CONF-940136--5

UCRL JC 117564

UCRL-JC-
PREPRINT

A database system for constructing, integrating, and displaying maps of chromosome 19

T. Slezak, M. Wagner, M. Yeh, L. Ashworth, D. Nelson, D. Ow, E. Branscomb, and A. Carrano

This paper was prepared for submittal to the 28th Annual International Conference on System Sciences, Maui, Hawaii, January 3-6, 1994

June 1994

Lawrence Livermore National Laboratory

A database system for constructing, integrating, and displaying physical maps of chromosome 19

Tom Slezak*, Mark Wagner, Mimi Yeh, Linda Ashworth, David Nelson, David Ow, Elbert Branscomb, and Anthony Carrano

Human Genome Center

Biology and Biotechnology Research Program, L-452

Lawrence Livermore National Laboratory,

7000 East Avenue

Livermore, CA 94550

U.S.A.

*To whom all correspondence should be sent: slezak@llnl.gov, (510) 422-5746, FAX (510) 423-3608

Abstract

Efforts are underway at numerous sites around the world to construct physical maps of all human chromosomes. These maps will enable researchers to locate, characterize, and eventually understand the genes that control human structure and function. Accomplishing this goal will require a staggering amount of innovation and advancement of biological technology. The volume and complexity of the data already generated requires a sophisticated array of computational support to collect, store, analyze, integrate, and display it in biologically meaningful ways. The Human Genome Center at Livermore has spent the last 6 years constructing a database system to support its physical mapping efforts on human chromosome 19. Our computational support team is composed of experienced computer professionals who share a common pragmatic primary goal of rapidly supplying tools that meet the ever-changing needs of the biologists. Most papers describing computational support of genome research concentrate on mathematical details of key algorithms. However, in this paper we would like to concentrate on the design issues, tradeoffs, and consequences from the point of view of building a complex database system to support leading-edge genomic research. We introduce the topic of physical mapping, discuss the key design issues involved in our databases, and discuss the use of this data by our major tools (DNA fingerprint analysis and overlap computation, contig assembly, map integration, and database browsing.) Given the advantage of hindsight, we discuss what worked, what didn't, and how we will evolve from here. As early pioneers in this field we hope that our experience may prove useful to others who are now beginning to design and construct similar systems. This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract No. W-7405-ENG-48.

Introduction to Physical Mapping

The Human Genome Project (HGP) is an ambitious multidisciplinary, international effort to locate, characterize, and understand the estimated 100,000 genes that determine the organization and functions of humans. Genes are regions of DNA which describe proteins. Each protein is composed of a sequence of smaller units called amino acids, each of which is in turn coded in the DNA by a triplet of nucleotides. Genes typically range from a few hundred to tens of thousands of nucleotide "base-pairs" in length, often split into several "coding regions" separated by "non-coding regions" (filler of uncertain purpose.) Over 90% of the total 3 billion base-pairs of the human genome are thought to be non-coding regions. Defects in genes (insertion, deletion, substitution, or rearrangement of nucleotides) may lead to trouble; there are an estimated 5,000 diseases of genetic origin.

Physical mapping is the use of various biological techniques to isolate the location of genes and other markers to specific portions of the 24 chromosomes that comprise the total human genome. The crudest techniques may place a gene only on a specific chromosome, or perhaps on a particular arm or "band" on a chromosome. These regions are still far too large for practical use, since a single band may span several million base-pairs. More sensitive mapping techniques can further refine the location of a gene to pieces of a chromosome of clonable size, such as Yeast Artificial Chromosome clones (YACs, from 100,000 to over 1,500,000 base-pairs in length) or cosmid clones (about 40,000 base-pairs long.) A collection of clones specific to a region of interest is called a library. Unfortunately, the process of making these clonable objects results in a complete loss of order, thus requiring the use of additional mapping techniques to order the objects on which the genes are eventually mapped.

At an even lower level, individual clones can be split into restriction digest fragments that range from 500 to 15,000 base-pairs in length, and genes can then be mapped onto one or more of these fragments. Current DNA sequencing technology can only reliably determine 400-500 base-pairs at a time, so the restriction fragment(s) containing the gene are split still further and sequenced in multiple small pieces, which again require a re-assembly process to restore order. Sequencing is a relatively expensive and time-consuming process, so it is generally highly desirable to sequence only regions of interest.

For some traits or diseases biological "probes" exist that allow the associated gene(s) to be mapped to the various objects described above. These probes often come from genetic studies of families that possess the hereditable disease or trait. The great bulk of the estimated 100,000 human genes have no probes at this time and must be detected via other methods. Establishing a physical map of overlapping clones of various sizes that spans all the linear DNA of each human chromosome will ensure the rapid mapping of any new genetic diseases or traits whenever probes are generated.

It should be noted that there are numerous possible different approaches to constructing a physical map, depending on the exact characteristics of the particular type(s) of clones used and the experimental techniques used to order them and map genes upon them. Approaches which rely on the larger YAC clones as their primary objects are often referred to as "top down" approaches, whereas the use of smaller cosmid clones is a "bottom up" method. Techniques employing new BAC and PAC clones which range from 50–300Kbp (kilo base–pairs) are now being evaluated as potentially more efficient schemes. The HGP is currently employing variations on several strategies as the various biological and computational techniques are being developed and evaluated.

Overview of the LLNL Mapping System

Livermore's current role in the HGP is to provide a physical map of human chromosome 19, which at an estimated length of 60 million base–pairs is one of the smaller chromosomes. Surprisingly, it is also considered to be "gene rich", containing an estimated 2,000 genes. As of this writing, probes are available for ~300 of those genes, most of which have been mapped to cosmid clones in our library. Included in this set of known genes are ones that cause a form of muscular dystrophy, cause a rare form of skin cancer, and control the sense of smell. Livermore focused on chromosome 19 because of prior work on a set of DNA repair genes that exist on chromosome 19. These genes are capable of locating and correcting certain mutations that are due to the effects of ionizing radiation (Weber, 1988). It is anticipated that the remaining estimated 1,000+ genes on chromosome 19 control many other important functions and may lead to significant advancement in human health knowledge, diagnosis, and treatment.

In the past 6 years we have constructed a system to facilitate the acquisition, storage, analysis, querying, and presentation of all our physical mapping data for chromosome 19. The processing of cosmid clone overlap data comprised the bulk of our work in the early phases. We have now "fingerprinted" over 15,000 cosmid clones, analyzed them for overlap, and reassembled 10,424 of them into 802 contigs that are spanned by 3,536 clones (Figure 1). (Some clones were rejected due to data quality problems related to the complexity of the underlying biological techniques. Other clones are "orphans" due to inability to reliably detect overlaps of less than 30%. Many clones in contigs are redundant to the near–minimal subset required to span all the linear DNA represented in the contig). Our efforts now have switched to utilize Fluorescence Insitu Hybridization (FISH) mapping, YAC/BAC/PAC hybridization, EcoRI restriction mapping, and other techniques to both reduce the number of gaps and to order the cosmid clone contigs.

Building a Lab Notebook and Physical Mapping Database

Since starting our human genome database in 1989 we have accumulated nearly two hundred tables containing over 215MB of data and indices. In addition, a separate database contains 1.5GB of raw cosmid clone fingerprint data (Figure 2). We have several SUN workstations with access to the genome databases via our custom browser serving 40 local and several remote end users. Our database has live links to other major genome repository databases (GDB at Johns Hopkins and Genbank at various sites). Our database is implemented using Sybase, a commercial server/client relational database, on a Sun Sparc 2 workstation. All our custom programming uses C, AWK, or PERL on the Unix operating system.

Physical mapping was in its infancy when we started designing our database. New cloning systems, experimental methods, and mapping concepts have been constantly developed over the life of this project. There still is no consensus about what is the necessary degree of resolution needed for physical maps or what level of confidence to place on maps constructed by varying methods. Groups that attempted to apply classical software engineering methodologies appropriate for static problem domains tended to have fared poorly supporting genome physical mapping. We adopted strategies for survival in this chaotic environment, providing useful tools in a timely fashion and measuring our success by that of the biologists we were supporting.

We decided quite early to separate our laboratory notebook database from that needed to produce final, integrated physical maps. This was primarily due to the fact that nobody could define what those final maps should be, and if we didn't solve the immediate experimental data problems we would never have survived to the end. Over 3 years elapsed before matters stabilized to the point where it was possible and necessary to worry about physical map integration.

Laboratory Notebook Database Design

Genome laboratory notebook database development at LLNL is done in an iterative fashion that is dictated by the rapidly–changing needs of the end users, rather than formal design theory. The major steps in the development cycle include defining biological requirements for data tables, table design, conducting client walkthroughs to confirm the design, table implementation, automating data conversion, designing and implementing customized user interfaces, and user training. Automated data conversion is often required because the users don't tell us about new types of data until they are certain that the experiments are working and data has piled up in random files or spreadsheets. We require the users to specify a query usage for every data field that they propose; data that they can't usefully query remains in paper notebooks. We did not attempt to achieve a paperless lab nor did we do explicit formal data or process modeling. Our database schema diagrams (Yeh, 1994) and data dictionary (Ashworth, 1994) are available via anonymous ftp.

