Visualization and Database Tools for YAC and Cosmid Contig Construction

S. W. Thomas1,2, E. A. Rundensteiner3, and A. J. Lee3

1Human Genome Center, 2MCIT, 3Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109

Abstract

Recently, construction of physical maps based on YAC and cosmid contigs has become increasingly common, since these maps are a fine-scale ordering basis for gene searches. We describe a combination of visualization and database tools that address this problem. The YAC-Cosmid contig tool is a map visualization and editing tool designed for large data sets. The Physical Map Assembler (PMA) uses a rule-based interval-ordering approach to infer ordering from experimental data. It is implemented as an active object-oriented database. The PMA visualizer displays interval orderings in a comprehensible fashion, and thus helps the scientist resolve conflicts and ambiguities identified by PMA. Both visualization tools let the user query the database to see the details behind the picture. We have found that a tight integration of visualization and analysis tools with the database enhances their usability, and speeds the dual tasks of analyzing and understanding the experimental data. The use of our tools is illustrated by examples from a gene search at the University of Michigan Genome Center.

1. Introduction

Construction of physical maps based on YAC and cosmid contigs is an increasingly common task in many research labs [4, 16]. Such maps serve as a fine-scale ordering basis for gene searches, and can provide genomic material for such searches and for large-scale sequencing efforts. We describe a combination of visualization and database tools directed at aiding solution of this problem, by either automating map construction or by providing understandable visualizations to the scientist. We have found that completely automatic contig construction is not achievable, given the ambiguous and incomplete data typical of this task. A combination of visualization, analysis, and map generation software, integrated with a database, is the key to a fruitful map construction environment. Use of the tools is illustrated by examples from an on-going gene search at the University of Michigan Genome Center. We want to stress that a tight integration of tools with the database enhances their usability, and speeds the dual tasks of analyzing and understanding the experimental data; and that visualization with interactive editing and query capability is essential to this task.

Visualization lets the investigator easily see what is, at heart, geometric data. We use it: (1) to get a first look at large quantities of raw data, and to interactively order it roughly; (2) to look at intermediate steps in the map-building process, so as to guide further experiments to those areas that are unresolved; (3) to see places where automatic processing has found conflicts in the data, and to resolve them if possible; and (4) to draw a "final" map.

The development of our tools and data model are strongly influenced by the methods used in the local gene search effort. In brief: YACs were isolated using STS markers ordered by genetic linkage and FISH mapping. Alu PCR products from the YACs were screened on a grided chromosome-specific cosmid library. The positive cosmid clones were ordered into "bins" using the YC tool, described in Section 3. In the process, the YAC contig was closed, by considering common cosmid hybridizations between YACs. Cosmid clones from the bins were then directly hybridized to determine cosmid overlaps, and cosmid contigs were constructed.

However, the use of clone-clone hybridization to find overlaps meant that tools based on STS content analysis performed poorly on the data. There are at least two reasons for this. Firstly, such tools (including our YC tool) generally assume that the probes and targets are distinct entities, so that reordering one set has no (direct) effect on the other. This is clearly not true when analyzing cosmid cosmid hybridization patterns. Secondly, STS probes can be treated as points that are contained in the clone targets. and can be unambiguously ordered using a less-than relationship. A clone probe and its target may overlap in a variety of ways, ranging from containment of the probe by the target to the inverse: containment of the target in the probe. The probes themselves may overlap, and so cannot be directly ordered from left to right. We present a solution using a novel visualization approach based on the overlap refinement hierarchy.

In this paper, we describe two complementary software tool sets for contig construction: the YAC-Cosmid (YC) contig builder, and the Physical Map Assembler (PMA). The tools, composed of integrated contig assembly, visualization, and database software, target different aspects of the contig assembly problem.
The first tool, YC, works well with large YAC-cosmid hybridization and STS content data sets. It allows the user to dynamically "drag" the probes and target clones, rearranging their order to quickly produce a rough map.

