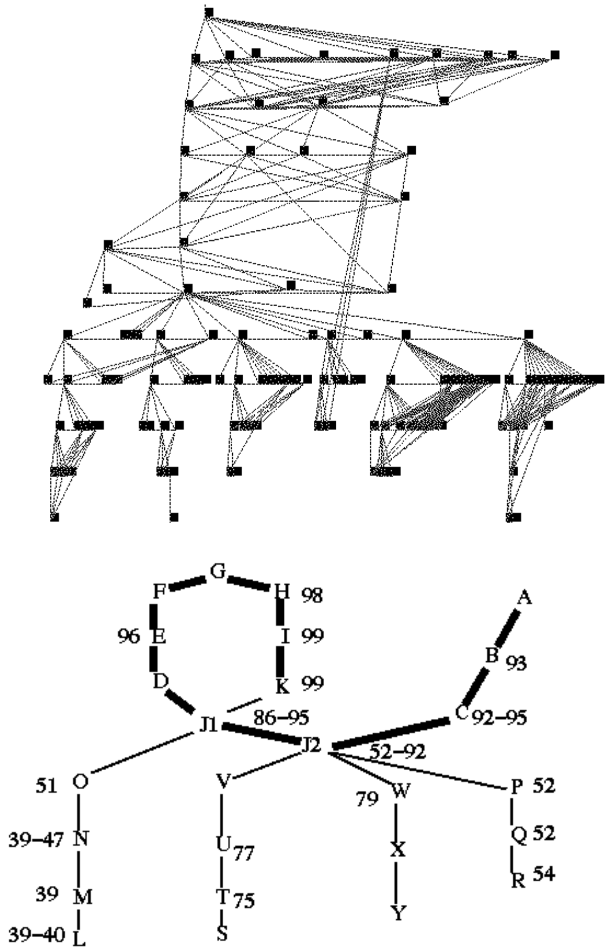

Figure 6: STS intersection graph of one component of Human Chromosome 5 (*top*), and structure graph (*bottom*).

being a contig, whereas paths P2 and P3 do not.

Analysis of this complex component of Chromosome 5 clearly illustrates three features of structure graphs: (*i*) they provide a visual aid in identifying errors and anomalies in physical mapping data; (*ii*) they provide a means to distinguish between *connected components* and *contigs*; and (*iii*) they help in extracting contigs from the much larger and more complex STS-intersection graphs. Thus, structure graphs simplify the construction of physical maps, and can improve their completeness and accuracy.

Discussion

This paper presents a new way to look at physical mapping data. The view is immediately informative — a difference in shape as obvious as the difference between a straight line and a branching line tells the observer whether or not the data are close to ideal. The view is easy to generate — it consists of two breadth-first search (BFS) traversals of an intersection graph, along with grouping of nodes which are connected within a BFS layer. Theorem 1 shows that ideal data from a linear chromosome is transformed in this way into a 'structure' graph which is linear. The linearity is robust to false negatives, which typically either have no effect or create very short branches or small cycles. However, false links within or among contigs are easy to spot in the structure graph, since they typically cause branches or loops. Paths corresponding to contigs can be identified in the structure graph, with the help of genetic or radiation hybrid mapping data. The structure graph method is applicable not only to STS-YAC data, but also to any form of physical mapping data which can be translated into intersection graphs. In (Harley et al., 1996) we present such a translation for Alu-PCR and fingerprint data.

For some types of data (e.g., cosmid overlap data), cross-linkage among contigs may be rare, permitting the assumption that a connected component in the intersection graph corresponds to a contig. However, for the STS-YAC data described in this paper, it is often the case that a connected component in the double-linkage (or even higher-linkage) intersection graph is not a contig. In fact, many of the early 'double-linkage contigs' in the WI/MIT human genome physical map were not true contigs, but rather complex components of interlinked contigs. They overcame this problem by collecting extensive radiation hybrid mapping data (Hudson et al., 1995). Interlinkage of contigs was also a problem in mapping the yeast genome (Mott et al., 1993), where components were subdivided into contigs using heuristics to identify weakly connected regions and repetitive markers. In this paper we propose structure graph analysis as another tool for contig identification, verification and extraction. As a verification step, the method does not require marker position data, and unlike ad hoc heuristics it has a firm theoretical foundation. As the approach is independent of previous methods, it can be used alone or in conjunction with them to improve overall efficacy. The benefit is that structure graph analysis reduces the risk of erroneously identifying a complex connected component as a contig, even in the absence of clues from external positioning data. It therefore provides a useful filtering step to screen out complex connected components prior to the application of an STS ordering algorithm.

Contig extraction using structure graphs is not automatic, but rather requires the involvement of a human expert. It remains to be seen whether the method can be made sufficiently robust as to be fully automated. We are frankly dubious that this will be possible: the data are so noisy that it seems inevitable that human judgment will be necessary to resolve conflicts in the data. The solution, of course, is to improve the quality and quantity of the data. In the meantime, practical application of our method (or any other map analysis method) will require the development of software tools suitable for use by human, mapping experts.