Our design effort involves a close collaboration of the scientists who produce and use the data, and the database specialists. The biologists are an integral part of the design of "their" tables, including any iteration steps that may be required. Since they "own" the data and will be responsible for all input of their data, they are highly motivated to get the tables designed properly. A useful side–effect is that our biologists now have acquired a working knowledge of relational databases and can often provide us with good initial designs. We conduct design walkthroughs with all the people affected by the related group of tables, updating our database schema and data dictionary at the same time. When initial consensus is reached, we implement the tables, rules, triggers, indices, and input scripts. If data already exists in files we write a conversion script. In most cases we implement the new tables within a day or two and await user feedback. Often two or three iterations are needed to bring the new tables to equilibrium, until the next change in techniques occur.

Database table design for the cutting edge of human genome research is much different from designing a database for a well–defined business or control system. The end users are constantly creating new objects and techniques. Sometimes they can not predict how future experiments will impact the relations among data presently stored in the database. We have found that our users generally need time to use and understand the data to arrive at the best design. We have evolved techniques and built systems which anticipate frequent change and attempt to minimize its cost.

Although we utilize all the available techniques to ensure database integrity and security (i.e., separate user accounts, field rules, triggers, read–only views, etc.) we found that it is sometimes necessary to break or bend some of the rules of good database design in order to satisfy practical concerns. We are not dogmatic about normalization; it is more important to us that our end users understand how their data is stored and accessed and that they are motivated to use the system. We also have found it necessary for performance reasons to construct some de–normalized tables to prevent certain multi–table joins from causing unacceptably slow response.

We use several different ways to enter human chromosome 19 data into database tables. Many of our analysis programs input directly to the database. We wrote our own Sybase/C interface library (Wagner, 1990) on top of the one provided by Sybase to ease the task of accessing the database from programs and to insulate us from various peculiarities. The most common way for our users to enter experimental data is to use Unix shell command script programs. These programs are easy to write, require no compilation and are especially suited for frequent modifications. Using minimal key strokes, the users can enter their data in a logical and efficient manner, with defaults or repetitive data automatically supplied. All custom data entry programs we have built offer both input error checking as well as the ability to hide all details of table linkage from the users. Most of our data entry programs are complicated, generally involving over 10 related data tables and more than 2000 lines of script code. We tried using the Sybase 4GL tool to create screen forms for data input, but our end users objected to the excessive amount of mouse movement that was required and the inability to feed it bulk data.

Physical Mapping Database Design

Other genome labs have taken the approach that genomic cartography should be seen as an extension of the lab notebook database itself. This seems to require that the notion of a physical map be concretely defined in terms of the local experimental data. We decided that rather than trying to build a perfect "object-oriented" definition of a physical map, intimately tied to the biology involved, we would instead opt for a "relation-oriented" view of all data that could potentially be represented in a physical map.

Our approach to the storage of map integration data has been to automate the extraction of all salient experimental data into a small set of basic relations (orientation, distance, length, overlap, etc.) on a single abstract class of generic "map objects". We note that this approach frees us from any pre-conceived limitations in the database of what a physical map should look like. It has also allowed us during this year to incorporate data from several new types of objects (BAC/PAC/P1) without any perturbation to the map integration code. We use a separate database for these generic map objects and their relations, since our lab notebook lacks a unique, stable identifier for all objects. Periodically we rebuild the map object database from the current state of the lab notebook tables. We will comment later on how we plan to improve this situation in the near future.

An unexpected benefit of this generic mapping approach was our realization that this method could also be of value in producing consensus maps from a variety of independent sources. All that would be needed is the use of the same name for identical objects across all input maps. All objects could be converted into generic map objects, and all salient relationships from the input maps could then be translated into the basic set of relationships on these pooled generic objects. Our automated map integration algorithm described below would then be run on this merged set of relations and a consensus of ordering would emerge (and likely conflicting data as well.)

Why we rejected Object-Oriented Databases

Our goal for data retrieval is to help our users manipulate their data easily and give them maximum freedom to access stored data in any way that makes sense to them. We note that the lack of an ad-hoc query language in current object-oriented databases makes them completely unacceptable to us at this time. We also feel that these are not really databases in a strict sense, but merely persistent storage mechanisms for C++ or SmallTalk objects. As such, some features needed for large-scale, multi-user operation are still lacking or immature. Finally, we do not see that there is any particular advantage for end users of using object-oriented databases in a domain where the underlying objects are subject to such constant flux, nor are claims of reusability especially compelling under such circumstances. We expect that our objections will be removed as object–oriented databases mature and as OO concepts are incorporated into relational databases.

Tools that our Database supports

– Cosmid Fingerprint Analysis and Overlap Calculation

Earlier we mentioned that LLNL has "fingerprinted" over 15,000 cosmid clones and analyzed them for overlap. Cosmid clone fingerprints are merely the lists of DNA fragment lengths that one obtains when the 40K base–pair clones are digested (cut) with one or more enzymes, each of which cuts the DNA anywhere a certain short (typically 4–6 base) sequence is found. These enzymes have been carefully chosen to generate, on average, at least 50 fragments in the range of 30–460 base–pairs for each cosmid clone. Data is generated on a gel electrophoresis system that typically runs 32 or more data samples in parallel during a 10–hour run. Each peak indicates the presence of a DNA fragment of a specific length in base–pairs. The order of these component fragments is completely unknown (Carrano, 1989).

Extracting fragment lengths (i.e., peaks) from gel electrophoresis signals turns out to be non–trivial due to the combination of a number of factors: edge effects, non–linearity in the electric field strength, irregular heating of the gel, gel inconsistencies, tumbling DNA fragments overlapping lanes, non–linear DNA mobility and high variation in the underlying DNA chemistry. Another complication is the fact that about 10% of the time peaks either vanish or appear, even when repeatedly processing the same DNA under stringent conditions. Finally, up to four different fluorescent dye colors are tagged to different DNA samples in the same lane to increase throughput (one is a size standard, to help overcome some of the variations mentioned earlier.) These dyes have a large amount of spectral overlap and great effort must be taken to suppress the spurious peaks in one color that arise only from the presence of a large peak in another color.

Analyzing restriction fingerprints presents several difficulties not normally encountered in signals generated, for example, by DNA sequencing reactions. The signals acquired are highly non–stationary, consisting of features of varying sizes and shapes superimposed on a noisy, slowly varying background. In addition, the features in the data are not at all well–separated, but rather consist of a random number of overlapping peaks of varying shapes. For these reasons most of the model–based approaches to peak fitting that we tried, such as numerical deconvolution, failed to give consistent, reproducible results. We abandoned our efforts to find models that fit the idiosyncratic nature of the peaks in our signals, and instead turned to modeling the noise. We could then simply clean the noise from the signal and declare anything left over to be "real." We fit a first-order autoregressive model provided by Splus (Statistical Sciences, Inc., 1990) to our data and used it as a description of the noise. We then fed this model to a smoother-cleaner, which identified all those data points which were indistinguishable from noise at a certain error level. After subtracting those points from the signal, we were left with a clean signal on which we could apply simple bump-hunting methods to determine peak locations. Input to this process is from our "raw" database; all peak location output is written directly to the ch19 production database.

We construct contigs based on data describing which pairs of cosmids are likely to overlap. That is, for each pair of cosmid clones, we use a statistical method to compute a number which represents how likely it is that the cosmids overlap. The larger the number, the more likely the overlap. This method, outlined in (Branscomb, 1990), uses data on how well two fingerprints match to compute the logarithm (base 10) of a "likelihood ratio". The ratio whose logarithm we compute is the ratio of two probabilities: the probability of seeing a given pattern of matching (and mismatching) bands in a pair of fingerprints when the two cosmids overlap, divided by the probability of seeing that pattern when the two cosmids do not overlap. The more the two fingerprints match, the more positive is the log of the ratio between these two probabilities. Conversely, the more two fingerprints do not match, the more negative is the log of the ratio between the two probabilities. It is this log likelihood ratio between any pair of clones that is the input to our subsequent reconstruction algorithm.

The probabilities that make up the likelihood ratio are rather complex to compute (Branscomb et al. give some details). The probabilities we compute are based on the actual distribution of fragment sizes in our data. This means that we can easily adapt our algorithm to regions of the genome where the probability of seeing fragments of certain sizes is unusual in some respect. For instance, in areas containing many similar genes, certain patterns of fragment sizes may be much more common than usual. In these areas we can adapt our algorithm to discount matches among fragments of those sizes, and concentrate instead on the matches that are more informative in nature.