The second tool, PMA, incorporates a rule-based interval-ordering approach [10] to automatically infer ordering based on experimental overlap data. PMA is implemented using an active object-oriented database [12, 13]. The PMA visualizer displays interval ordering in an easily comprehensible fashion, and thus lets the scientist resolve conflicts and ambiguities discovered by PMA. Both visualization tools also let the user easily query the database to see the details behind the picture.

The remainder of this paper is structured as follows. Section 2 gives a quick introduction to our data model. This lays the foundation for our visualization and database tools. Section 3 then presents the YAC-Cosmid contig assembly tool designed for quickly building YAC-cosmid contigs, while Section 4 describes the Physical Map Assembler tool developed for automating map construction on arbitrary overlap data sets, in particular cosmid-to-cosmid experiments. Finally, the visualization approach and associated tool developed for the Physical Map Assembler system are described in Section 5. In Section 6, we briefly present related work. Section 7 concludes with a summary and an outline of future research directions.

2. The Data Model

The object-oriented model was developed to capture data relevant to the physical mapping task. The model (Figure 1) can be roughly divided into two parts. The left hand side covers concepts related to the genomic experiments. The right hand side covers concepts related to generating and maintaining physical maps.

The object model for genetic experiments includes the classes Library, Probe, and Clone. A library is a collection of unordered clones, contained in wells. A well can hold zero or more clones (usually one). A clone is a colony of organisms with identical genetic makeup, or the DNA fragment(s) of interest from it. A clone may reside in one or more wells, because library construction is a random process. A clone can be refined into classes such as YAC (Yeast Artificial Chromosome) and Cosmid. The Experiment class is an association among the Probe, Library, and Clone classes. In any experiment, a probe is used against a target library, and the result is a set of clones that react positively to the probe.

The concept of ordering and orientation is fundamental to contig assembly, especially that of relative ordering. When a chromosome is broken into pieces, we cannot tell which way a DNA fragment is oriented with respect to the chromosome. Furthermore, the experiments are done on a collection of DNA fragments, and we thus can determine their ordering only relative to this collection. In short, the relative ordering information we get from experiments may not be orientable with respect to the global ordering viewed at the chromosome level. Mapping tools thus must handle the impacts of a local orientation on the overall ordering strategy.

On the ordering side of the model, DNA Fragment and Map are subclasses of Map Element. Since Map is a subclass of Map Element, we can construct complex maps, with elements that are themselves (sub)maps. A Local Orientation Frame (LOF) is used to group a set of ordering relationships that are known to have the same orientation. A map contains a set of map elements and the ordering relationships between them, grouped into LOFs. A DNA fragment is a contiguous part of a chromosome, while a clone is a contiguous sequence of base pairs. A clone could be composed of several DNA fragments in an ordered set (e.g., from internal deletions). Until proven otherwise, we assume a clone is a single DNA fragment. A contig is a set of contiguous overlapping DNA fragments. It can be viewed as a map or as a DNA fragment. The binary ordering relationship between two DNA fragments is represented explicitly.

3. YC: Dynamic contig building

The YAC-Cosmid (YC) contig builder is a dynamic graphical tool for ordering YAC-cosmid hybridization results to build a YAC contig, and to group the cosmids into bins according to this contig. Further experiments can then order the cosmids within the bins to produce a cosmid contig. A snapshot is shown in Figure 2.

The experimental data for which it was designed is derived as follows: A chromosome-specific cosmid library is screened by hybridizing it with Alu-PCR product from a single YAC. The cosmid library is also screened with previously ordered genetic markers, to provide a gross initial ordering.

3.1. YC Tool Operation

The user starts by selecting a set of probes from the database. YC automatically selects all cosmids that reacted positively with the probes. The cosmids are then grouped into bins, such that all cosmids in a bin are positive for...
the same set of probes, as these cosmids are indistinguishable by the selected probe set, and so may be treated identically by the program. This increases the program’s efficiency, simplifies the display, and reduces the number of elements with which the user must deal. If the user later selects additional probes, some bins may be split into smaller bins.

