Physical Mapping by STS Hybridization: Algorithmic Strategies and the Challenge of Software Evaluation

David S. Greenberg
Sandia National Laboratories
Mail Stop 1110
P.O. Box 5800
Albuquerque, NM 87185-5800
dsgreen@cs.sandia.gov

Sorin Istrail
Sandia National Laboratories
Mail Stop 1110
P.O. Box 5800
Albuquerque, NM 87185-5800
scistra@cs.sandia.gov*

March 22, 1995

To appear in the Journal of Computational Biology, summer 1995

Abstract

An important tool in the analysis of genomic sequences is the physical map. In this paper we examine the construction of physical maps from hybridization data between STS (sequence tag sites) probes and clones of genomic fragments. An algorithmic theory of the mapping process, a proposed performance evaluation procedure, and several new algorithmic strategies for mapping are given. A unifying theme for these developments is the idea of a "conservative extension." An algorithm, measure of algorithm quality, or description of physical map is a conservative extension if it is a generalization for data with errors of a corresponding concept in the error-free case.

In our algorithmic theory we show that the nature of hybridization experiments imposes inherent limitations on the mapping information recorded in the experimental data. We prove that only certain types of mapping information can be reliably calculated by any algorithm. A test generator is then presented along with quantitative measures for determining how much of the possible information is being computed by a given algorithm. Weaknesses and strengths of these measures are discussed.

Each of the new algorithms presented in this paper is based on combinatorial optimizations. Despite the fact that all the optimizations are NP-complete, we have developed algorithmic tools for the design of competitive approximation algorithms. We apply our performance evaluation program to our algorithms and obtain solid evidence that the algorithms are capable of retrieving high-level reliable mapping information.

*This work supported in part by the U.S. Department of Energy under contract DE-AC04-76DP00789.

1 Introduction

A central question for the Human Genome Program is how to bridge the gap between the size of DNA fragments which can be directly sequenced and the size of the human genome. Researchers continue to look for ways of extending the size of fragments which can be directly sequenced and ways of picking out important pieces to sequence. However, at present and in the near term, large scale sequencing projects depend on some process which involves dividing a large segment of DNA into overlapping pieces, analyzing the smaller pieces separately, and determining the order of the pieces so as to combine information about the pieces into information about the whole.
It is this last step, reordering fragments, which is the focus of this paper. Many methods have been proposed for reordering fragments – ranging from tools for helping experts reorder the fragments by hand to automatic programs employing maximum likelihood analysis, combinatorial optimizations, or a variety of heuristics. These tools have been aimed at data which includes STS probe-clone interactions, FISH data, radiation hybridization data, genetic map orderings, break point positions, etc.
Typically a given tool or algorithm is evaluated by applying it to actual data and displaying the resulting "map". In this paper, we attempt to provide a framework for a more rigorous analysis of mapping algorithms. We aim to answer the following questions:

1. For a given type of data, what type of information is reasonable to expect from an algorithm?

2. For a given algorithm, how well does it work on different types of data?

3. Are there good algorithms for particular types of data and if so what are they?

Since these questions are clearly broad and difficult to answer we have begun by concentrating on a particular type of data, hybridizations of sequence-tag-site (STS) probes with clones of genomic fragments, and a particular class of algorithms, those based on combinatorial optimizations. We pay especial attention to data in which there are errors of various types: clones which contain multiple unrelated fragments (i.e. chimera), clones with internal deletions, and both false-positive and false-negative errors in the hybridization data.
In this first study we were able to show:

1. The information available in a hybridization matrix of even error free experiments is limited. The amount of available information is directly related to the information in the PQ-tree of the matrix.[1]

2. The presence of errors further degrades the information available. Furthermore the errors may obscure redundancies and singularities which could easily have been filtered in the error-free case.

3. Despite these limitations on information available in hybridization matrices it is possible, even data containing errors, to determine relevant facts about the target genome.

[1] Some terms here such as PQ-tree may be unfamiliar to readers new to the subject. They are defined in later sections.

In particular, we identify "local properties" which can be determined, such as the fact that certain probe pairs are adjacent.

4. Given the correct ordering for a matrix it is possible to divide the adjacent probe pairs into a weak and a strong set where the weak set cannot be reliably determined by any algorithm and the strong set is potentially determinable. The definitions of these sets is formal and precise for error-free data and heuristic for data containing errors.

5. The percentage of strong adjacent pairs identified by an algorithm is a reasonable measure of algorithm success and can be used to compare algorithms. However, this measure is local in nature and more global measures are still needed.

6. We formulate a "noise model" which allows us to unify the seemingly dissimilar challenges of finding the correct order of probes and of dealing with the errors in the hybridization matrix. We show that combinatorial optimization functions can be used to search for solutions which require the postulation of as few errors as possible. We believe that the success of these functions is due to their being conservative extensions of the error-free case and their having monotonicity with respect to common error types.

Figure 1 gives a table of contents for this paper.
The remainder of the paper is divided up as follows. Section 2 presents the basic biology required for the rest of the paper. Section 3 rigorously defines the physical mapping problem and shows that the information about a genomic target which can be reliably inferred from an error-free hybridization matrix is related to the PQ-tree of the matrix. In Section 4 we discuss possible sources of errors in the hybridization matrix and extend our theory to include the errors. In Sections 5 and 6 we discuss how to generate synthetic data for testing mapping algorithms and show experimentally that hybridization matrices will tend to have ambiguities which algorithms cannot resolve. In Sections 7 and 8 we describe how combinatorial optimizations can be used to search for good maps and present several algorithms based on the optimizations. In Sections 9 and 10 we describe a pilot experiment in which we examine the performance of our algorithms on varying amounts of errors. In Sections 11 and 12 we contrast our work with other studies in the field and describe the large amount of work which still remains to be done.

2 A biology primer

Readers who are already familiar with the process of physical mapping can skip this section. Readers who desire more details than present in this section are encouraged to read Brown's or Nelson and Brownstein's books[5, 28].
The essence of the physical mapping process is as follows. The experiment begins with a sample of target DNA (recall that DNA is a linear sequence of base-pairs A,C,G, and T.) Pure samples of the target DNA are cut at specific points and then each fragment of DNA is inserted into a circular DNA molecule called a *vector* to produce a *recombinant DNA molecule.* The DNA fragment incorporated into the vector is called an *insert.* The vector

Contents

Figure 1:

transports the DNA fragment into a host cell. Within the host cell the vector multiplies, producing numerous identical copies of itself, and therefore, of the DNA fragment it carries. When the host divides, copies of the recombinant DNA molecule are passed to the progeny and further vector replication takes place. After a number of clone divisions, a colony, or *clone*, of identical host cells is produced. Each cell in the colony contains one or more copies of the recombinant DNA molecules. The DNA fragment is now said to be cloned.
It is possible to construct *probes* of various types. In general a probe is a piece of DNA which is complementary to some section of the target DNA. Probes may be short random sequences, copies of the ends of clone fragments, genetic markers, or practically any previously identified piece of DNA. One particular useful type of probe is the *sequence-tag-site (STS)* probe[26]. An STS is constructed to match a single site on the target DNA.
Given a set of clones and a set of probes it is in principle possible to determine which probes *hybridize* to which clones. A probe should hybridize (i.e. stick under experimental circumstances) to a clone if and only if its sequence is complementary to some subsequence of the clone. Thus a record of all hybridizations tells us which probes are contained within which clones.

2.1 Experimental errors

Unfortunately, the simple process described above neglects many important shortcomings of actual experiments. Typically these shortcomings are described as errors in the hybridization data although some of them are strictly speaking merely deviations from the simple model.

Chimerism One important type of experimental error, *chimerism,* results from the cloning process itself. A chimera is a clone which contains two unrelated fragments of DNA. The formation of chimeric clones occurs with all vector types (phages, cosmids, YACs and megaYAcs) and can occur both in the adoption of the fragment into the vector and in the subsequent clonal replication. Furthermore the proportion of chimeric clones tends to increase as the size of the DNA insert increases. Since the presence of chimeric clones greatly complicates the mapping process, great effort has gone into reducing their occurrence and/or detecting the chimeric clones.
There are several methods used for chimerism reduction. They are all characterized by a trade-off between labor-intensive procedures and accuracy. One method uses physical mapping of the ends of the YAC insert. It involves the isolation of the ends of the YAC insert which requires a considerable amount of experimental effort. Although very accurate in detection, the method is impractical for the detection of every chimeric clone. Another method uses fluorescence in situ hybridization (FISH). It is also a very laborious method which requires sophisticated techniques not available in all laboratories. Also, it has sensitivity limits and it fails to detect short noncontiguous sequences. Another difficulty that the methods must face is the one of being able to distinguish chimeric clones from clones subject to cotransformation events (i.e., the introduction, and stable independent maintenance of two or more YACs in the same cell).
Despite the methods developed for chimerism detection the percentage of chimerism in genomic libraries continues to be high. When the YAC technology was developed in 1987,

the inventors predicted that the technology would suffer from some chimerism, with an estimate of about 10% chimerism in the clones of a YAC library. In the clone libraries that were used in the creation of the first high-resolution maps [6, 13, 31] the chimerism was discovered, however, to occur at much higher percentages – reaching 40% in the chromosome 21 map and about 59% in the Y chromosome map. Also, the frequency of chimerism has been estimated at 40-60% for the two most widely used human YAC libraries [17, 28]. The recent advance in the discovery of megaYACs[7] seems to confirm the expectation that the larger the YAC is, the more chimerism is introduced in the library.
In the face of all these experimental difficulties and the fact that chimerism is intimately connected to the basic operations of recombinant technology, the computational support for mapping chimeric clones is vital for the creation of reliable physical maps of the chromosomes.