When we implemented our overlap algorithm we quickly discovered that we would need access to a significant amount of computing power in order to be able to process all overlap calculations in a reasonable amount of time. Given 15,000 clones we must compute about 112,500,000 likelihood ratios, each requiring up to 100K floating point operations. We split our overlap job to run in parallel over all our workstations, using Unix socket inter-process communication calls (Comer, 1988) to establish a single "server" process which handed out work to do to the parallel "client" processes. Each pair-wise comparison is totally independent of all others, so the order of completion was immaterial. The server hands out rows of pair-wise comparisons to be done and writes results into the database. The mechanism is robust against the loss of any client process (Slezak, 1989).

At present we have access to over 40 workstations with a combined potential throughput of over 100 Mflops. We can completely re-calculate the ~112 million pair-wise overlaps from 15,000 cosmid clone fingerprints in about 2 days using this network.

- Cosmid Contig Assembly

Once the pair-wise overlaps have been calculated, we are still faced with the daunting problem of reassembly of the entire original chromosome. We noted that the population distribution curve of the overlap likelihood function gave us a clue to a simpler approach (Branscomb, 1990). The great body of potential overlaps clearly show absolutely no chance of true overlap. A small number of overlap scores indicate dead-certain overlap, and there remains only a small, steep region where there is any uncertainty. This suggested that we try a fully-automatic simple greedy approach, ordering the pair-wise overlap likelihood values and beginning assembly with the most confident overlaps first, continuing down to some threshold where we lost confidence in the veracity of the overlap. Such an approach is embodied in our *humpty* re-assembly program.

Input to *humpty* consists of triplets (cosmid1, cosmid2, overlap_score) extracted from the ch19 database sorted in descending order on overlap_score. Run time program options allow the user to specify which overlap score is the lowest one we are completely confident of (**sure_match**), which score is the lowest one we will examine for reconstruction (**sure_miss**), and which score is the lowest one we will consider potentially significant in special circumstances (**min_sig**). Of course, **sure_match >= sure_miss >= min_sig.**

On real data subject to the errors mentioned earlier, false joins occur which are later detected via other methods. Given our 112 million overlaps and an assembly cutoff of 6 logs (real data requires a higher cutoff than errorless simulated data), we expect over 100 errors from the statistics alone. In most cases, a single false positive overlap greater than our threshold incorrectly joins 2 otherwise internally-valid contigs. In the remaining cases, tandem repeat gene families or similar repetitive regions cause scrambling that was cleared up either by Eco restriction mapping or by re-running the overlap calculation tuned to the peculiar peak frequencies of those regions. Methods described below detect and assist in repairing errors in cosmid contig assembly.

– Partial Order Tree Generation

We derive order data from genetic, FISH, and restriction mapping methods. Several resolutions of FISH data provide us with distance estimates as well (Brandriff, 1994). We chose to store all order data in terms of pairs (i.e., clone A is P-terminal of clone B) and automatically derive the backbone partial-order tree that will be used as the core of our map integration.

Our partial ordering algorithm uses pair-wise order data extracted from the ch19 database. It constructs adjacency and maximal path length matrices and uses these to generate compacted partial order trees. The algorithm also detects and reports cycles, so that errors or inconsistencies in the underlying data can be examined and corrected. Output from this algorithm is written directly to the database and can be viewed in our browser (Figure 3). It has been used to help resolve the ordering of a large set of markers on chromosome 19.

– Automated Physical Map Integration

The integration of the cosmid contig maps with other mapping information covering the same domains is a complex but important problem. The LLNL chromosome 19 physical mapping effort involves more distinct experimental sources of data than most other mapping projects. We have cosmid contig data, four types of FISH data, Eco restriction maps (Figure 4), YAC/STS data, and many types of hybridization data (cosmid->cosmid, plus bi-directional cosmid<->YAC, cosmid<->BAC, and cosmid<->PAC/P1). In addition, we have order data derived from genetic mapping markers which are linked to cosmid clones via unique probes. This wealth of diverse data has

By ordering the sets of triplets on descending overlap_score, we ensure that the algorithm makes the most confident reconstructions first. This allows us to rely on contextual evidence to support or deny any purported further reconstructions at lower levels of confidence. A pseudo-code form of the algorithm follows:

```
loop:     set C1, C2 from the triplet on top of the list. stop if overlap_score < sure_miss
        if C1 and C2 are both not in contigs, make a new contig.
        else if C1 and C2 are already in the same contig, do nothing.
        else if only one of C1, C2 is already in some contig,
                    if overlap > sure_match,
                            add the new cosmid to that contig.
                            update minimal spanning path
                    else check all contigs for best fit based on total system "energy"
                     i) add new cosmid to that existing contig
        else C1 and C2 are already in different contigs
                    choose best options based on total system "energy"
                     i) do nothing
                     ii) attempt to merge contigs
                     iii) remove one cosmid and place in other contig
        remove top triple from the list,
          if the list is not empty, go to loop
```

The overlap likelihood has been shown to have the additive property of an information statistic (Branscomb, 1990) and hence we can compute the total "energy" of the solution at any time to make informed decisions about placing clones in contigs or moving them to different contigs.

The heart of the *humpty* algorithm is the logic that controls the merging of two existing contigs and the dynamic determination of a minimal spanning path. Preventing false joins is of the utmost importance in contig assembly, so stringent sanity tests must be passed before a purported merge is allowed. (The cosmid clone in the purported merge is projected onto its appropriate minimal spanning path position if not already present. If it does not project to a path end, all path members between it and one end must confirm overlap to the other contig.) Similarly, presenting a reliable near-minimal spanning path for each contig has at least two impacts: it limits how many clones need to be stored to cover the DNA contained in each contig, and it guides high-resolution Eco restriction mapping. Spanning paths are easily computed on the fly as the descending list of overlap probabilities brings new clones into contigs, one at a time. The new clone's overlap with all existing spanning path members is checked and the new clone either 1) extends one end of the path, 2) replaces one or more existing contiguous path members, or 3) has no effect on the path. Similar logic handles contig merges. All contig membership data is written directly to the ch19 database.

It took one workstation 15 minutes to reassemble 8,000 simulated errorless chromosome 19 cosmid clones into 482 spanned contigs. Examination of the resulting contig gaps shows that they either are due to the lack of suitable cutting sites in that region or the inability to reliably detect overlap at that location (ie, overlap $< \sim 30\%$). There were 2 false joins at a log-likelihood cutoff level of **min_sig** = 4 Logs, out of 3.2×10^7 total potential overlaps. No other clones were placed incorrectly into contigs. A recursive routine was written to exhaustively check the accuracy of our spanning paths by generating all possible valid spanning paths to find the shortest. Using several workstations we were able to verify 476 of the 482 spanning paths over the course of a week. The largest contig we were able to confirm contained 62 total members, with 19 on the spanning path. It took 1.2 *billion* recursive calls (and several days) to verify that there was no shorter spanning path for this single contig. Only 83 spanning paths (17.4%) were not true minimal paths, 72 of these being 1 member too long and 11 being 2 members too conservative. Another way of stating this is that we required 2411 cosmids to span the DNA in 476 contigs, instead of the true minimum of 2317. All near-minimal spanning paths were true spanning paths (i.e., covered all the DNA represented in the contig), with the exception of 8 path ends that were called incorrectly. In each case these were nearly complete overlaps with the "true" end element (i.e., the chosen end was a subset of the "true" end), yet based on overlap data alone the algorithm had made the proper choice. Based on these tests we conclude that our dynamic spanning path assembly algorithm provides both high speed and accuracy that reflects the quality of the input data, giving us a reliable base for completing the physical map via other methods.

challenged us to develop automated methods to integrate the over 500,000 relations that we track among 180,000 potentially-mappable objects.

Our primary design goal for integrating physical map data is to use fully automated map assembly techniques as far as proves feasible. We find that it is useful to distinguish between "maps for map builders" and "maps for map readers." While the later is the end-goal, our tremendous diversity of data dictated that we first build an intermediate map which integrates all data democratically and thereby makes visually obvious all the "interesting areas" of the map. This includes errors of many sorts (using the wrong clone, data entry error, database error, analysis software error, etc.) and also unusual biology ("unique" probes that are not unique, repeat elements, deletions, chimerism, etc.)

Having stored all integrated mapping data as a set of relations on a single class of generic map objects, we noticed that if we ignored distance and length data we could greatly simplify the problem of map integration. Our data can then be considered to be "generic hybridization" data (including all true hybridizations plus contig membership and restriction map membership since they similarly imply overlap of DNA). We decided to ignore distance and length data in our initial integrated mapping approach and concentrate solely on the order of cosmid clone probes. We refer to this first-level integrated physical map as our "partial order" map, as distinguished from the "metric" map that is following (Wagner, 1994). Having made the X-axis of our integrated map unitless, we now saw that map integration could be reduced to an optimization problem where "probes" (e.g., cosmid clones, some of which might be associated with STS markers or loci) intersected larger "objects" (e.g., cosmid clone contigs, restriction fragment maps, YACs, BACs, PACs).