A rough initial ordering of the bins (called ‘segments’ by the program) is produced by assuming that the probes were selected in their correct order.

Looking at the display, we see a list of probe names (“anonymized” for confidentiality of ongoing research projects) down the left hand side, and a list of segment sizes and numbers across the top. Probe names are tagged with the probe type, such as ‘Y:’ for YACs and ‘S:’ for STSs. Each segment is drawn with thick horizontal line segments in line with the probes for which it is positive. The horizontal segments are joined by a thin vertical line to give visual continuity. The length of the horizontal segment is proportional to the number of cosmids it represents — if the density of cosmids is constant along the chromosome, then the length of a clone probe should be roughly proportional to the number of cosmids it hybridizes with. The visual effect is such that when the segments are properly ordered, each YAC probe appears as a continuous thick line. The visual representation of STS probes is less intuitive; a probe that hits several clones (e.g., S:1) spans an interval in this case the width is indicative of the depth of the cosmid library.

Once the initial display has been drawn, the user can interactively rearrange the segments and probes, until a proper contig ordering is achieved. Both segments and probes can be selected and “dragged” to new positions.

Probes can be removed from the display, and new ones added from the database at any time. The segment list is recomputed, such that the ordering is minimally disturbed.

3.2. Interactive Database Queries

The data “behind” the display can be interrogated by clicking on a display element (lines, probe names, etc.). For example, querying a segment pops up a list of the cosmids in that segment. Selecting cosmids in this list brings up another window that lists the probe(s) that reacted with those cosmids, including some details about the experiment. The list also includes probes that were not selected for display, providing a convenient way to choose probes to expand or refine the contig. An example is included in the project walkthrough, below.

In short, this “dynamic visualization” is a powerful tool to better understand and analyze the experimental data and results, while maintaining a direct connection with the underlying data. Having information about the experimental data at one’s fingertips while interacting with the visualization increases the effectiveness of the analysis. For instance, noting the type of experiment that produced the data indicating a particular overlap could be an important factor in deciding which experimental data is likely to be faulty when faced with contradictory data.

3.3. Imperfect Data

YC uses the computer to keep track of the interactions between tens of probes and hundreds of clones, while harnessing the pattern recognition power of the human visual system to make the judgments necessary to create
an ordering from imperfect data. The experimental data is not flawless. On the contrary, both false negatives and false positives are relatively common. False positive results can come from low-copy repeat sequences, contaminated wells or filters, and other sources. False negatives may result from uneven amplification of the Alu PCR product, methylation of primer sites in some YACs, internal deletions in clones, etc. There is also the ever-present possibility of human error.

The data shown in Figure 2 is a typically “noisy” data set, in a plausible ordering. This sort of data is highly resistant to automated ordering methods, without some sort of human intervention to “clean” it up. For example, the Mott and Grigoriev tools [16] computed orderings that we knew were incorrect from genetic and FISH marker data. Automated methods can be used to generate a rough ordering, which is then interactively edited with our tool. The human user, applying intuition and experience, can make judgments that would be difficult to program. In short, a combination of automated ordering and visualization with manual editing represents the most powerful mix for generating good maps.

3.4. YC Implementation

YC is implemented in about 1500 lines of Tcl/Tk code [17, 18]. Tcl/Tk is a “freeware” programming system, developed at the University of California at Berkeley, and is a good environment for prototyping and developing interactive graphical programs.

3.5. Project Walkthrough

The YC tool was used in a local gene search effort. Starting with a few YAC probes on a cosmid library, we began building a rough map as shown in Figure 3.