Deletions Deletions are another way in which gaps between fragments can occur. Even when a single fragment is inserted in a vector, a piece of the fragment may be deleted in the replication stage. The result is that the clone represents two fragments which are not contiguous in the genome, i.e. a chimera. Some deletions appear to be random and some are the result of "unclone-able" genomic regions in regular or megaYACs. It was even conjectured that there may be an unclone-able region of the human genome, on the average, for every 2-3 million bases. Particularly, the megaYACs contain a lot of deletions; it seems that they are internally scrambled. The splicing mechanism of yeast, which is part of the yeast's system of DNA repair, seems responsible for putting together pieces of non-yeast DNA, i.e., chimeras and deletions. Other factors responsible for these errors are repetitive sequences and fragility of sequences. Indeed, some sequences of the human genome containing large number of repeated sequences tend to be unclone-able, as they seem to trigger the yeast's repair system. Other sequences seem too fragile to be inserted without breaking them into pieces.

Hybridization errors The process of hybridization of probes to clones is complex both biologically and experimentally. Inevitably there are certain percentages of both false positive (incorrect recording that a probe hybridizes to a clone) and false negative (the opposite error) hybridization results.
One reason for false hybridization results is that a probe may not hybridize strongly to its location. Clones of different sizes and relative positions to the probe site may have conformations which make hybridization more or less likely. This sort of problem leads mostly to false negatives.
Another reason for false hybridization results is that while a probe may have a unique exact matching site on the target genome there may be other sites that are pretty good matches. Under some hybridization conditions this may yield a false positive.
The number of probe/clone pairs which must be checked can be very large. Therefore it is common to pool clones and check for hybridization[23, 30]. Often the pooling techniques do not precisely define which clone in a pool was the cause of a hybridization. In this case either false positives or false negatives may arise. In addition the pooling techniques may introduce correlations between false positives and/or false negatives.

The reading of a gel to determine hybridization is not simple. Sometimes the human or machine reader simply makes a mistake. Since the volume of data is already high it is not common to do redundant checks for accuracy. Again this can lead to either false positives or false negatives.

Other errors Although we will concentrate on resolving the problems due to chimeric clones and hybridization errors there are several other types of errors common in physical mapping.
Repeated occurrences of probes. The knowledge that each probe sticks to a unique location in the genome greatly increases the information yielded by a clone/probe match. STSs usually have a unique – or almost unique – occurrence in the genome and, therefore, an STS actually defines a position in the genome. Other types of probes, however, may have a large number of occurrences, and therefore, the information they provide is much weaker. An STS is usually difficult to find while generating small probes is usually routine. Although it is computationally harder to deal with probes with repeated occurrences, it is important to use the information they can provide due to the simplicity of the experiments involving them.
Errors in restriction fragment fingerprinting. Restriction maps are difficult to construct for the entire genomes because the sites for the most suitable enzymes are distributed non-randomly and are sometimes blocked by the action of methylation systems; restriction maps also fail to address the need of most map users for ready access to the cloned DNA. Errors specific to restriction fragment maps include: measurement errors in the lengths of fragments, missing fragments, and extra fragments (due to a cut not occurring at some sites).
Gaps are regions of the genome not represented in a map. Typically there are many gaps in a map, and for statistical reasons, complete closure of a map for a large genome is practically impossible. The problem is not only due to the random sampling of the genome libraries – some libraries do not contain all the sequences present in the genome. It is estimated that maps for the human genome will be likely to contain between 200 and 1000 gaps.

3 A theory of physical mapping

Despite the many experimental and theoretical studies of physical mapping [1, 10, 17, 27] the very notion of a physical map is still poorly understood. In this section we present a formal mathematical model of the physical mapping problem using hybridization data. We use this model to investigate the algorithmic tractability of the problem. In particular we quantify the amount of information inherent in hybridization data. This will allow us to evaluate the performance of mapping algorithms in a fair and rigorous way.
We start by examining hybridization data corresponding to "perfect" experiments. That is, we assume in this section that the hybridization matrix is error free (all probe/clone overlaps are faithfully identified and no clones are chimeric). In the next section we will expand the model to include errors. The evaluation of these error-free data sets will show that there is a non-trivial structure to the information which we can hope to algorithmically retrieve. An understanding of this structure will be critical to the analysis of noisy data, that is, data in which hybridization errors and cloning errors occur.

In particular, we will show that many properties of the target DNA are not retrievable from a hybridization matrix alone. In fact, we will show that often there are several possible orders of probes along the target DNA which could have yielded the hybridization matrix. In these cases it is not reasonable to expect an algorithm to always determine the true order of the probes. Instead we show that an algorithm of Booth and Lueker [4] can be used to encode information which is true of all ordering which could have led to the hybridization matrix and thus surely is true of the actual order.

3.1 Error-free hybridization experiments

The basic components of a physical mapping experiment are a target *genomic region,* a set of *clones,* a set of *probes,* and a *hybridization matrix.*

The goal of a mapping effort is to produce a map of some target **genomic region.** Let $\mathcal{G}$ be the genomic region for which we desire a genomic map. For the purposes of mapping, the region $\mathcal{G}$ is completely determined by its sequence of base pairs. We represent this sequence as an interval $I(\mathcal{G})$ over the real line.

Experimentally, the target genomic region is studied through a set of **clones.** Each clone consists of one or more fragments of the target genome. Each fragment is naturally associated with the subinterval of $I(\mathcal{G})$ corresponding to its piece of $\mathcal{G}$. When a clone is a single fragment we call it a *regular clone* and when its consists of at least two fragments we call it *chimeric.* (Throughout Section 3 it will be assumed that all clones are regular.)

Information about the clones is gathered through hybridization with **probes.** An STS probe is designed to define a single location on the genome. Although an STS has measurable extent on the genome, for our purposes it is reasonable to consider it to mark a point location. Thus each probe is associated with a point in the interval $I(\mathcal{G})$.

Genomic Placements Using the correspondence between components and the interval $I(\mathcal{G})$ we can now formally define the input to a mapping experiment. The experiment consists of a set of clones, $\mathcal{C} = \{C_1, C_2, ..., C_m\}$ and a set of probes $\mathcal{P} = \{P_1, P_2, ..., P_n\}$. Each regular clone, C_i, is defined by the location of its endpoints b_i and e_i on the interval $I(\mathcal{G})$. (Chimeric clones are defined by several pairs of endpoints.) Each probe, P_j, is defined by its location p_j on $I(\mathcal{G})$. We call the set of clones plus the set of probes plus the positions of all endpoints and probes the *genomic placement* of the experiment. We represent the placement as $(\mathcal{G}, \mathcal{C}, \mathcal{P}, \texttt{pos})$ where `pos` is the function which maps fragment endpoints and probes to their position on $I(\mathcal{G})$. Ultimately, the goal of a physical map is to find out as much as possible about the genomic placement as possible.

Fingerprints and Hybridization matrices The wet-lab portion of a physical mapping experiment takes genetic material (which can be described by a genetic placement) and returns experimental results. Just as a genomic placement can be used describe any input to a hybridization experiment, we can formalize the possible outputs of an experiment. For a given set of clones, the *fingerprint* of a position in $I(\mathcal{G})$ is defined to be the set of clones that contain that position. We will see that the output can be described as a set of fingerprints.

An actual experiment records the hybridizations of some probes with some clones. The results of all the hybridization tests can be encapsulated in a single **hybridization matrix**. The *hybridization matrix* A is defined by: $A[C_i, P_j] = 1$ if the interval of C_i is determined experimentally to hybridize to probe P_j and 0 otherwise. (The recording of uncertainty about the hybridization result is possible by using values between 0 and 1 but we leave this extension for future work.) In practice the hybridization is rarely tested directly but rather through a pooling scheme which may introduce biases. In this section we assume that the testing is somehow done perfectly.

It should be noted that the hybridization matrix does not contain any direct information about the actual position of the endpoints of the clones or of the probes. Instead each probe's hybridizations corresponds to the fingerprint of the probe's position. Let us call a set of positions *complete* if their collection of fingerprints contains all possible fingerprints for the given clone arrangement. That is, the fingerprint of any position not in the set would be the same as some fingerprint in the set. If the set of probe positions is complete then no additional fingerprints can be determined. Having multiple probes with the same fingerprint may give added reliability for data containing errors and gives some evidence of the relative size of fragments we will see that in a certain sense only the fact that there exists at least one probe with a given fingerprint is useful to physical mapping.

A hybridization event can equally well be described as a probe hybridizing a clone or a clone hybridizing a probe. This duality between clones and probes carries over to fingerprints. The fingerprints described above might be called *probe fingerprints* (i.e. for P_j the set $pf(P_j) = \{C_i \mid A[C_i, P_j] = 1\}$ while a *clone fingerprint* (i.e. for C_i the set $cf(C_i) = \{P_j \mid A[C_i, P_j] = 1\}$ could be defined analogously.