We solved this optimization problem via simulated annealing running on 40+ machines in parallel (Slezak, 1994). Current maps of 2,900 cosmid clone probes intersecting 750 larger objects (Figures 5 and 6) take about 5 hours to construct on this network and are written to the integrated map database. These maps are being used to efficiently guide further work towards map closure.

Browsing the Map

The implementation of our database browser (an X/Motif graphical tool for viewing and navigating our database) has been an ongoing effort for over five years, with 40 researchers using it daily. No prior work existed when we started and there was no consensus locally about what it should look like.

The browser was therefore designed to be a flexible framework, capable of adding new data types and display methods as they were requested. We currently have 2 modes for examining cosmid contigs, a viewer for raw cosmid fingerprint data, a mode for examining orphan cosmid clones, modes for examining restriction fragment maps and sequencing data, a general multi-database query mode, and 3 modes for viewing various integrated maps. Each mode in our browser can be thought of as an independent "display object." These objects are self-contained, possessing both the data and the appropriate display functions needed to draw them in an independent window. Since many of the map modes are closely interrelated (i.e., a cosmid clone in a contig display may also appear in a restriction map display and also in several integrated map displays) we have provided a feature that always tracks the currently selected object of focus, reporting in the global selection menu which other modes have data pertinent to that focused object. This allows a user to track through a wide range of maps for the right level of detail.

One of the most important features of the browser is our query menu interface (Figure 7). We needed to satisfy a range of users, from total computer novices to experts, with an interface that would neither be intimidating nor frustrating. We also wanted to avoid limiting the types of queries that could be made. We accomplished this with a generic user interface with which users can select from a list of "canned" queries which do the most common tasks. They can also type in direct SQL through an editing feature or use a canned query as a template, and then edit that query to suit their particular needs. Canned query templates are stored in the database, as are all other data that customize the query menu for a particular usage. The query menu code knows nothing about biology or SQL and therefore could be used as a general database interface. It only knows several methods of allowing the user to construct and expand queries, which are then passed to the SQL server. Return values are assumed to be database keys of a type that is valid for the current display mode.

Besides canned queries that take user-supplied values at run time, our interface also allows user queries to be read from and stored to files. Not all desired functions can be done with SQL alone (e.g., recursive queries) so we had to allow for arbitrary external processes (or pipelines of processes) to be run and their output to be a stream of database keys that could feed directly as a "hit list" to the query menu interface. An unexpected benefit of this interface is that most of our biologist users have learned enough SQL from the canned queries that they can satisfy most query needs on their own.

Summary and Future Directions

We would like to stress that our system should be viewed not as a particularly elegant or optimal solution, but rather as a case study of the types of problems faced by those designing systems to meet the needs of researchers in a rapidly–changing field. In such environments we feel that the ability to cope with the tradeoffs and compromises required to quickly produce working tools is more important than rigid adherence to any particular set of formal design rules or tools. End–users were involved heavily in design and for feedback during rapid prototyping and were not "protected" from database realities.

In hindsight, many of the problems our database system now has are a direct result of the barnacle–like growth of the project as the discipline of physical mapping was born. Viewpoints changed, some tables atrophied, and techniques mutated and overlapped over time. Our use of a separate database for generic physical map objects is a prime example of such a barnacle. When we began the lab notebook database we had no idea what our physical mapping needs would be. Four years later when we designed the generic map object database we weren't sure that it would suffice, so it was done separately. For the last 2 years as we raced towards map closure we have not had the time or resources to stop and "do things right". Now that our techniques are beginning to stabilize we will take the opportunity to learn from our experience.

The LLNL Human Genome Center is preparing to do mapping on other human, animal, plant, and microbial chromosomes. Much of this data will be cross–referenced (i.e., homologous genes found on many species). This will require a complete re–design of our database, which was consciously designed to only handle human chromosome 19. Some required changes are:

- Stable, unique identifiers for all objects in the database. This will allow us to not have to keep separate lab notebook and generic mappable object databases. Aliases can supply familiar names.

- Higher level of data abstraction. We need only one clone table, capable of holding YACs, BACs, PACs, P1s, cosmids, and any other cloning vehicle used for any genome we study. This table will act as a "base class", with other tables as necessary holding specific details unique to each clone type.

This will greatly reduce the number of database tables and query complexity, and views can provide the illusion of single chromosome or species databases.

– Higher level of relationship abstraction. Our current database had separate tables for array, pool, and southern hybridization to track the experimental techniques closely. Just as a single clone table can handle all generic "clones", we need a single generic hybridization table (to handle all hybridizations from all types of clones.)

– Generic concepts of clonal overlap. A single table can handle all pertinent information from a multitude of hybridization techniques, again as a base class with detail tables as needed.

– Generic concept of probes. Our current lab notebook database now tracks about 20 different types of probes, with too many important details (like the probe name) buried in the 20 detail tables. This makes the SQL to extract such data quite messy. Again, we need a single "base class" probe table that contains all data generic to all probes, with lab notebook details confined to tables that are not vital to the central task of map creation.

– Generic concepts of attributes and ownership. The presence of stable, unique identifiers will allow us to have a single table to tie attributes to objects. It will also allow us to link each data entry to an owner.

– Generic concept of privacy. Some data is not publicly releasable for a variety of reasons. We need to be able to control this information on a per–fact basis so that safe public access can be offered without violating any of our 200+ collaborative agreements.

Work on this new database has recently commenced.

Acknowledgment

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract No. W–7405–ENG–48.

References

Ashworth, L., (1994). Data Dictionary for the LLNL Human Genome Center chromosome 19 database. Available via anonymous ftp at humpty.llnl.gov in /pub/datadictionary.ps.Z

Brandriff, B., *et. al.*, (1994). Human chromosome 19p: A Fluorescence in situ hybridization map with genomic distance estimates for 79 intervals spanning 20Mb, *Genomics* (In Press).

Branscomb, E., *et al.* (1990) Optimizing Restriction Fragment Fingerprinting Methods for Ordering Large Genomic Libraries. *Genomics* **8**, 351–366.

Carrano, A.V., Lamerdin, J., *et al.* (1989). A high–resolution fluorescence–based semi–automated method for DNA fingerprinting. *Genomics* **4**, 129–136.

Comer, D. (1988) Internetworking with TCP/IP: Principles, Protocols, and Architecture. Prentice Hall.

Slezak, T. (1989) Quick and Dirty Parallel Processing on a Network of Workstations, Tentacle (LLNL Internal publication) IX:6–7 16–28. Copy available via ftp from humpty.llnl.gov in /pub/tentacle

Slezak, T. (1994). Automated Integration of Genomic Physical Mapping Data via Parallel Simulated Annealing, in Proceedings of 3rd International Conference on Bioinformatics and Genome Research (in press). Available via ftp from humpty.llnl.gov in /pub/supercomp94.text.ps.Z

Statistical Sciences, Inc (1990). Splus Users Manual. Seattle, Washington.

Wagner, M., (1990). Sybase C interface library. Available via ftp from humpty.llnl.gov in /pub/genomestuff/mdi.tar.Z

Wagner, M., (1994). Automated Construction of Physical Maps. Master's Thesis, University of California, Davis.

Weber, C.A., Salazar, E.P., *et al.* (1988). Molecular cloning and biological characterization of a human gene, ERCC2, that corrects the nucleotide excision repair defect in CHO UV5 cells. *Mol. Cell. Biol.* **8**, 1137–1146.

Yeh, M., (1994). Database schema diagrams for the LLNL Human Genome Center chromosome 19 database. Available via anonymous ftp at humpty.llnl.gov in /pub/databaseschema.ps.Z

Figure Legends

NOTE: Most of the figures are color screen dumps from our database browser and probably will not be visible here with full fidelity. Contact the author for access to copies via ftp or WWW.

Figure 1 – A view of one of our 800 cosmid clone contigs, as seen from our browser. Each number in the lower left side is the database key of a 40Kbp cosmid clone. The 9 clones at the bottom are the near–minimal spanning path of the contig; all other clones are redundant and stacked over their strongest overlap to the minimal path. A database query for a gene–specific probe got us into this contig; the 4 clones positive for that probe are enclosed in boxes. We are currently focused on clone 16922 and all attributes for this clone are shown in the scroll–able window at top right. Buttons on the bottom right allow over 20 specialized functions to be performed. The rectangular window labeled "Browser" is the main menu. The 12 viewing modes in the top half are available for independent launching. The bottom half of the window indicates what other viewing modes contain data pertinent to the object currently in focus (cosmid clone 16922 in this instance.)