There are four apparent contigs in this map (0–22, 23, 24–26, and 27–41), so we will concentrate on trying to extend the ends of those (e.g., between segments 22 and 23). We look to see if there are additional hybridization results in the database, by selecting segment 22 (positive only YAC Y:16) and opening the information window. The contents of the information window are shown below, in a slightly condensed form. Parenthetical remarks are comments further describing the probe. For YAC probes, these list the STSs for which the YAC is positive.

```
Cosmid(s) C:58 C:420 C:423 C:424
C:425 detected by:
YAC Y:16 Jan 29 1993 (S:6)
```

Probes not in display:

```
Feb 1, 1993 Results on Grid 7
YAC Y:129 Feb 10 1993 (S:14, S:20, S:61)
YAC Y:3G8 by Alu/PCR Feb 19 1993 (S:6)
YAC Y:8G8 by Alu/PCR Feb 19 1993 (S:6)
YAC Y:122 by Alu/PCR Feb 19 1993 (S:6)
YAC Y:128 by Alu/PCR Feb 19 1993 (S:6)
```

From this, we note that the YAC probe Y:B12 also reacted positively with some of these cosmids, and that it is positive for the STS probe S:21, so it should close a gap between the S:6 group and the S:21 group. This forces S:25 to move to the right.

To completely close up the contig, it was necessary to create new STSs from a few YAC ends. The map display helped decide which YACs were probably longest, and thus had the best chance of picking up new overlaps. Figure 2 shows the map near the end of the YAC mapping phase.

The next phase involved picking cosmids from the segments identified in the YC map, and hybridizing them to gridded cosmids from the region of interest (as identified by the earlier YAC-cosmid screenings). A YC map from this process is shown in Figure 4. This map covers only a subset of the original region, but at much finer resolution. Since the identities of individual target cosmids are now important, we have turned on display of their names. The cosmid probes are grouped in the top 3/4 of the display, with some YAC and STS probes near the bottom to “anchor” the cosmid contigs.

We note almost immediately that the cosmid-cosmid hybridization results are much less noisy than the YAC-cosmid results. There are still 3 distinct cosmid contigs in this map. We could try to close the gaps by picking cosmids from the bins at the ends of each, and using them to further screen the library.

Figure 3. First YAC-cosmid map.
Figure 4. YC visualization of cosmid-cosmid hybridizations.

This map also illustrates a problem associated with using this type of display for visualizing cosmid-cosmid hybridization results. Near the upper left corner, is an apparent "gap" in cosmid C:41 (4th row). The clones cannot be rearranged to eliminate this gap, without a similar gap appearing in another clone, so this gap apparently indicates a problem with the data. To look more closely, a few clones from the region are shown on the top of Figure 5. However, it is possible to arrange the clones consistently with the hybridization data, as shown at the bottom of the figure (drawn by the PMA visualizer, described in Section 5.)

Figure 5. Seemingly inconsistent hybridization data with a consistent arrangement.

This sort of problem led us to consider the process of ordering overlapping clones in more detail. The Physical Map Assembler and visualizer, described in the next two sections, resulted from these considerations.

In summary, we have found the YC tool to be useful in organizing, selecting, ordering, extending, and refining large contigs based on clone-clone hybridization. It smoothly integrates visualization with human intervention to support map construction. The user can experiment with various orderings, with immediate feedback to assess the impact of a change on the rest of the developing map.

4. The Physical Map Assembler

Simultaneously with the development of the YC tool, we have been studying the more general problem of automatically ordering contigs, given a variety of types of experimental data, such as STS content mapping and clone-clone hybridization, in combination with a priori knowledge of ordering derived from other sources (genetic mapping, FISH, etc.).

4.1. The Interval Overlap Refinement Hierarchy

For this purpose, we have developed an interval overlap refinement hierarchy (Figure 6), which can be used to classify ordering and overlap relationships for genomic intervals. The model is designed to handle ordering information from experiments, from external sources, or that is derived by the system. We describe the ordering model briefly, for a detailed discussion see [10].

The refinement hierarchy is based on an analysis of the types of experiments conducted with DNA fragments, as outlined in Sections 1 and 2. It has several nice properties. (1) It captures both precise and partial information about interval relationships in a single relationship; (2) the representation is closed, that is, the composition of two interval relationships is again one of these relationships; and (3) it handles both overlaps between pairs of intervals as well as between intervals and points.

While some concepts from previous work on temporal ordering [1] can apply to DNA fragment ordering, several significantly different characteristics of the genomic domain drove the development of our overlap refinement model.