The fact that the output of a hybridization experiment is restricted to fingerprint information will have a major impact on the possible maps which can be algorithmically determined. In particular, we will show that the exact sizes of the clones and the exact distances between probes are not determinable.

3.2 Mapping Information

The basic question of this paper is *"What information about the genomic region can be determined from a given set of experimental results?"* We can make this question somewhat more formal by asking "What information about the genomic placement can be determined from a hybridization matrix?" We still, however, lack a formal definition of information. In this section we start with some examples of possible types of information and then show that it is only possible to compute some types of information from the hybridization matrix. Some simple facts about the genomic placement are readily apparent from the hybridization matrix. The positive information that a clone contains a probe (that is, the probe's position is within the clone's interval) is directly contained in the matrix. If two clones contain the same probe (and the probe occurs uniquely on the genome) then it can be deduced that the clones overlap. The negative information that a clone does not contain a probe can be used to refine the overlaps deduced by positive information.

The following two examples illustrate the interplay of positive and negative information.

Example 1. Suppose that clone C_1 contains two probes P_1 and P_2 but no other, and clone C_2 contains P_1, P_2, and P_3. The fact that both clones share at least one probe shows that

the clones overlap, the fact that clone C_2 contains an additional probe shows that on at least one side its interval extends beyond the interval of C_1.
Example 2. Let us now consider three regular clones C_1, C_2, C_3 and three probes P_1, P_2, P_3. Suppose that C_1 contains all three probes, C_2 contains P_2 and P_3, and C_3 contains only P_3. Then the positive information tells us that every pair of clones overlaps. This is enough to tell us that the clones mutually overlap but is not enough to tell their order. The negative information, however, allows us to determine that the order of the positions within the placement of the probes and begin points of the clones must be is $b_1, p_1, b_2, p_2, b_3, p_3$ or its reverse. All the end points must come after p_3 but their relative order is not defined.
Intuitively, probes yield information by witnessing regions in which clones overlap or regions in which clones do not overlap. We have already noted that there are a fixed number of possible probe fingerprints possible. Each fingerprint witnesses a region of overlap. (It is possible, when one fragment is completely contained within another, for distinct regions to share the same fingerprint, however.)

Mapping properties In order to define mapping information more formally we define a *mapping property.* A mapping property distinguishes between two different genomic placements. Formally a mapping property divides the universe of possible genomic placements into two sets: those having the property and those not having the property. For example, one possible property is "There is a probe at position p in $\mathcal{G}$." Another is "Probe j is between the position of the beginning of clone i and the beginning of clone i'."

Definition 1 *A* mapping property *is a function from the set of all genomic placements to the set* $\{0, 1\}$. *If the function is 1 on a placement then it is said to have the property otherwise it does not have the property.*

The information contained in a map can be defined as a collection of mapping properties about the genomic placement. Several natural levels of information can easily be defined. We will show that, though all are natural to define, it is not possible to determine some properties from the hybridization matrices alone.
Based on the definition of a genomic placement the maximum amount of information which could possibly be in the matrix is the position of the endpoints of each clone and the position of the probes. We define = *Complete Mapping Information* as knowing the beginning and end position of every clone and the position of every probe in $\mathcal{G}$.
Complete mapping information allows us to determine the size of each clone and the distance between two consecutive probes. However, the hybridization matrix does not directly encode sizes so we may have to settle for relative positions rather than absolute positions. The maximum amount of relative position information is a total order on the endpoints of clones and the positions of probes. We define *Complete Order Information* as knowing the *order* of the clone endpoints and probe positions along $\mathcal{G}$.
For ease of exposition we ignore here the possibility that the position of two components coincide. It is straightforward to extend the ordering to allow sets of equal components within the ordering. Typically we will not be able to distinguish a total order from is reverse order but we will not repeatedly point this out.
As will become clear later, the relative order of some components may not be available. We might hope that a total order of the probes or a total order of the clone endpoints would

be possible. Thus we define *The True Probe Order Information* as knowing the *order* of the probe positions along $\mathcal{G}$. Similarly *The True Clone Order Information* is knowing the *order* of the clone begin and end points along $\mathcal{G}$.
In fact, as we will show below, it is rarely possible to determine all the information at these total ordering levels. Instead, it will be possible to quantify exactly what information can be determined from the matrix.

3.3 Algorithmically retrievable mapping information

Having defined the input and output of an experiment and the information one might hope to retrieve from an experiment we are now ready to show that some information is irretrievable. Intuitively, a property is irretrievable from a hybridization matrix if, based on the hybridization matrix alone, one is not sure whether the property holds for the input genomic placement or not. If information is irretrievable then *no* algorithm, regardless of computation power, can determine the information with 100% certainty. Formally:

Definition 2 *A genomic placement is* consistent *with a hybridization matrix if the matrix rows can be place in one-to-one correspondence with the placement's clones and the matrix columns can be place in one-to-one correspondence with the placement's probes so that the entry in the matrix is a one if and only if the corresponding probe is contained in the corresponding clone.*

Definition 3 *A mapping property for a hybridization matrix is* algorithmically irretrievable *if there exist two genomic placements consistent with the matrix such that for one the mapping property is true and for the other the mapping property is false.*

3.4 Probe restrictions

As a first step toward determining the retrievable information from a hybridization matrix we define the *probe restriction* of a placement by a matrix. The intuitive idea is that if a matrix does not contain a complete set of fingerprints then there is missing information. Furthermore the fingerprints only witness regions of overlap and not the exact endpoints of fragments.
Since a hybridization matrix may not contain a complete set of fingerprints for its input genomic placement, we define a canonical placement for which the matrix is complete.

Definition 4 *Let $\mathcal{E} = (\mathcal{G}, \mathcal{C}, \mathcal{P}, \mathtt{pos})$ be a genomic placement. The* probe restriction *of $\mathcal{E}$ is $\mathcal{E}|_{\mathcal{P}} = (\mathcal{G}, \mathcal{C}', \mathcal{P}, \mathtt{pos}')$ where: $\mathcal{C}'$ consists of one clone for each clone in $C_i \in \mathcal{C}$ such that at least one probe is in C_i, the value of $\mathtt{pos}'(P_j) = \mathtt{pos}(P_j)$, and values of $\mathtt{pos}'(b'_i)$ and $\mathtt{pos}'(e'_i)$ equal the position of the first and last probe contained in the corresponding clone C_i.*

Thus the probe restriction is derived from a placement by shrinking each clone so that C'_i is the interval from the position of the first probe contained in C_i to the position of the last probe in C_i.

Lemma 1 *For any genomic placement, $\mathcal{E} = (\mathcal{G}, \mathcal{C}, \mathcal{P}, \mathtt{pos})$, the set of probes $\mathcal{P}$ is complete for the set of clones in $\mathcal{E}|_{\mathcal{P}}$.*

Theorem 1 *Any mapping property which is true of a genomic placement but not true of its probe restriction is algorithmically irretrievable.*

Proof: Any hybridization matrix which is consistent for the genomic placement is also consistent for its probe restriction. ∎

Unless a hybridization matrix happens to contain probes which coincide with every endpoint of a clone then for at least one clone the position of the clone's endpoints are different in the genomic placement and its probe restriction. Even if the hybridization matrix does contain all such probes a genomic placement in which the positions are perturbed slightly will yield the same matrix. Thus unless the probes occur at every possible position (thereby eliminating the possibility of perturbation) the exact position of clone endpoints or probe positions is irretrievable. Excluding this highly improbable situation we have the following corollary.

Corollary 1 *Level 1 of mapping information is not algorithmically achievable.*

Since Theorem 1 and its corollary precludes any property which depends on exact positions of probes and clones from being retrievable we define a restricted class of properties which depend only on probe order.

3.5 Consecutive ones matrices

Since information about the exact position of clones and probes is irretrievable we turn to information about *relative* position. The importance of probe restrictions in defining the available information points toward looking at the relative order of probes. If we determine the relative order of the probes then we can retrieve the relative order of the clones in the probe restriction, which is as much as we can expect. In the probe restriction, in some sense, a clone *is* its ordered set of probes.
In order to examine probe orders we introduce the following notation. Let $Perm[n]$ be the set of permutations of the set $\{1, ..., n\}$. A permutation $\pi \in Perm[n]$ is a *probe order* and we denote it by $\mathcal{P}^\pi = (P_{\pi(1)}, ..., P_{\pi(n)})$. Finally, let A^π be the matrix resulting from permuting the n columns of A according to the permutation π.

Definition 5 *A probe order π is* consistent *iff for every clone C of $\mathcal{C}$, the probes in the clone fingerprint of C, occur consecutively in π.*

Definition 6 *The* true order π_0 *of the genomic placement $(\mathcal{G}, \mathcal{C}, \mathcal{P}, pos)$ is the order of the probes on $\mathcal{G}$.*

Certainly, for error-free matrices, π_0 is consistent, but there need not be a unique consistent probe ordering. In extreme cases requiring consistency may not eliminate any permutations. For example, if the number of clones is equal to the number of probes and clone i hybridizes probe j if and only if $i = j$ then any probe ordering is consistent. On the other extreme, it is possible to construct a matrix for which only one order (and its reverse) is consistent. In simulations such as those described in Section /refsec:data-qual, an $m \times n$ hybridization matrix of regular clones both has a number of consistent probe orders which is exponential in n and has exponentially many non-consistent orders.