Figure 2 – An overall view of the major databases in our system. The raw database stores over 1.5Gbytes of restriction fingerprinting data. The ch19 database stores all derived cosmid contig data plus all other experimental data used in mapping. The integrated mapping database expresses all data in terms of basic relationships on a single class of generic mappable objects, allowing for fully automated integration of data from many sources.

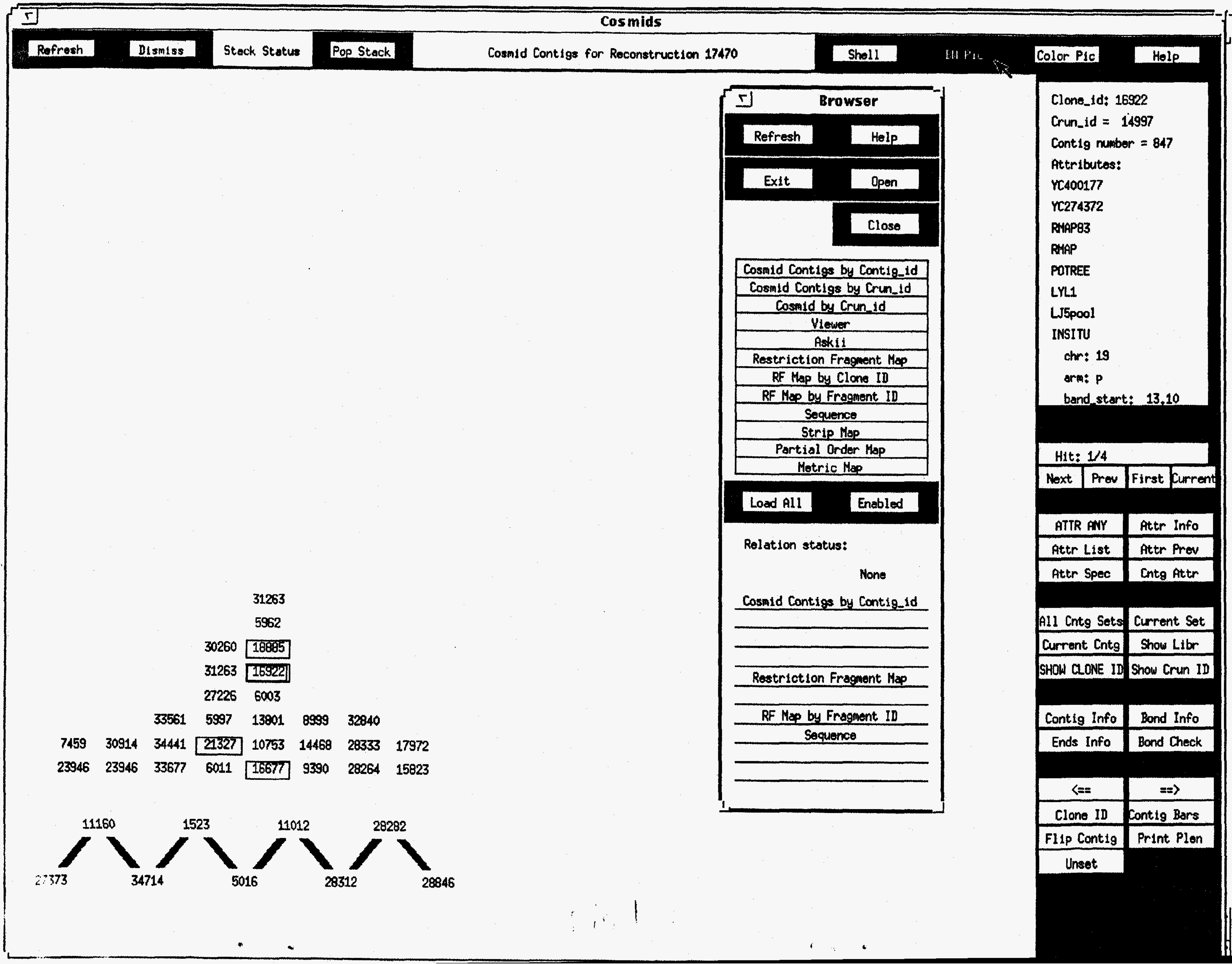
Figure 3 – This display presents one way of examining the relationships between a large variety of data objects. Near the top, a diagram shows the chromosome and the band(s) associated with the current object of focus. Below the diagram are controls for scrolling and manipulating the map object stack, as well as controls for the query hit list (if any). Below that are the indicators to control the display of up to 4 of the available map object types at any time (17 types are shown in this image, which does not show the use of color to indicate object selection.) We have chosen to display YACs, contigs, clones, and loci. An information window at top right shows that object 44049, which is clone id 16922, has been selected for focus (the red color used is lost in this rendering.) This clone is in a partial–order tree in the cosmid clone display, and has links to 2 YACs, a contig, and 2 loci. We can see from the object selection boxes that there is data for this clone in linked to 6 other map object types that we could choose to show, by turning off one or more of the current views.

Figure 4 – This is a portion of one of our many EcoRI restriction fragment maps. Each horizontal row represents one 40Kbp cosmid clone, with the clone name on the left and contig number on the right. The fragments are drawn to scale and alignment is easily checked visually. With a click of a button, the display changes to show the numeric sizes of the fragments (which range from 0.5Kbp to about 20Kbp.) Although not shown, these fragments can be linked to sequence–related attributes such as genes, loci, etc.

Figure 5 – A full–chromosome view of our integrated map, containing 2,941 probes (columns) intersecting 760 larger objects (contigs, restriction maps, YAC/BAC/PAC clones on the rows), with clones partially ordered via FISH or genetic mapping shown in green. Note that off–diagonal elements indicate data that needs to be examined closely. The region enclosed in the red box is enlarged in Figure 6.

Figure 6 – A tiny region of our chromosome 19 integrated map has been shown in detail, to demonstrate how our map builders are using this tool to guide closure. The long object composed of black crosses is a PAC clone; a BAC clone (black diamonds) is located in 3 pieces 4 rows above. Note that the BAC and PAC clones are deduced to overlap each other, since 11 cosmid clones have been found to hybridize to both of them. Furthermore, the BAC and PAC clones show evidence of overlapping 4 cosmid clone contigs (in blue). Also note that this region is anchored on one side by a clone (shown in green) which has been FISH ordered. The four contigs have been restriction mapped (blue crosses) based on the hybridization evidence and were found to overlap using this higher–resolution mapping method.

Figure 7 – Browser query menu interface. This is our generic database interface. The information lines near the top describe which mode of the database browser we are in and what type of database keys must be returned by all queries. A scroll–able list of canned queries is on the left, with a mouse–click the user can get a description of the query or pull up the SQL template in the workspace on the right. The attribute locator query has been selected. Note the embedded marker ("@I1@") in the SQL indicating that the user must interactively supply the value of the particular attribute desired, by clicking the "Expand Next" button. Once the query is fully expanded, the red "Run Query" button is hit (invisible in this B/W picture, it is to the right of the "Expand Next" button.) Other buttons allow an editor window to appear, with the workspace contents pre–loaded, and for external queries to be run using either the workspace contents or specified external files. Queries can be saved or loaded from files. This interface is used in all browser modes except the raw data viewer.



Cosmid Contigs for Reconstruction 17470

Clone_id: 16922
Crun_id = 14997
Contig number = 847
Attributes:
YC400177
YC274372
RMAP83
RMAP
POTREE
LYL1
LJ5pool
INSITU
chr: 19
arm: p
band_start: 13.10

Hit: 1/4

Relation status:

None

Cosmid Contigs by Contig_id

Restriction Fragment Map

RF Map by Fragment ID

Sequence

```
                                  31263
                                  5962
                        30260   [18885]
                        31263   [16922]
                        27226     6003
                33561   5997     13801    8999    32840
  7459  30914   34441  [21327]   10753   14468    28333   17972
 23946  23946   33677   6011    [16677]   9390    28264   15823

        11160        1523        11012        28282
 27373        34714        5016        28312        28846
```