Determining the equality of endpoints between DNA fragments is generally costly. Instead, we determine as much as possible about the interval relationships, without necessarily pinpointing endpoint relationships.

Initially imprecise information will gain precision as more experiments are performed. The model is designed to...
Temporal reasoning deals with events and relationships between events, where there is a global orientation. In the genome domain, where we are likely to be able to infer only relative ordering information. The relationships of DNA fragments are specified within a local orientation frame (LOF), as described in Section 2. Any LOF, LOFi, has a mirror image, LOF\textsubscript{i+1}, which can be constructed as needed. An example of this will be discussed in Section 5.

4.2. Ordering and Orientation Inferencing

Next, we present mechanisms to support inferencing based on the interval overlap refinement hierarchy. Our goals are: (1) to keep the number of ordering relationships as small as possible in each LOF; (2) to reduce the number of LOFs whenever possible; and (3) to infer as much new information about fragment ordering as possible. To achieve the first goal, a relationship R\((A,B)\) is removed from a LOF whenever a more precise relationship R\'(\(A,B)\) is added to that LOF. The second goal is addressed by LOF composition operators that merge two LOFs when their orientation relative to each other can be determined. Ideally all ordering relationships could be combined into one LOF with fixed orientation with respect to the chromosome, i.e., the global orientation frame. The third goal achieved by relationship refinement and transitivity rules, that derive new relationships from existing ones.

As an illustration, we present an example of a relationship refinement rule, see [10] for more detail. Relationship refinement rules are used to infer more precise ordering information about a pair of intervals in a single LOF. Each rule considers two relationships in the same LOF with the same pair of arguments. For example (Figure 7), if we have \(\text{not-contained}\sim\geq(A,B)\) and \(\text{not-contained}\sim\leq(A,B)\) in LOFi, then we are able to infer \(\text{not-contained}\sim\geq(A,B)\) and retract the two original relationships from LOFi. The hierarchy in Figure 6 shows the derivation of this rule — the highest common descendant of the relationships \(\text{not-contained}\sim\geq(A,B)\) and \(\text{not-contained}\sim\leq(A,B)\) is \(\text{not-contained}\sim\geq(A,B)\).

Lastly, note that experimental data can easily be expressed in our ordering model. For instance, for each YAC, \(Y\), that is hit by a probe \(P\), we create a \(\text{contains}(Y,P)\) relationship, and for each YAC, \(Z\), that is not hit by \(P\), we create a \(\text{disjoint}(Z,P)\) relationship in a new LOF. Similarly, for a pair of cosmids, \(C1\) and \(C2\), that hybridize, create a \(\text{not-disjoint}-1(C1,C2)\) relationship.

4.3. Crystal: Active Object-Oriented Database.

In order to build a tool that realizes the PMA model described above, we need to incorporate both data modeling and inferencing into one integrated system. We have thus designed and implemented an active object-oriented database system, which incorporates production rules and associated inference processing directly into the OODB. Activities can be represented directly in a declarative format in the database by treating rules as objects. Given that the database is in control of both rule and data management, it can effectively optimize rule processing. This is an increasingly critical issue given the expected size of genomic databases [5].

A detailed design and implementation description is given in our technical report [11]. We summarize here some of the key concepts.

We have chosen GemStone\textsuperscript{1} as implementation platform for this purpose, since it is a mature OODB product and since it has been adopted by other genome researchers. Our active OODB system is called Crystal.

Crystal currently supports classical Condition-Action (CA) rules, which are sufficient for the rules in the PMA model. Crystal treats rules as objects [2], which allows them to exist independently of other objects. Therefore, in Crystal, rules can be created, modified, and deleted at runtime like any other object. Furthermore a rule can monitor instances belonging to multiple classes.