Despite the lack of specificity of the requirement that a probe order be consistent it is nonetheless an important concept for mapping. In particular it is often interesting to know whether any consistent orders exist. In the literature such matrices occur in many contexts and are defined as follows.

Definition 7 *A* 0/1 *matrix A has the* consecutive ones property *(C1P for short) if there exists a permutation* $\pi \in Perm[n]$ *such that in the column-ordered matrix* A^π *every row has all the ones occurring consecutively. Such a permutation is called a* C1P-permutation *for A.*

Thus a matrix is C1P iff there exists a consistent order for it.
Although the matrix A^{π_0} is C1P when all the clones are regular and there are no hybridization errors, it is not clear whether one can make stronger statements about it. As we will see, in some ways knowing that the true order is consistent is all that one can know. Therefore we define the basic notion of a "map" as follows.

Definition 8 *Let* $(\mathcal{G}, \mathcal{C}, \mathcal{P}, pos)$ *be a genomic placement and A its hybridization matrix. A* map *for the genomic placement is a column-ordered matrix* A^π*, where* π *is a consistent probe order. The* true map *of the genomic placement is* A^{π_0} *where* π_0 *is the true probe order.*

3.6 Booth and Lueker's PQ-tree algorithm

Since the true order is a consistent order it is desirable to determine all possible consistent orders of a hybridization matrix. Fortunately, the problem of determining all C1P orders is well understood. In fact, a linear algorithm due to Booth and Lueker[4] constructs the set of all such permutations in a concise way – the so called *PQ-trees.*
Since the PQ-trees will be important to the remainder of this section we give a sketch of the theory of PQ-trees here. Readers interested in a more detailed discussion are encouraged to read Booth and Lueker's paper. The PQ-tree is a method of encoding a set of permutations. It consists of three types of objects, leaf-nodes, P-nodes, and Q-nodes. The leaf-nodes contain the basic element being permuted – in our case the columns of the matrix. We label the leaf nodes $\{1, 2, ..., n\}$.
The allowable orderings of the leaf nodes are constrained by their inclusion in P and Q nodes. All the leaf nodes in a Q-node are kept in a *fixed* order. That is to say, in any permutation in the set encoded by a PQ-tree the leaf nodes within a Q-node appear in the same order (but in either possible direction, eg. $1, 2, 3$ or $3, 2, 1$.) The leaf nodes in a P-node, on the other hand, have no specified order. That is to say, in any permutation in the set encoded by a PQ-tree, the leaf nodes within a P-node can occur in any order (eg. $1, 2, 3$ or $1, 3, 2$ or $2, 1, 3$ or $2, 3, 1$ or $3, 1, 2$ or $3, 2, 1$.)
Thus if all the leaf nodes were in a single Q-node then the set of permutations encoded would be just the order within the Q-node and its reverse while if all the leaf nodes were in a single P-node then the set of permutations encoded would be *all* permutations of the leaf nodes.
A *PQ-tree* allows P and Q nodes to be placed within other P and Q nodes. Thus each constituent of a node can either be a leaf or a set of possible orderings of a subset of

the leaves. The set of permutations which keep the sets $\{1,2,3\}$, $\{4,5,6\}$, and $\{7,8,9\}$ together but allow any ordering within sets or among sets (i.e. $3,1,2,8,9,7,4,5,6$ but not $1,2,4,3,5,6,7,8,9$) can be represented as a P-node containing three P-nodes, each of which contains the leaves in one set.

The surprising property of PQ-trees is that each PQ-tree corresponds to all possible consistent orderings of a 0/1-matrix like our hybridization matrices and that each matrix has a corresponding tree. Thus, by using the Booth and Lueker algorithm it is possible to encode *all* possible consistent maps in one PQ-tree.

Knowing all possible consistent maps allows us to be certain that we have the true map in our set, but the number of consistent maps may be very large. The PQ-tree, however, does more than just list all consistent maps – it tells us a great deal about the structure of these maps. Intuitively it tells us two types of information about the maps. On the one hand it encodes "consensus" information, that is parts of the ordering which are true of all consistent maps, through the leaves contained directly in Q-nodes. On the other hand it encodes "restricted variability" through the grouping supplied by the P and Q-nodes.

Due to the fact that the PQ-tree can nest P nodes and Q nodes there is actually a continuum from strong consensus information to weak variability information. At the strongest end are leaves within Q nodes. The order of these leaves is known to be the same in all consistent maps and therefore in the true map. Next strongest are leaves within P nodes. The leaves are known to occur together with no intervening leaves in all consistent maps. Although their local order is unknown the global fact that they occur together is guaranteed to be true of the true map.

As one works one's way up from leaf nodes the information encoded refers to larger and larger portions of the genome (when the PQ-trees refer to hybridization matrices). Thus a high level Q node might represent the fact that several segments occur in a fixed order although the order within the segments is not fixed. A high level P node gives only the weaker information that there are segments but nothing about their order.

Thus, given the PQ-tree for a hybridization matrix the uncertainty about the true order is due to the P-nodes. If no P-nodes exist then the tree collapses to a single Q-node. On the other hand, if any P-nodes exist then the complete true map is not retrievable.

Lemma 2 (Unique consistent map) *A C1P matrix has exactly two C1P-permutations, a permutation and its reverse, iff its PQ-tree consists of exactly one non-leaf node which is a Q-node.*

Lemma 3 (P-node information about the true map is irretrievable) *It is algorithmically irretrievable to find the order of the children of a P-node in the PQ-tree of a hybridization matrix.*

Proof: By definition there are two orderings of the children which result in consistent orderings for the matrix and each of these orderings corresponds to a different probe-restricted genomic placement.

3.7 The true map is almost never algorithmically retrievable

If the hybridization matrix, the input of the mapping problem, would assure that the map is unique (up to reversal) then the task of mapping would be clear: find "the map". In this

case, the map must be the true one. The next theorem relates the question of finding the true map to the question of finding a unique consistent order of a matrix.

Theorem 2 *Let us consider a genomic placement and its hybridization matrix A. There is an algorithm which takes as input A and computes the true map if and only if the matrix has a unique consistent order (up to reversal).*

The theory of when a matrix has a unique will be presented elsewhere[19] but the conditions necessary are quite strict. We do not know of any biologically relevant model of matrix generation in which the resulting matrices are likely to meet the conditions for unique C1P ordering. In other words *the hybridization matrix will almost never have a unique C1P ordering.* This means that it will almost always be beyond the capability of any algorithm, irrespective of the amount of computing time used, to compute the true map.
Given that the true map cannot be computed the natural question then is: what is a fair goal for a mapping algorithm? We know that the true map must be C1P but there may be exponentially many C1P maps to consider. Since computing the one true map seems to be problematic we instead ask what *information* about the true map is retrievable. For example, although we cannot know the complete order corresponding to the true map, we might be able to determine a partial order which is true of all C1P maps.

3.8 PQ-trees encode information in common to all maps

We are now looking for information which is true of *all* C1P maps and therefore true of the true map. Fortunately the PQ-tree gives us exactly what we want.

Theorem 3 *A mapping property is retrievable from a given hybridization matrix if and only if the property is true for all genomic placements which are consistent with a permutation encoded by the PQ-tree of the hybridization matrix (or for all genomic placements it is false).*

Proof: (if) In order for the property to be irretrievable there would have to be two genomic placements consistent with the matrix, one for which the property was true and one for which the property was false. However, by construction the PQ-tree encodes all consistent permutations so a genomic placement which is consistent with the matrix must be consistent with some permutation encoded by the PQ-tree. By assumption such placements either all true or all false for the property and thus the property is retrievable.
(only if) By the definition a mapping property is irretrievable if it is true of one consistent placement and false of another. ∎

We have already seen that the order of the children of a P-node is irretrievable. Theorem 3 implies that, the order of the children of a Q-nodes meets our definition of retrievable. Similarly, other structural properties of the PQ-tree such as that certain probes are in the same sub-tree are retrievable. It seems, in fact, that the only useful retrievable properties are those which describe the structure of the PQ-tree.
Although we have been forced to progressively reduce our goal concerning the information about the genomic placement, all has not been lost. We have shown that it is possible to

retrieve information about the genomic placement which is encoded in the PQ-tree and in some sense this is the most information which we can expect to retrieve.
Unfortunately, when we look at mapping in the presence of errors such an algorithm will not exist. In the next sections we examine in more detail the sort of information present in the PQ-tree. Since we do not know how to create PQ-trees (or any similar structure) which the data contains errors, it will be our goal to describe as much information encoded by the PQ-tree as possible in terms which are independent of the PQ-tree.
As noted in Section 3.6 the PQ-tree contains a complex amount of mapping information common to all maps. The recursive structure of the tree induces a hierarchy from local information (pairwise probe adjacencies) to global information (components which must be together.)

3.8.1 Local resolution: adjacent pairs of probes

The finest level of resolution concerning the ordering of the probes is pairwise adjacencies. Let us consider a revealing case: mapping instances with unique (up to reversal) maps. What can we say about the pairwise adjacencies of probes in the unique map? In a trivial way, they are the same in all maps.