Overall System Data Flow

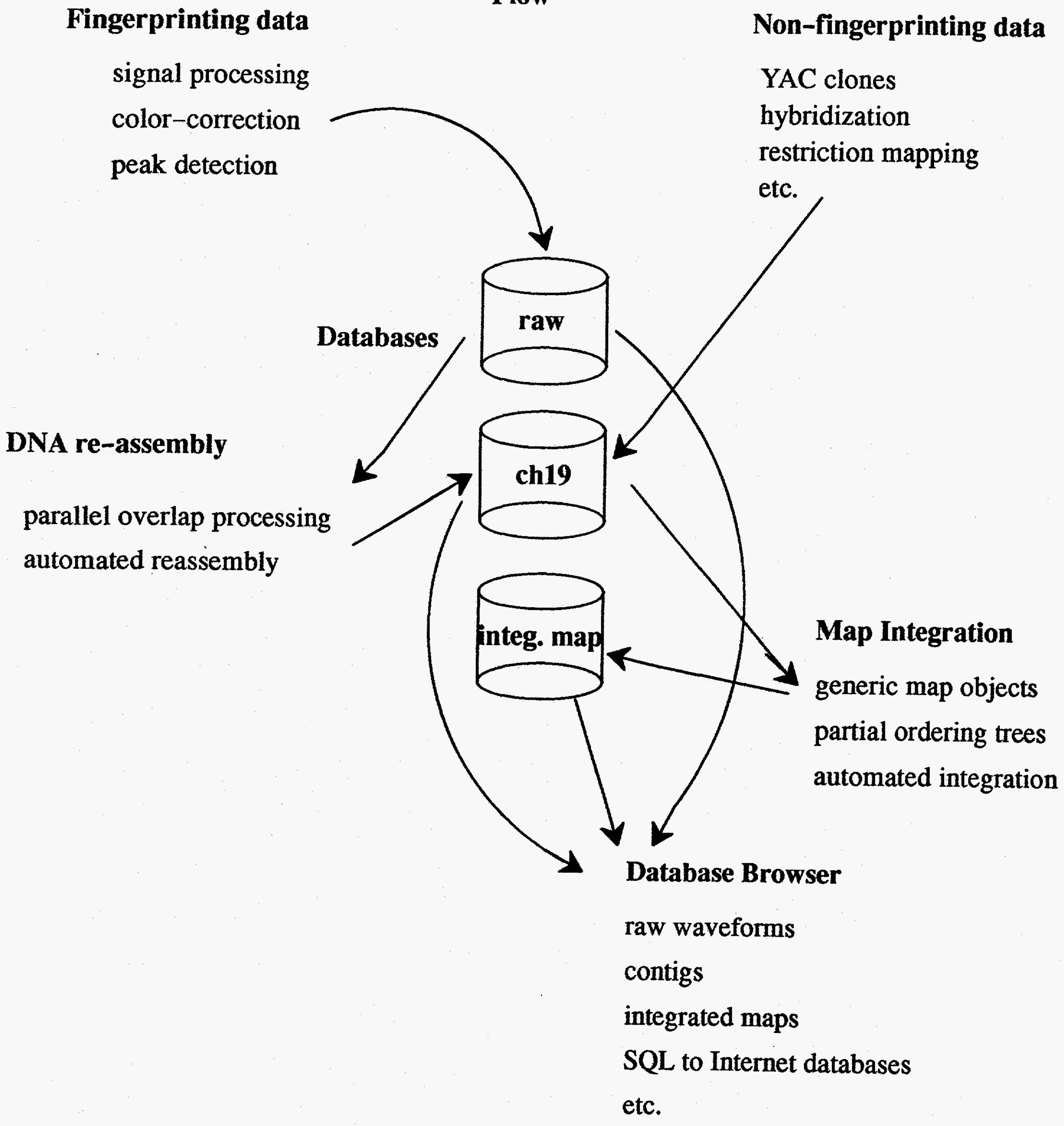
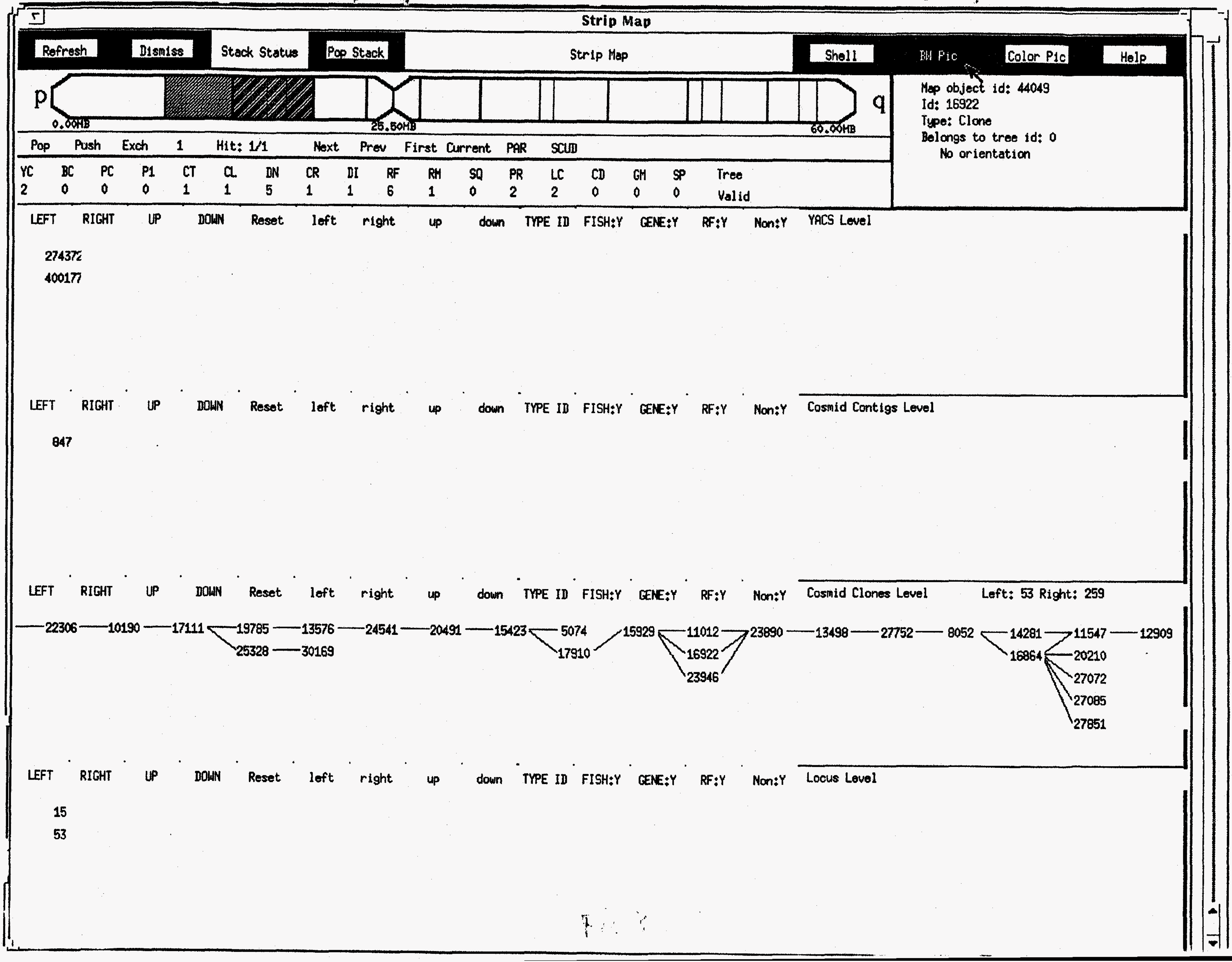


Fig. 2



```
Map object id: 44049
Id: 16922
Type: Clone
Belongs to tree id: 0
   No orientation
```

```
Pop   Push   Exch   1   Hit: 1/1   Next   Prev   First  Current   PAR   SCUD
```

YC	BC	PC	P1	CT	CL	DN	CR	DI	RF	RM	SQ	PR	LC	CD	GM	SP	Tree
2	0	0	0	1	1	5	1	1	6	1	0	2	2	0	0	0	Valid

```
LEFT   RIGHT   UP   DOWN   Reset   left   right   up   down   TYPE ID   FISH:Y   GENE:Y   RF:Y   Non:Y   YACS Level

  274372
  400177
```

```
LEFT   RIGHT   UP   DOWN   Reset   left   right   up   down   TYPE ID   FISH:Y   GENE:Y   RF:Y   Non:Y   Cosmid Contigs Level

  847
```

```
LEFT   RIGHT   UP   DOWN   Reset   left   right   up   down   TYPE ID   FISH:Y   GENE:Y   RF:Y   Non:Y   Cosmid Clones Level     Left: 53 Right: 259

—22306—10190—17111—19785—13576—24541—20491—15423—5074  15929—11012—23890—13498—27752—8052—14281—11547—12909
                   \25328—30169               \17910/      16922/                   \16864—20210
                                                           23946                           27072
                                                                                           27085
                                                                                           27851
```

```
LEFT   RIGHT   UP   DOWN   Reset   left   right   up   down   TYPE ID   FISH:Y   GENE:Y   RF:Y   Non:Y   Locus Level

  15
  53
```