Given large data sets, we need to have an efficient pattern matching algorithm to find which rules have their condition parts satisfied. In Crystal, we have an active rule evaluation strategy based on the following concepts: (1) each rule subscribes to the classes that it needs to watch, (2) monitored objects in the subscribed classes send signals to relevant rules when relevant changes are taking place, and (3) rules check their condition parts only when notified. These mechanisms allow for efficient rule processing over large databases, when the number of simultaneous, relevant database updates are small, as in the map construction task. [3]

In Crystal, we also adopt an incremental rule evaluation approach, similar to TREAT [15], to reduce the time spent on re-evaluating predicates. We store IDs of the objects that partially satisfy each predicate. This is used to incrementally update the status of the rules to determine when a rule is ready to be fired.

\textsuperscript{1}GemStone is a registered trademark of Servio Corporation.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{figure7.png}
\caption{Example of a Relationship Refinement Rule}
\end{figure}
We have built Crystal as a layer of active system classes on top of the system class hierarchy of GemStone. The ProductionRule class, a subclass of the new ActiveObject class, is used to support the storage, addition, modification, and deletion of the classical Condition-Action (CA) rules. It can subscribe to the classes that it needs to watch, and react to relevant event signals that are sent by these classes. Its component Predicate objects maintain the information needed for incremental rule evaluation.

4.4. Building the Physical Map Assembler Tool Using Crystal

We have built the Physical Map Assembler tool on top of Crystal. As application developers, we did not have to worry about the rule processing and event signaling issues, because Crystal provides the active system support for the inferencing capabilities. The resulting system forms a three-layered architecture (OODB, Crystal, PMA).

The data model (Figure 1) now is not only capable of storing raw experimental data and complex derived data, but also directly incorporates the rule processing capabilities for fragment ordering, by virtue of being built on top of Crystal. Note that the OrderingRelationship class must be a subclass of the ActiveObject class in Crystal, because the OrderingRelationship instances generate events that the rules react to.

Since our physical map assembler is built on top of GemStone, all database capabilities of GemStone are readily available. In particular, while GemStone does not provide a high-level query language, it provides rudimentary query mechanisms for searching over collections. The physical map assembler thus easily supports a set of pre-defined user queries, ranging from “Give me all STSs contained in YAC Y”, to “Give me a map between clones C1 and C2.” Of course, other basic database functionality, such as recovery and concurrency, are also available for free.

The maps generated by PMA can be viewed through a visualization tool, called the PMA visualizer, described below. Although PMA can detect conflict data and notify the user, the system does not currently have a conflict resolution strategy built in. Instead, the user uses the visualization tool to view the physical maps to solve the conflicts manually, as further outlined in the next section. Suggestions about resolving conflicts can be fed back into PMA, e.g., by removing some conflicting overlaps from the working set, in order to continue map generation.

5. The PMA Visualizer

Once we’ve computed an ordering using PMA, as described in the previous section, then we need to present the results in a meaningful form. Recall that PMA utilizes a refined interval overlap model (Figure 6) as its internal representation. Furthermore, PMA will typically generate a transitive closure of all interval ordering relationships. For a completely determined ordering of n YACs, this would include n^2 pairwise relationships, resulting in an information overload. Thus, a condensed visualization is important. As we have seen, the matrix-like visualization used in the YC tool does not work well, because it shows apparent conflicts where none are present (Figure 5), and also because it is hard to keep track of the identical elements being used both as probes and targets. We now describe our visualization approach.

5.1. Computing an Order

We must translate the ordering relationships generated by PMA into a visual representation involving intervals with concrete starting and ending points, since, in the two-dimensional medium of the computer screen, each line must begin and end at some pixel. We first translate each interval relationship into endpoint relations (see [10]). For example, not-disjoint(A,B) is translated to start(A)<start(B) and start(A)<end(B) and end(A)>start(B). The resulting set of endpoint relations induces a partial ordering on the endpoints. A topological sort [20] gives a consistent endpoint ordering, when one exists. It can be used to draw a visualization of the results, as illustrated in Figure 8.