Definition 9 *Let us call a pair of probes P_1, P_2 a* strong adjacency *if P_1 and P_2 are adjacent in every map, i.e. consistent order. An adjacency of a map is* weak *if it not strong.*

In this terminology, an instance of the mapping problem has a unique solution if and only if all adjacencies are strong. What are the strong adjacencies when the mapping problem has multiple solutions? The answer is: the set of adjacencies given by the Q-nodes which have only leaves for children.
Clearly for any map an adjacency of two probes is strong, or "fixed" if they are adjacent in all maps. On the other hand, an adjacency of two probes in map A^{π_1} is weak when there exists another map A^{π_2} such that in this map the two probes are not adjacent.
As we consider probe orders up to reversal, this distinction, adjacent or separated constitutes a complete analysis of pairwise properties. This is exactly the mapping information captured by the Q-nodes which contain only leaves. For weak adjacencies, we can find out from the PQ-tree all their "degrees of freedom". For example, one can determine all probes which are ever adjacent in any map to a given probe.

3.8.2 Global resolution: components in the clone-cover graph

At the other extreme from strongly adjacent probe pairs, there are probes pairs for which there is no information relating the probes within the pair. Ultimately, all information concerning the relative position of probes derives from having clones which "cover" more than one probe. Two probes that are adjacent on $\mathcal{G}$ will be likely to be "covered" by clones, i.e., both would hybridize to a number of clones. The more clones that cover them, the stronger is the linkage between them. On the other hand, probes that are faraway on the genome will be unlikely to be covered by a clone.
The structure of this coverage is recorded in the following graph.

Definition 10 *The* clone-cover graph *of A is given as follows. $CC = (V, E)$ where, $V = \mathcal{P}$ and $E = \{(P, P', w) \mid P, P' \in \mathcal{P}, w =$ number of clones containing both P and $P'\}$.*

Clearly, the connected components of the graph CC (disregarding edges of weight 0) correspond to disjoint and unlinked regions on $\mathcal{G}$. In the PQ-tree terminology, they are exactly the children of a P-node at the root of the PQ-tree. By the definition of a P-node, any ordering of its children defines a valid $C1P$-permutation. It is then a corollary of Theorem 3 that the true order of the components of CC cannot be computed. That is, pairs of probes from distinct components have no information relating them.
On the other hand, we do know something about probe pairs within a component. They should not have probes from outside the component between them.
Since the clone cover graph removes explicit information about clones and retains only counts of their coverage it is interesting to ask how much information is lost. It turns out that the clone cover graph still contains enough information to determine whether the matrix is C1P or not.

Lemma 4 *The mapping information contained in the clone cover graph $CC(A)$ is sufficient for determining whether A is C1P.*

3.8.3 Matrix singularities

There are certain properties of a matrix which guarantee that the PQ-tree will lose information. Two examples are non-hybridizing probes and redundant probes.
It is possible that a probe does not hybridize any clones. Its corresponding column in the matrix will be all zeroes. The probe will thus be a singleton component in the clone cover graph and all we know about it is that it does not fit inside any other component. Clearly we can just discard these probes and lose nothing. However, in the when the data contains errors it may be difficult to identify these probes since errors may cause them to not correspond to columns which are all zeroes.
Another possibility is that two or more probes hybridize exactly the same clones. In this case they will be inside a leaf P-node since any ordering yields the same matrix. The probes may be adjacent on the genome or non-adjacent. In the adjacent case no clone happened to begin or end between them. Perhaps they are very close together or perhaps the intervening region is unclone-able. In the non-adjacent case it may be that all clones which start between them also end between them. This seems to be a less likely possibility but definitely can occur.
As with non-hybridizing probes it is tempting to fix the matrix by removing all but one copy of each redundant set. However, it may again be difficult to do so when errors make probes which hybridize the same clones appear different.

4 Experimental errors

In the previous section we showed that the theory of PQ-trees captures the information available in a hybridization matrix from an "ideal" experiment. In the ideal experiment every clone corresponded to a single contiguous region on the target DNA, every probe

corresponded to a unique location on the target DNA, and every overlap of clone and probe was correctly witnessed by a hybridization experiment and faithfully recorded in the hybridization matrix.
Unfortunately molecular biology, like all experimental sciences, does not produce perfect data. As was described in Section 2 the actual experiments are much more complicated then the simple model used in the last section. A variety of errors derived from the lack of desired precision of the experiments – some of which are inherent to the current technology – make the resulting data less than ideal. The determination of which probes stick to which clones will yield both false positives and false negatives. Some sections of the target DNA will tend to shatter into tiny pieces which are lost while other sections will contain no probe sequences. Some clones will correspond to multiple regions on the target DNA.
In this section we will formalize the effect of errors on hybridization matrices and describe what an algorithm might be expected to do to counter their effects. Unlike the error-free case, we will not be able to point to an algorithm which captures the information in the hybridization matrix. Instead we discuss what sorts of information are likely to be retrievable and give some possible strategies for finding information.

4.1 Extending the model to data with errors

Prior to the inclusion of errors, the major pillar of our theory was the fact that, if the columns of the hybridization matrix were ordered in the same order as the probes in the genomic placement then the matrix was C1P. Unfortunately all our types of errors invalidate this guarantee. Thus, we are led to ask again, "What does a map look like?"
One possibility would be that enough were known about the errors that we could at least have a statistical model of what the true ordering of the hybridization matrix would look like. Unfortunately, very little is known about the statistical nature of the errors. It is possible to place some bounds on the error behavior but no hard and fast rules exist. For example, one might assume that all (or most) clones will have one or two fragments or assume that no clone will have more than a few percent false positives but these would only be guesses. Therefore, we present a theory in which the nature of the errors is abstracted into a general cost function.
We begin our formalism by returning to our basic assumption: our goal is to retrieve as much information about the genomic placement as possible from the hybridization matrix. We know from the error-free case that at most we can determine the "true" probe ordering along the genomic placement. From the probe ordering we were able to infer from the matrix the probe-restricted placement which gave us an ordering of the clones.
When the data contains errors the probe ordering does not suffice to tell us the clone order. For example, if a clone's row in the true ordering is 001100100 then we could have a single fragment hybridizing probes 3 and 4 only (that is the clone without errors would read 001100000), or a single fragment hybridizing probes 3 through 7 only (that is the clone without errors would read 001111100), or two fragments, one hybridizing probes 3 and 4 only and the other hybridizing only probe 7, or many other possibilities. The three possibilities mentioned correspond to the least amount of just false positives, just false negatives, or just chimeric clones which could explain the row's value.
Thus we extend our definition of a map from the error-free case to require both a *permutation*

of the columns of the matrix and an *explanation* of errors which could lead to each row's value.

Definition 11 *An* explanation, η, *of a permutation of a hybridization matrix is a function which maps each row of the matrix to a set of fragments in a probe restricted placement.*

For example, the explanation of a C1P matrix might map each row to the single fragment which begins at the first probe in the row's block of ones and continues to the last probe. For non-C1P matrices the explanation may involve marking some entries of the matrix as being incorrect. In the error-free case we defined a consistent as meaning each row corresponded to a single continuous fragment (or equivalently that each row contained a single contiguous block of ones). We extend the notion of consistency to data with errors by allowing the explanation to fix hybridization errors and/or split chimeric clones.

Definition 12 *The probe order/explanation pair* (π, η) *is* consistent *if for every clone* C *of* $\mathcal{C}$*, the correspondence created by* η *causes all and only probes within fragments to have value one in the matrix.*

Since there are many possible explanations of any matrix row we need some way of determining which are best. We therefore define the concept of a *noise model.* A noise model encodes what is known or thought to be known about the possible errors in the data by specifying a set of explanations which "undo" the errors and a cost function which rates the likelihood of each explanation being correct. The cost function need not be a formal likelihood function but should as faithfully as possible reflect what is known about the errors.

Definition 13 *A* noise model, N, *consists of a set of possible explanations and a function which takes each explanation,* η*, on a fixed permutation of a hybridization matrix,* A^π*, to a cost* $f(\eta, A^\pi) \in \mathcal{R}$.

Just as we can extend the notion of consistency to data containing errors by including an explanation of the errors we can extend the notion of map to include not just and ordering but also an explanation drawn from the noise model.

Definition 14 *Let* $(\mathcal{G}, \mathcal{C}, \mathcal{P}, pos)$ *be a genomic placement,* A *its hybridization matrix, and* N *be a noise model.* *A* map *for the genomic placement is a column-ordered matrix* A^π *and an explanation* η*, such that* (π, η) *is consistent for* A*. The* true map *of the genomic placement is* (A^{π_0}, η_0) *where* π_0 *is the true probe order and* η_0 *describes the errors that actually occurred.*

As in the error-free case we are unlikely to be able to determine whether a particular map is the true map. In fact, we cannot at this point rule out any probe ordering because there may always exist some explanation which is consistent with the ordering for the matrix. We define three simple noise models:

Definition 15 *The* false positive only*, FPO, noise model allows explanations of the form: the entries of the matrix in set* S *are changed from 1 to 0 where* S *is any subset of matrix entries whose value is 1 and the resulting matrix is C1P. The mapping to fragments then proceeds as in the C1P case. The cost of an explanation is the size of* S.