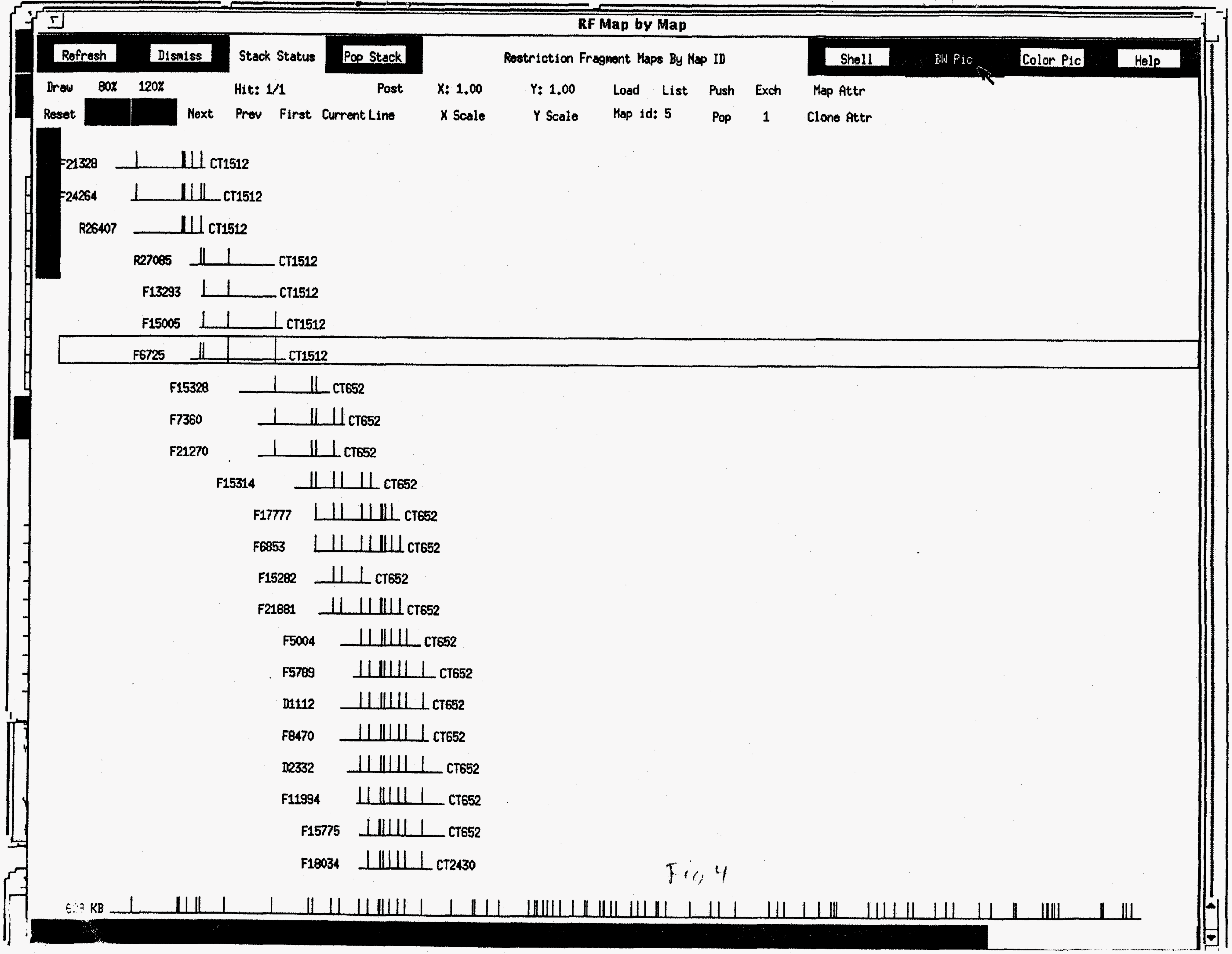
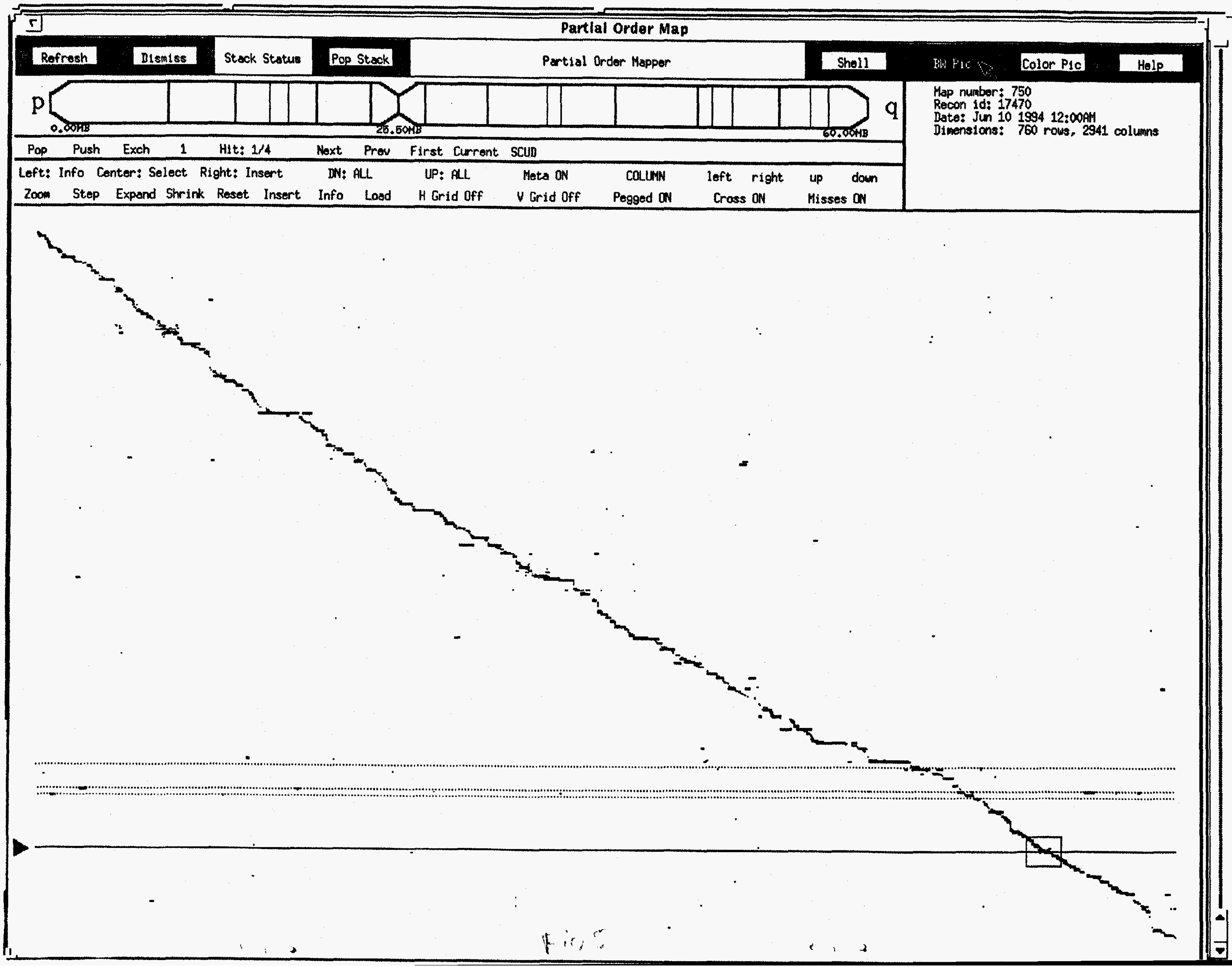
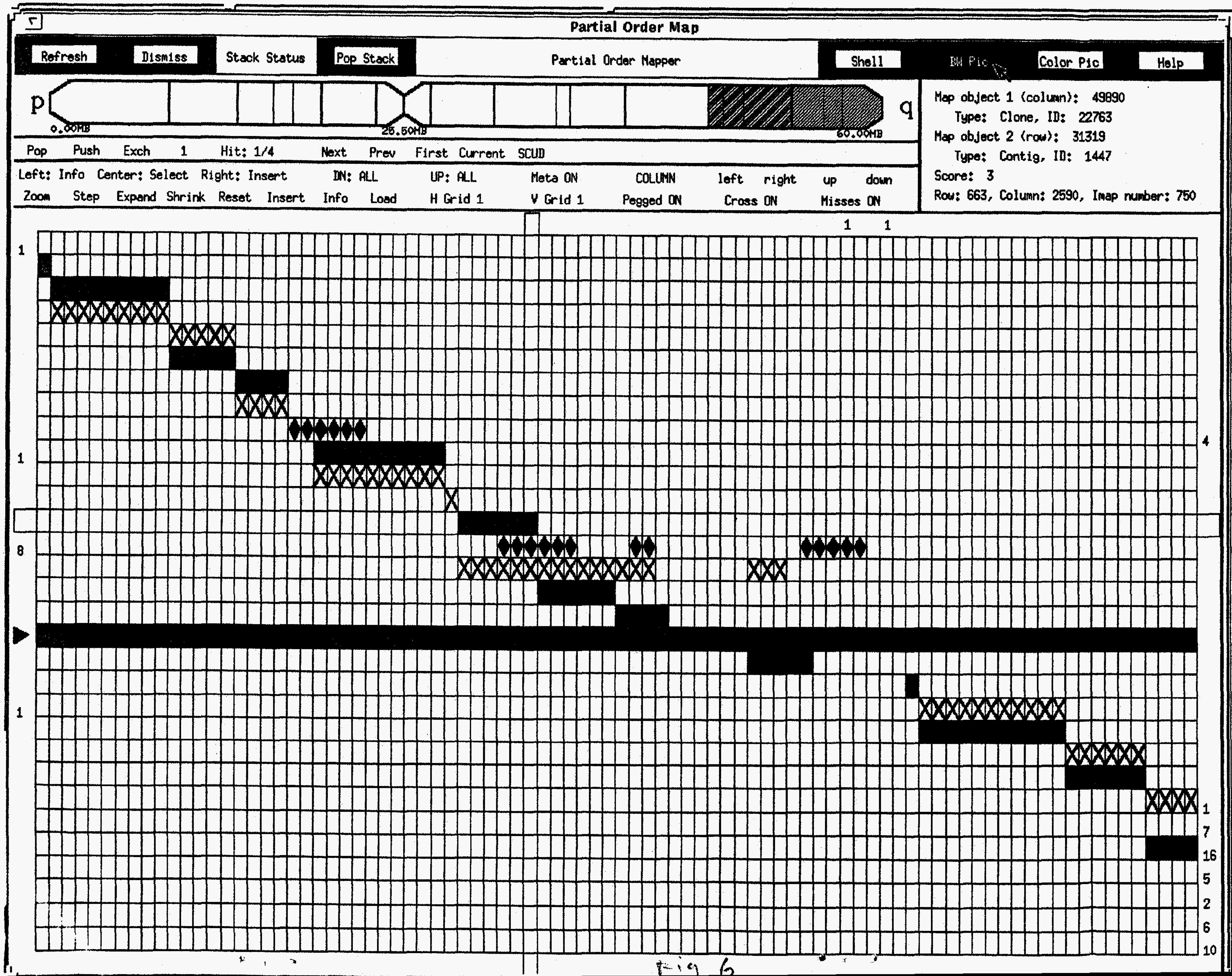