This figure shows the results of ordering the “walkthrough” data from [10]. Simulated STS-YAC content mapping data were used as input. The heavy black lines show plausible placements of each of the STS probes (p1-p5) and YAC clones (y1-y8). The endpoint positions of the solid lines are derived directly from the ordinal position of each endpoint in their sorted ordering. The gray ambiguity bars show the range within which each endpoint may vary while remaining consistent with the rest of the data. They are derived by constraining each endpoint by its relationships with the other endpoints in the diagram. Each black endpoint is given a unique horizontal position, but the ambiguity bars remind us that these are, to some extent, arbitrary. Consider, for example, y5 and y6. While they are drawn in quite different locations, we can easily see, from the ambiguity bars, that their positions are constrained identically.
5.2. Conflict Visualization

As a second example, let us consider part of the cosmid-cosmid hybridization example presented in Section 3, illustrated in Figure 9 (generated by YC). We applied PMA to the data set, and it found a conflict when an attempt was made to merge two LOFs, near the end of the processing. Thus, the LOFs were not merged by PMA. There are two possible relative orientations of the two LOFs, i.e., LOF1 with LOF2 and LOF1 with LOF-2. PMA determined that both orientations lead to conflicts.

We now will use the visualization to investigate the possible origin of the conflicts. By combining the relationships from the LOFs in first one relative orientation, and then the other, without trying to infer any new relationships after merging, we can produce two visualizations. The results are shown in Figure 10. A new visual element, a conflict bar, is used to show conflicts in the ordering data. The dashed bars (red in the on-screen display) show endpoint constraint regions that are reversed—the endpoint is constrained to the left of the left end of the dashed bar and to the right of the right end.

For example, the dashed line to the left of c3 in Figure 10a results from the relationships O(c3,c5) and D(c7,c3). O(c3,c5) constrains the left endpoint of c3 to be to the left of the left endpoint of c5, while the D(c7,c3) constrains the left endpoint of c3 to be to the right of the right endpoint of c7. I.e., the left endpoint of c3 should simultaneously be to the left of the bar and to its right. This is clearly impossible.

Conflicting constraints such as these arise when the topological sort process must break loops in the constraint graph in order to produce a linear ordering of the endpoints. Thus, when we see conflict bars, we know that there is no arrangement of the clones that is consistent with the current constraint set. Instead, we need to modify or remove one or more of the constraints, by returning to the experimental data and modifying it. We will use the visualization to investigate the origin of the conflicts, and thus determine how to resolve them.

Note that many of the conflict regions in Figure 10a have one end aligned with the left endpoint of c5. By manually moving this endpoint to the right, we obtain the diagram shown in Figure 11, in which only c1 and c5 have conflicts. The data says they should overlap so that the right endpoint of c1 should be to the right of the left endpoint of c5, but they are not. Armed with this insight, we can re-examine the original experimental data, to find that c5 used as a probe did not react positively with c1, while the reaction between c1 used as a probe on c5 is weak. Removing this positive reaction from the experimental data then allows PMA to compute a single consistent ordering (Figure 12).

5.3. Implementation

The PMA visualizer prototype, with a command-based interface, has been implemented in 700 lines of Tcl/Tk. We anticipate that adding a "GUI" and additional database query functions will bring it to about the same size as the YC tool; i.e., close to 1500 lines.
5.4. Summary

In summary, the PMA visualizer lets us see the results of ordering the clones, to help resolve conflicts in the original data, and to help direct further research using the ordered contig. The PMA visualizer is thus a valuable addition to our PMA system, helping to simplify the process of contig assembly. Furthermore, it complements our YC tool by nicely handling cosmid-cosmid hybridization data, which are not well visualized by YC.

6. Related Work

Letovsky and Bertlyn [13] describe a rule-based system for inferring order among genetic markers, including the concept of local orientation. Guidi, et al [7], present a discussion of issues in inference of ordering in genetic systems, including a review of previous work.

Mott and Grigoriev [16] describe a set of tools for ordering and visualizing contig map data. The automated ordering tools are based on distance-minimization heuristics. They use a flat-file “database.” This enhances the tools’ portability, but loses the generalized query and selection capabilities of a true database. The toolset includes a matrix-like visualization tool. There appears to be no way to override the computed ordering when it is in conflict with a priori ordering data from other sources such as genetic markers and FISH analysis.