Definition 16 *The* false negative only*, FNO, noise model allows explanations of the form: the entries of the matrix in set S are changed from 0 to 1 where S is any subset of matrix entries whose value is 0 and the resulting matrix is C1P. The mapping to fragments then proceeds as in the C1P case. The cost of an explanation is the size of S.*

Definition 17 *The* 2-chimera only*, C2O, noise model allows explanations of the form: the entries which are 1's in each row are partitioned into at most two groups such that each group occurs consecutively. The mapping to fragments then proceeds as in the C1P case except that rows having two groups are mapped to two fragments. The cost of an explanation is the number of rows having two groups.*

These simple noise models correspond to our three major error types. Each has the desirable property that the cost of the null explanation is always minimal and that if more errors are required by the explanation then the cost is higher. We call noise models with this property *monotonic* and we will see in Section 8 that monotonicity is a helpful property in algorithm design.
There are many possible variants. The cost of changing two adjacent zeroes to ones might be less than the cost of changing non-adjacent zeroes, thereby approximating the effects of deletions. The cost of changing two adjacent ones to zeroes might be higher than the cost of changing two distant ones, thereby penalizing the use of false positives to remove true chimera. The chimeric noise model might allow an unlimited number of groups per row. In this case the cost might be defined as the maximum number of groups per row or as the total number of groups.
We are not arguing for a particular noise model but merely providing a framework in which to define them. For example, a noise model which is designed to match one type of error might also be useful when other error types occur but the one type is dominant. The simple noise models also provide intuition for algorithm design.
Some noise models are very forgiving in that any permutation admits some explanation.

Lemma 5 *Under the FPO or FNO noise model, for any hybridization matrix and any permutation there always exists a consistent explanation for the permutation on the matrix.*

Proof: Under FPO all ones could be eliminated (or more reasonably all ones except those in a single block on each row). Under FNO all zeroes could be eliminated (or more reasonably all zeroes between the first and last one in each row.) ∎

In order to limit the power of forgiving noise models and in general to reduce the number of candidate permutation/explanation pairs we make the *parsimony hypothesis.*

Definition 18 *The* maximum parsimony set, MPS, *for a hybridization matrix is the set of permutation/explanation pairs with the lowest cost.*

Unlike the error-free case, in which we could limit our search to C1P maps and be certain that the true map was in the search set, limiting our search to the MPS does not guarantee that the true map is in the search set. We will often, thus, want to consider all permutation/explanation pairs with close to minimal cost since we are then more likely to include the true map.

4.2 NP-completeness barriers

It has been shown elsewhere[2, 16], though not in these terms, that finding even one member of an MPS for some noise models is NP-complete. (Readers not familiar with NP-completeness may want to look at Garey and Johnson's standard reference[15]. Intuitively, a problem being NP-complete means that computer scientists agree that no efficient algorithm exists which solves the problem exactly.) For example, the C2O model allows one to search for an ordering of the matrix with at most two blocks of ones per row and finding such an ordering is known to be NP-complete. The FNO noise model is closely related to the NP-complete problems of bandwidth and of envelope.
In fact, we conjecture it will be NP-complete to find a single member of the MPS for any noise model corresponding to real experiments.

4.3 Reasonable goals

Since the NP-completeness results imply that we cannot efficiently find a best permutation/explanation pair for a given hybridization matrix we are in a similar position to not being able to distinguish between C1P matrices in the error-free case. The difference is that now we have no algorithm like the PQ-tree algorithm to capture all the possibilities.
Thus we aim for a subset of what the PQ-trees told us. In particular we have looked at how many strong adjacencies we can find. We might hope that there will be regions of the genome that are so well covered by clones that even in the presence of errors there will remain strong adjacencies. We also ask what are the connected components in the CC graph. False negatives tend to increase the number of connected components while false positives and chimera tend to decrease them. We can also look for components that are more strongly connected. We might demand that there are no single edges whose removal would split the component into two components. This would remove some connectivity due to errors while having less effect on well-connected true components.

5 Evaluating Algorithms on Hybridization Data

It is a postulate of algorithmic work on physical maps that the ultimate test of an algorithm is its success on real genomic data. However, our limited understanding of real data (and the paucity of existing data for which we know the correct map) requires that algorithms be evaluated on synthetic data. In work which will be reported elsewhere we are working with the Los Alamos Center for Human Genome Studies and with the Whitehead Institute to evaluate our algorithms on real data but the purpose of this paper is to explore the evaluation of algorithms on synthetic data.
Our evaluation consists of two parts: using data designed to match the observed characteristics of a particular data set from Los Alamos and using data chosen to cover a range of possible data characteristics. Both sets of data are produced using a matrix generator based on a simple model of probe and clone placement and of error occurrences. The simplicity of the model makes it relatively easy to control the characteristics of the generated matrices. On the other hand the simplicity undoubtedly obscures some important details of actual matrices. One of the goals of our experiments has been to evaluate the generator. More

will be said about this in later sections. Here we simply describe the generator used.

5.1 Generator

5.1.1 Input

The hybridization matrix generator used in this work has the following essential inputs.

- Number of probes and clones. Each matrix will have one column for each probe and one row per clone. Error-free clones consist of a single fragment which is determined to hybridize a consecutive set of probes. (See below for more detail.)
- Error rates: chimerism rate, false negative rate, and false positive rate. Each clone has an independent probability of being chimeric and thereby consisting of two fragments rather than one. Each position in the matrix corresponding to a probe which hits a clone has an independent probability of being a false negative (and thus being recorded as a zero instead of as a one.) Each position in the matrix corresponding to a probe which does not hit a clone has an independent probability of being a false positive (and thus being recorded as a one instead of as a zero.)
- The range of clone sizes. Each clone has an independent probability of being a small clone or a large clone. Size ranges are specified for both small and large clones.

5.1.2 Description of the generator algorithm

The generator begins by choosing positions for the probes. The target genetic data is abstracted as an interval of the real line. Probe positions are chosen independently and uniformly at random in this interval. This leads to a Poisson distribution of their locations. The locations are sorted by position to produce the "correct" order of the probes.
Once the probes' positions on the target genome have been chosen each clone is generated and a row of the matrix is created corresponding to it. For each clone an initial fragment is chosen by randomly choosing a size range according to the input probability and then choosing a size uniformly at random in the range. A start position on the target genome is then chosen uniformly at random from those positions at which a fragment of this size could occur (i.e. not too close to the end). The fragment thus corresponds to a sub-interval of the interval corresponding to the target genome. The start positions are Poisson distributed while the end positions are not. Comparison with a generator which chooses both the start and end positions of the clones according to a Poisson distributions would be an interesting follow-on experiment. The current approach, however, allows rapid generation and a significant amount of control over clone sizes.
Once the start and end position of a fragment have been determined the identity of those probes whose positions fall in the fragment's interval can easily be determined. The size of the clone will determine the *expected* number of probes which hit the clone. If this clone is determined to be chimeric then a second fragment is generated in the same manner as the first. Since the two fragments may overlap the clone may or may not correspond to two intervals. In the row of the matrix created for this clone, each column corresponding to a probe which is not contained in the clone's intervals is initially set to 0. If a coin tossed

with the input false positive probability indicates a false positive then the matrix entry is changed to 1. Similarly, each column corresponding to a probe which is contained in the clone's intervals is initially set to 1. If a coin tossed with the input false negative probability indicates a false negative then the matrix entry is changed to 0.

5.1.3 Discussion

There are many aspects of this generator which we believe could be made more detailed as well as many questions we would like to answer concerning its biases.

1. Probe generation. Choosing positions based on double precision real numbers is of a finer grain than the actual genome – does this matter? Actual probes are often not random strings of DNA but have some genetic meaning. Is there some way to include this knowledge in the generator? In particular, some probes are known to be the PCR ends of clone fragments. What would be the effect of explicitly defining some probes to be fragment ends? It is also known that some regions of DNA are unclone-able. Should this be represented by gaps in between probes or by modification to clone generation?

2. Clone generation. The actual biological process of producing clones is quite complex. Can a better model be produced by generating base-pair sequences and then simulating the fragmenting of copies of the target DNA and the uptake by vectors? Is there some intermediate model which captures more of the biology than our current model?

3. Error generation. How should chimeric clones be treated? Should they be allowed to have more than two pieces? Should the individual pieces have a size distribution identical to non-chimeric pieces? In an earlier generator we made the size of the *combined* chimera be distributed equivalently to the size of non-chimera. This led to one or both pieces being very small. On the other hand the current approach leads to chimera being on average larger than non-chimera and thus providing more (though potentially misleading) information.

   Should two false positives or false negatives be correlated? Deletions might be modeled as correlated false negatives. However, for clones which hit at most three probes (which is true of most of our clones) a correlated false negative simply produces a smaller clone. Should the pooling strategies used for hybridization experiments be included to identify correlations in the pooling process? Should false positives be more common near clones than far from clones?

   Should the rate of false positives depend on the size of the matrix? A 1% rate of false positives corresponds to an average of two false positive per row with 200 probes and an average of one false positive per row with 100 probes. If the ratio of ones due to false positives to the ratio of ones due to true hybridizations is too high can any algorithm succeed?

4. Extensions. Some experiments yield hybridization data which is neither positive nor negative but which is instead an estimate of likelihood of hybridization. Can the matrix be given fractional entries to represent this data? It is difficult to produce

probes which correspond to unique locations on the target DNA. Can each probe be assigned a set of positions in order to represent this type of data?