Acknowledgements:

We gratefully acknowledge the assistance and expert advice of the following people: Lincoln Stein at the Whitehead Institute/MIT Center for Genome Research; Derek Corneil, Alberto Mendelzon, Dimitra Vista, Gloria Kissin at the University of Toronto; Mariano Consens at the University of Waterloo, and anonymous reviewers. Special thanks to Derek Corneil for pointing out the connection between Theorem 1 and the theory of unit interval graphs.

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.

Booth,K.S., and Lueker,G.S. (1976) Testing for the Consecutive Ones Property, Interval Graphs, and Graph Planarity Using PQ-Tree Algorithms *Journal of Computer and System Sciences* 13:333-379.

Chumakov,I.M., Rigault,P., Le Gall,I., Bellanné-Chantelot,C., Billault,A., Guillou,S., Soularue,P., Guasconi,G., Poullier,E., Gros,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. Thesis, University of Toronto.

Corneil,D.G., Kim,H., Natarajan,S., Olariu,S., and Sprague,A.P. (1995). Simple linear time recognition of unit interval graphs. *Information Processing Letters 55*, 99-104.

Corneil,D.G., Olariu,S., and Stewart,L. (1998). The Ultimate Interval Graph Recognition Algorithm? *Proc. of the 9th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 175-180.

Cox,D.R., Burmeister,M., Price,E.R., Kim,S., and Myers,R.M. (1990) Radiation Hybrid Mapping: a somatic cell genetic method for constructing high-resolution maps of mammalian chromosomes. *Science* 250: 245-250.

Cuticchia,A.J., Arnold,J., and Timberlake,W.E. (1993) ODS: ordering DNA sequences — a physical mapping algorithm based on simulated annealing. *CABIOS* 2: 215-219.

Dib,C., Sabine,F., Fizames,C., Samson,D., Drouot N., Vignal,A., Millaseau,P., Marc,S., Hazan,J., Seboun,E., Lathrop,M., Gyapay,G., Morissette,J., and Weissenbach,J. (1996) A comprehensive genetic map of the human genome based on 5,264 microsatellites. *Nature* 380, 152-154.

Gyapay,G., Morissette,J., Vignal,A., Dib,C., Fizames,C., Millasseau,P., Marc,S., Bernardi,G., Lathrop,M., and Weissenbach,J. (1994) The 1993-94 Généthon human genetic linkage map. *Nature Genetics* 7: 246-339.

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

Green,E.D., and Green,P. (1991) Sequence-tagged site (STS) content mapping of human chromosomes: theoretical considerations and early experiences. *PCR Methods Appl*, 2: 77-90.

Green,E.D., Riethman,H.C., Dutchik,J.E., and Olson,M.V. (1991) Detection and Characterization of Chimeric Yeast Artificial-Chromosome Clones *Genomics*, 11: 658-669.

Haldi,M., Perrot,V., Saumier,M., Desai,T., Cohen,D., Cherif,D., Ward,D., and Lander,E.S. (1994) Large Human YACs Constructed in a rad52 Strain Show a Reduced Rate of Chimerism. *Genomics*, 24: 478-484.

Harley,E., Bonner,A.J., and Goodman,N. (1996) Good Maps Are Straight. In *Proceedings of the Fourth International Conference on Intelligent Systems for Molecular Biology*, AAAI Press, 88-97.

Hudson,T.J., Stein,L.D., Gerety,S.S., Ma,J., Castle,A.B., Silva,J., Slonim,D.K., Baptista,R., Kruglyak,L., Xu,S.H., *et al.* (1995) An STS-Based Map of the Human Genome. *Science* 270: 1945–1954.

Alizadeh,F., Karp,K., Newberg,L., and Weisser,D. (1993). Physical mapping of chromosomes: A combinatorial problem in molecular biology. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 371-381. ACM Press.

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.

Magness,C. and Green,P. (1996) SEGMAP User's Manual, Version 3.48. Copyright 1994-1996.

Mott,R., Grigoriev,A., Maier,E., Hoheisel,J., and Lehrach,H. (1993) Algorithms and software tools for ordering clone libraries: application to the mapping of the genome of Schizosaccharomyces pombe. *Nucleic Acids Research* 8: 1965-1974.

Nadkarni,P.M., Banks,A., Montgomery,K., LeBlanc-Stracewski,J., Miller,P., and Krauter,K. (1996) CONTIG EXPLORER: Interactive Marker-Content Map Assembly. *Genomics* 31: 301-310.

Roberts,F.S. (1968) *Representations of indifference relations.* Ph.D. Thesis, Stanford University.

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

Schuler,D., Boguski,M.S., Stewart,E.A., Stein,L.D., Gyapay,G., Rice,K., White,R.E., Rodriguez-Tome,P., Aggarwal,A., Bajorek,E., *et al.* (1996) Gene Map of the Human Genome. *Science* 274: 540-546.

Soderlund,C. and Dunham,I. (1995) SAM: A system for iteratively building marker maps. *CABIOS* 11:645-655.

Wang,Y., Prade,R.A., Griffith,J., Timberlake,W.E., and Arnold,J. (1994) ODS_BOOTSTRAP: assessing the statistical reliability of physical maps by bootstrap resampling. *CABIOS* 6: 625-634.

Weissenbach,J., Gyapay,G., Dib,C., Vignal,A., Morissette,J., Millasseau,P., Vaysseix,G. and Lathrop,M. (1992) A second-generation linkage map of the human genome. *Nature* 359: 794-801.