SEGMAP [6] is a computer tool for automatically ordering clone maps based on STS content. It does not handle hybridization based screening methods. It too, works from data stored in ASCII files, and so suffers from the disadvantages listed above. As far as we know, it has not been described in the literature.

Pecherer describes a graph-based visualization tool for investigating fingerprint-overlap data for contig construction [19]. It automatically identifies graph features resulting from conflicting data, such loops and 3-way branches, and includes a heuristic ordering algorithm for drawing the overlap graph. It does not draw the more traditional horizontal line segment-based contig maps; its model of interval overlap relations is more limited than ours; nor can it infer new and more precise relationships between the clones in a map. The visualization becomes cumbersome as the number of clones in a contig grows beyond a few dozen.

SIGMA [21] is a tool for graphically arranging and presenting integrated maps, developed at LANL. While it incorporates a notion of overlap of intervals, it does not provide any automated ordering method. We have found it to be good for making “publication-quality” drawings of maps. We adopted a variation on SIGMA’s ambiguity bars in the PMA visualizer.

The Hy-Graph system [9] uses a graph-based visual interface to describe and implement ordering inference rules as graph-pattern queries. It is based on a logic-programming language. Ordering results returned by the graph-based queries are not entered into a database, so manual intervention is not facilitated. They focus on STS-based orderings.

ContigMaker [8, 22] is a logic-based system for inferring clone overlap and partial ordering from STS content mapping. It incorporates the local orientation frame concept and rules for interval relationships. They also discuss visualization methods.

7. Conclusions

We have described a suite of tools, centered around a database, that provide differing approaches to map and contig construction. The YC tool provides a visual method to easily select, view and order many cosmids over a relatively large region. The map in Figure 2, generated by the YC tool, has over 300 cosmids ordered in bins, spanning approximately 2 megabases. The PMA tool and visualizer provide an automated rule-based method for map construction based on an interval-ordering model. The visualizer helps the scientist determine the source of conflicts and propose resolutions to them.

The research presented here has revealed the following key observations:

- The matrix-like representation commonly used to visualize STS-content mapping of clones is leads to erroneous perceptions when applied to hybridization-based results, as illustrated in Figure 5.

- A new visualization approach based on the concept of endpoint-constrained regions successfully addresses this problem, and provides a way of investigating alternative solutions to conflicting data, as seen in Figures 10–12.

- The tools described here each provide different capabilities, together they complement each other in support of the physical map construction and contig assembly tasks.

7.1. Future directions

It would be desirable to persistently maintain the contigs constructed through visualizations as alternative map proposals, without overriding existing ordering relationships stored in the database. We thus plan to experiment with the integration of workspace concepts into our system, as found, for example, in the Genome Topographer [15].

Furthermore, visualization of numerous alternative maps is also a challenging research issue, since of course each decision made while constructing and reordering a contig through a visualization leads to another map. It will be important to clearly depict the difference between consecutive maps, and the underlying assumptions made to construct a particular map.

Closer integration between YC and PMA is desirable. We use YC to select data for PMA, but the ordering computed by PMA can not be easily “fed back” to YC.

Our Physical Map Assembler takes a rule-based approach to clone ordering. In its simplest (and current)
form it requires noise-free data, in the sense that it identifies but does not resolve conflicts when given inconsistent data. It is flexible, in that rules can be added to deal with imperfect data in a variety of ways, for example, by incorporating a "data quality" factor, and/or by using voting-based rules. Further, rules can be based on factors such as the type of experiment, e.g., to "know" that false negative reactions are more likely in an Alu-PCR based screening than in a direct hybridization screening. Formulating rules to deal with experimental error is a major focus of our ongoing research.

Acknowledgement

This research was supported in part by the NIH under grant NIH P30HG00209-03, by the NSF under grant RIA IRI-9309076, and by the University of Michigan Award Program.

References