5.1.4 Technical details

In order to ensure that any algorithm running on the generated matrix does not make use of the known generation order of the probes or clones both the rows and columns of the matrix are randomly permuted before being output. We believe that these permutations are critical to fair evaluations of algorithms.

5.2 Mimicking real data

Our first use of our generator was to attempt to produce matrices which were as similar as possible to real data. Norman Doggett's group at Los Alamos (LANL) was kind enough to share some early hybridization data with us. This data consisted of 261 STS probes and 424 clones. They had recorded 1101 clone-probe hybridizations. Recently they have increased the number of probes and clones and by using break-points and other non-hybridization data have published a physical map of their data. We were, initially, interested in how much information was available in their early data so we have not used any of the later data.

Given a hybridization matrix, it is not immediately obvious what characteristics to attempt to match. We decided that as a first approximation we should attempt to match the distribution of clone sizes. The average number of probes hit by a clone was approximately 2.6 so we endeavored to produce matrices with an average of 2.6 ones per row. In the LANL data every clone hit at least one probe (presumably clones hitting no probes had been removed before giving us the data) and no clone hit more than 11 probes. More than one third (160 out of 424) of the clones hit only a single probe and thus provided no mapping information. More than three quarters of the clones hit at most three probes.

In order to simulate matrices with these characteristics we made 95% of our clones small, in the 2 to 3 range, and the remaining clones large, in the 3 to 7 range. In 25 trials with no simulated errors, the average number of ones produced was the desired 2.6 per row, the maximum was 9 per row, and about 8 rows had no ones. Clearly the probes were randomly bunched so as to cause some clones which expected 2 or 3 hits to miss all probes and others expecting 6 or 7 to hit many more (in one trial a row hit 13 probes). Although not exactly matching the distribution of clone sizes in the LANL data, we considered it to be a good match.

However, the LANL data definitely contained some amount of chimerism and hybridization errors. In another project we are attempting to see whether we can use our algorithms to estimate the amounts of each error but for this study we merely tried several possibilities. Each time we increased the chimerism or false positive rate we had to reduce the average size of clone fragments in order to maintain the same distribution of number of ones per row. Similarly, we had to increase the fragment size when we increased the false negative rate.

In all we looked at eight cases: no errors, 10% and 30% chimerism only, 10% and 30% false negatives only, .2% and 1% false positives only, and a combination of 10% chimerism, 10%

false negatives, and .2% false positives. In order to be able to explore many trials in each of these cases we scaled the number of probes down to 100.

5.3 Exploring a range of data

The LANL data is only one example of what data might look like. We wanted to be able to discuss a wide range of data so that we would cover other existing data and also be able to suggest directions in which to push the data in order to achieve better results.
In this first study we chose to investigate each error type in isolation. We also did not want to have too many clones which gave no information so we slightly increased the average clone size to an average of five probe hybridizations per clone. We tried error rates of 10,20,30,40, and 50% chimerism of 1,2,4,8,16, and 32% false negatives and of 1,2,4,8, and 16% false positive. These rates were intended to include values at which all algorithms would be expected to do well and at which all would be expected to fail. In retrospect the false positive range should have included some much smaller values.
In order to allow a large number of trials we performed all our experiments using 50 probes. We were quite interested in the effect of coverage on algorithmic success so we tried each case with 50, 100, and 150 clones as well as some cases with 75, 125, 250, and 500 clones.

6 The Quality of Data

In Sections 3 and 4 we saw that the amount of information available in a hybridization matrix can be limited and discussed some of the reasons for its limitations. In the error-free case we were able to link the information available to the theory of PQ-trees but when the data contains errors we merely could say that things will be worse. Thus, as our first experiments we measured, as best we could, the amount of information available. We looked at the number of connected components in the clone cover graph, at the number of redundant probes, at the number of non-hybridizing probes, and at the number of weak adjacencies.

6.1 Connected Components

For any hybridization matrix we have defined the clone cover graph. Recall that the clone cover graph is formed by associating a vertex with each probe (i.e. column of the matrix) and placing an edge between two vertices for each clone which hits both probes (i.e. a row with a one in the two corresponding columns). Multiple edges between vertices are collapsed into a single edge with weight equal to the multiplicity. One can compute the connected components, that is, sets of vertices which are connected by some path of non-zero weight edges in the graph.
In a very strong sense, there is no information available about the relation of probes in different components. Consider two orderings of the probes in which the probes within components are kept in the same order while the order of entire components differs. There is no way to prefer one such order from another. Thus if there are many components then *any* algorithm will fail to reliably retrieve the correct complete order of the probes (an

algorithm may occasionally guess the order correctly but it cannot be sure of getting it correct.)
There are many ways in which multiple components can occur. One simple way is if a probe hits no clones. It is then in a component of its own. In hindsight, these probes should have been eliminated from the matrix but it is still instructive to see how often they occur. In the 25 trials of matrices constructed to match the LANL data, if no errors occurred there were between 0 and 7 probes which hit no clones with an average of 2.56 out of 100. Thus these types of probes are common enough to cause some trouble but not overwhelming.
When matrices were created assuming different types of errors, the number of such probes went up slightly for false negatives, down slightly for chimera, and reduced to essentially 0 with even 2% false positives. The interesting point to note is that although false positives can greatly reduce the apparent number of nonhybridizing probes, the number of probes which actually hit no clones has not been reduced. See Figures 2 and 3.

Lesson 1 *If there is a significant amount of false positives then it will be difficult to screen out non-hybridizing probes.*

Of course the number of connected components can be increased by other means besides non-hybridizing probes. If there is a large gap between probes then it is possible that no clone bridges the gap. For the LANL-like data there were between 9 and 17 components not including the non-hybridizing probes with an average of 14.72 when no errors were simulated. However, even 2% false positives reduces the number of components to 1. Simulating chimerism also greatly reduced the number of components, while simulating false negatives increased the number of components. For the moderate mixture of errors case, the number of components was reduced to 4.8. See Figures 4 and 5. Again note that the number of actual components was not reduced, only the number of apparent components.

Lesson 2 *If there is a significant amount of chimera or false positives then it will be difficult to identify connected components.*

6.2 Redundant probes

There is another simple way in which matrices can lack ordering information – two probes can hybridize identical sets of clones. If more than one probe hybridizes exactly the same set of clones than there is no information about the relative order of these probes. The same matrix would result from any two total orders which differed only in the order of these probes.
Redundant probes occur naturally whenever there is no clone which starts or ends in the interval between two successive probes. In real data this may be due to the fact that two probes are actually very close together on the target DNA or by chance if there just are not very many clones. Both these situations are faithfully modeled by our generator.
In some sense the problem of redundant probes is less severe than that of multiple components because all of the equivalent orders are quite similar. In fact, each equivalent probe order will lead to the same clone order. However, in evaluating the success of an algorithm it is important to know whether incorrect ordering is due to equal probes or not.

Error amounts	Min trial	Max Trial	Avg of Trials
none	1	7	2.9
10% chimerism	0	8	2.5
30% chimerism	1	4	2.6
10% false neg	1	6	3.1
30% false neg	1	7	3.3
.2% false pos	1	7	2.3
1% false pos	0	2	1.1
10% chi, 10% fn, .2% fp	0	5	2.3

Figure 2: Non-hybridizing probes for LANL-like data

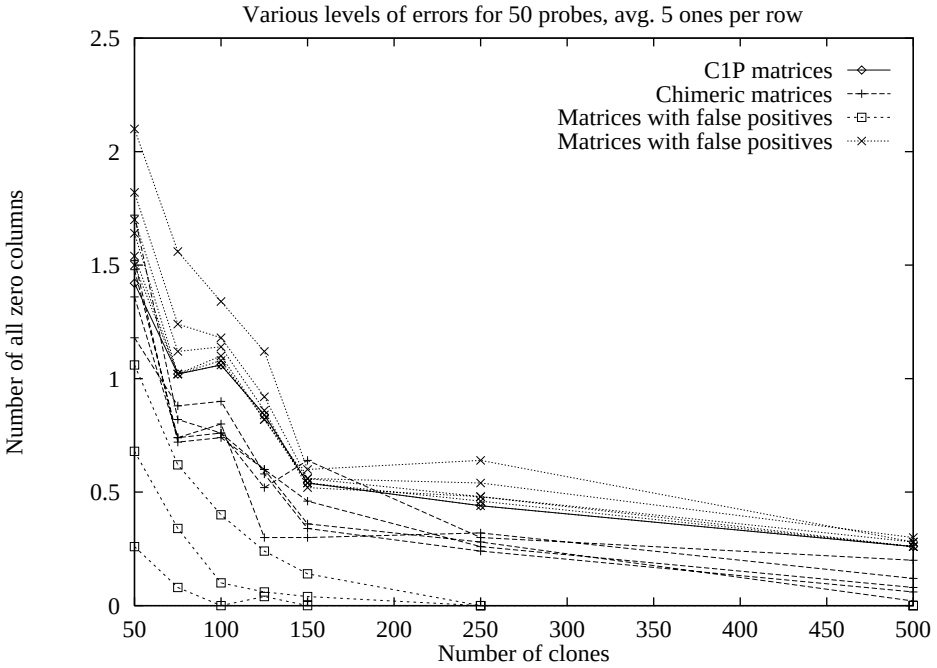

Figure 3: Non-hybridizing probes for varied data

Error amounts	Min trial	Max Trial	Avg of Trials
none	12	22	18
10% chimerism	5	19	10.5
30% chimerism	2	10	6.5
10% false neg	12	21	15.9
30% false neg	9	15	12
.2% false pos	2	9	6.2
1% false pos	2	5	3
10% chi, 10% fn, .2% fp	2	8	4.8