RF Map by Map
Refresh
Dismiss
Stack Status
Pop Stack
Restriction Fragment Maps By Map ID
Shell
BW Pic
Color Pic
Help
Draw 80% 120%
Hit: 1/1
Post
X: 1.00
Y: 1.00
Load
List
Push
Exch
Map Attr
Reset
Next
Prev
First
Current Line
X Scale
Y Scale
Map id: 5
Pop
1
Clone Attr
F21328 CT1512
F24264 CT1512
R26407 CT1512
R27085 CT1512
F13293 CT1512
F15005 CT1512
F6725 CT1512
F15328 CT652
F7360 CT652
F21270 CT652
F15314 CT652
F17777 CT652
F6853 CT652
F15282 CT652
F21881 CT652
F5004 CT652
F5789 CT652
D1112 CT652
F8470 CT652
D2332 CT652
F11994 CT652
F15775 CT652
F18034 CT2430
KB

Fig 4


Partial Order Map
Refresh
Dismiss
Stack Status
Pop Stack
Partial Order Mapper
Shell
BW Pic
Color Pic
Help
p
q
0.00MB
25.50MB
60.00MB
Pop Push Exch 1 Hit: 1/4 Next Prev First Current SCUD
Left: Info Center: Select Right: Insert DN: ALL UP: ALL Meta ON COLUMN left right up down
Zoom Step Expand Shrink Reset Insert Info Load H Grid Off V Grid Off Pegged ON Cross ON Misses ON

```
Map number: 750
Recon id: 17470
Date: Jun 10 1994 12:00AM
Dimensions:  760 rows, 2941 columns
```

Fig 5

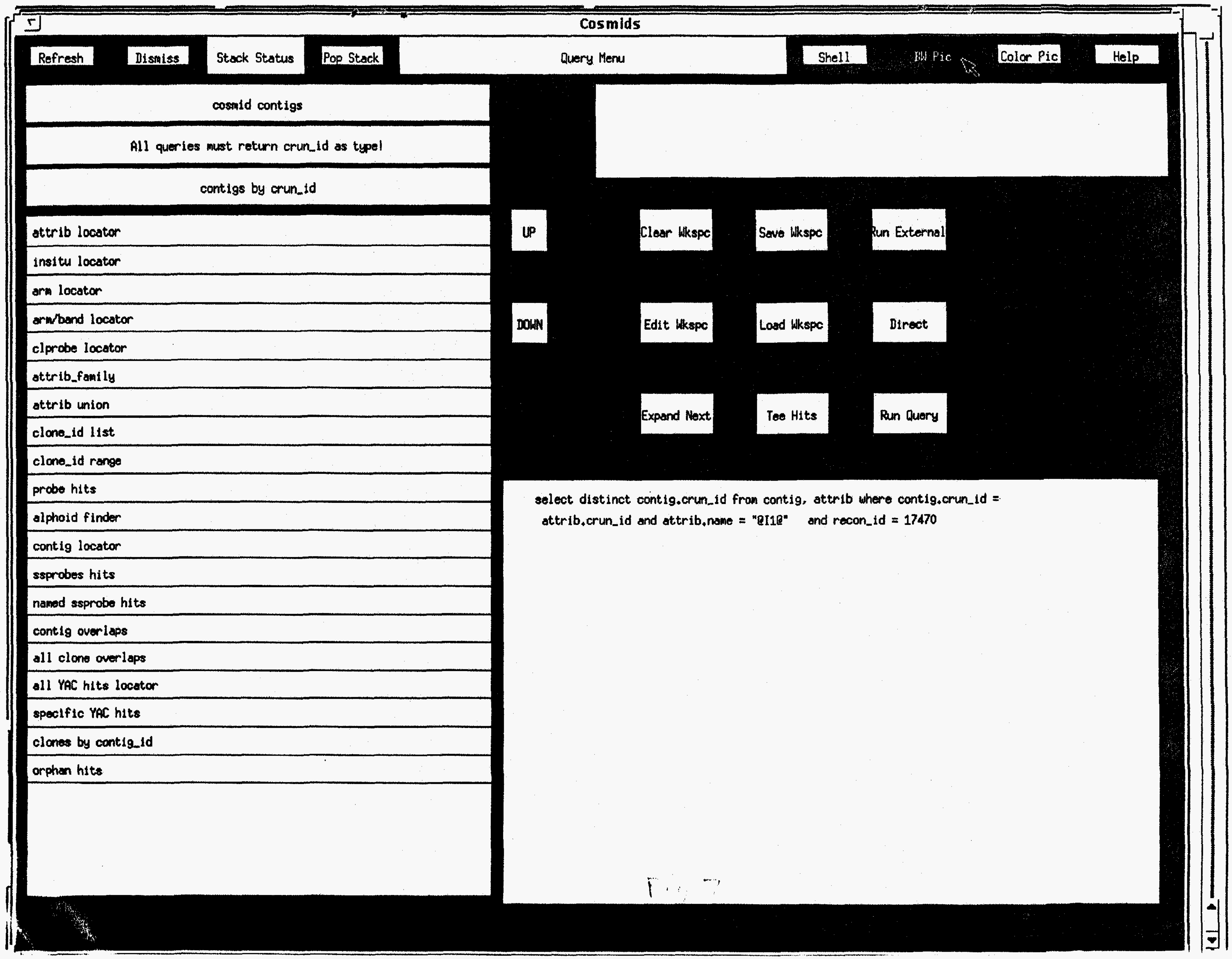


Partial Order Map

Map object 1 (column): 49890
Type: Clone, ID: 22763
Map object 2 (row): 31319
Type: Contig, ID: 1447
Score: 3
Row: 663, Column: 2590, Imap number: 750

Refresh
Dismiss
Stack Status
Pop Stack
Partial Order Mapper
Shell
BW Pic
Color Pic
Help
p
q
0.00MB
26.50MB
60.00MB
Pop Push Exch 1 Hit: 1/4 Next Prev First Current SCUD
Left: Info Center: Select Right: Insert DN: ALL UP: ALL Meta ON COLUMN left right up down
Zoom Step Expand Shrink Reset Insert Info Load H Grid 1 V Grid 1 Pegged ON Cross ON Misses ON
1
1
1
8
1
1
1
4
1
7
16
5
2
6
10

Fig 6



```
select distinct contig.crun_id from contig, attrib where contig.crun_id =
 attrib.crun_id and attrib.name = "@I1@"   and recon_id = 17470
```

Fig 7