Figure 4: Connected components for LANL-like data

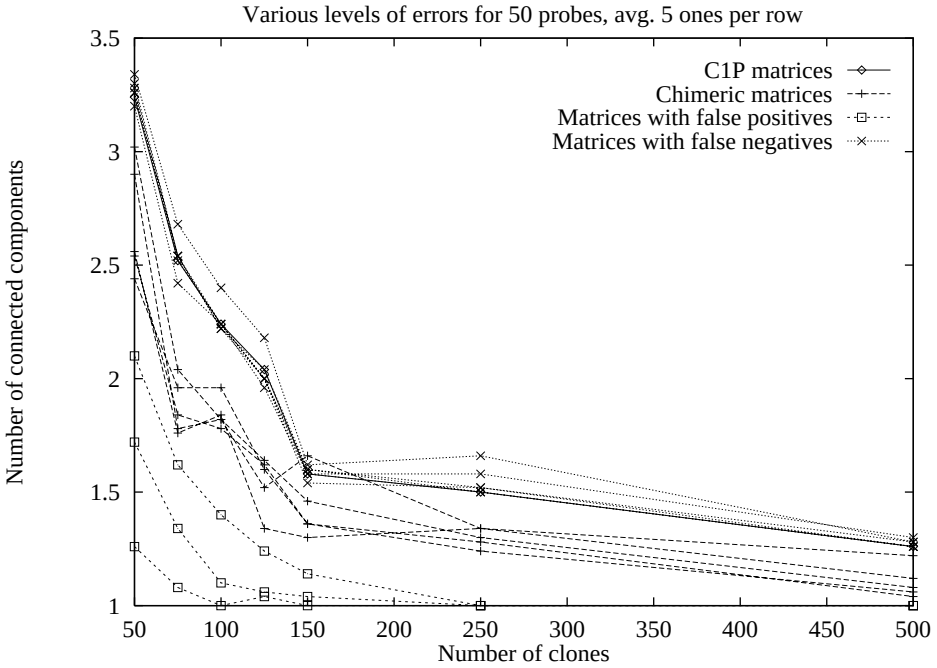

Figure 5: Number of connected components for varied data

It is, of course, relatively easy to screen input matrices for redundant columns and remove all but one of each type. However, as in the case of multiple components the presence of errors may obscure the fact that in the error-free matrix there are redundant columns.
In the LANL-like data when no errors are assumed, there are an average of 24 (out of 99 possible) probes which are identical to the probe before it in the ordering. (We used the fact that we know the correct ordering to simplify the search for equal probes but the columns could be checked for duplicates without knowing the order. In fact, our check of adjacent probes only may undercount the number of duplicate columns.) See figures 6 and 7.
As with multiple components the presence of errors tended to screen the occurrence of redundant probes. Chimerism only slightly decreased the number of apparent redundancies (presumably because there are more fragments with a chance to begin or end between probes.) Hybridization errors, both false positives and false negatives, had a much greater effect since even two probes which map to the exact same position on the target genome could appear different due to a hybridization error.

Lesson 3 *If there are hybridization errors (or to a lesser extent chimera) then it will be difficult to identify redundant probes.*

6.3 Weak adjacencies

The existence of multiple connected components and of redundant probes result in information theoretic constraints on the ability of an algorithm to solve the exact version of the mapping problem, i.e. reconstruct the exact sequence of probes along the target DNA. We believe that there are additional sources of weakness in the data. For example, the presence of masked components and redundancies mentioned in the earlier sections are examples of weaknesses not witnessed directly by multiple components or redundant probes.
As mentioned in Section 3, for C1P matrices there exist multiple different orderings which are C1P. Although some of the possible orderings are due to rearranging connected components and/or scrambling redundant columns there are other reorderings which correspond to "flipping" non-redundant groups of columns. More formally, any C1P orderings can be transformed into any other C1P ordering by a series of *translocations* in which each intermediate ordering is also C1P. A translocation corresponds to taking a section of the probe order and reversing it in place. The translocation $(i, j), 0 \leq i < j < n$ converts the n element permutation π to the permutation $\pi_0, \ldots, \pi_{i-1}, \pi_j, \pi_{j-1}, \ldots, \pi_i, \pi_{j+1}, \ldots, \pi_{n-1}$.
Using a PQ-tree representation it is thus possible to identify sequences of probes which are a subsequence of *all* C1P orderings of a matrix. As in Section 3 we refer to the adjacent probe pairs within these sequences as strong adjacencies. In the correct ordering of a C1P hybridization matrix, only these pairs are completely defined by the information in the matrix. Any other adjacencies in the correct order are *weak*, that is, it is not possible for an algorithm to always determine them. It should be noted that there may be partial information about these adjacencies. For example, if there are only two C1P orders (not counting reversing the entire sequence) which differ by a single flip then the two adjacencies on either side of the flip will be weak. However, an algorithm might be able to identify the two orders as the only possible orders.
In our LANL-like data without errors (thus with C1P matrices) on average *over half the adjacencies were weak.* Since the chance of identifying a weak adjacency correctly based on

Error amounts	Min trial	Max Trial	Avg of Trials
none	16	30	23.7
10% chimerism	16	29	23.6
30% chimerism	11	25	19.8
10% false neg	9	19	13.6
30% false neg	3	7	5.4
.2% false pos	10	22	14.8
1% false pos	1	3	1.7
10% chi, 10% fn, .2% fp	5	12	8.2

Figure 6: Number of redundant probes for LANL-like data

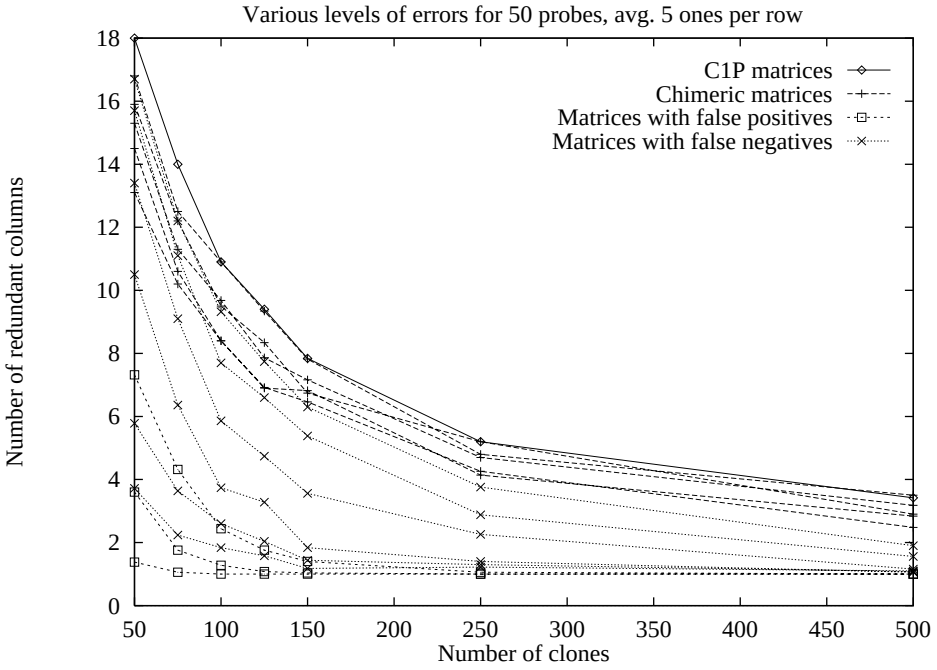

Figure 7: Number of redundant probes for varied data

the hybridization matrix alone is at most 50% *any* algorithm is expected to miss identify at least 25% of the correct adjacencies.

Lesson 4 *If there are weak adjacencies then judging an algorithm on whether or not it retrieves the entire correct ordering is futile.*

While it is possible to exactly define a weak adjacency for a C1P matrix (any adjacency which is at one end of a flip which preserves C1P-ness) it is less clear how to define weak adjacencies for nonC1P matrices. We are exploring the following definition but believe that it will require refinement as we learn more about these matrices. The intuition is that if the permutation after a translocation has no more fragments than the original permutation then most noise models would allow an explanation which was no more costly than the explanation of the original permutation. Although this assumption cannot be rigorously proven it does seem reasonable and the data seems to confirm its usefulness.

Definition 19 *Given an n column hybridization matrix, the adjacency $i, i+1$ for $0 \leq i < n-1$ is* weak *iff there exists a translocation $(i+1, j)$ or (j, i) such that for every row of the matrix the number of blocks of ones after the translocation is no greater than the number before the translocation.*

This definition has two clear weaknesses. First, it is possible that despite the fact that the number of fragments in each row has not increased, the noise model is such that the cost of the explanation has increased greatly. In this case, an adjacency we are considering to be weak should be considered strong – it is occurs in all close to minimal maps. Our current understanding of noise models makes this case seem unlikely but it is nonetheless possible. The second weaknesses is the opposite in nature, an adjacency which is weak may be identified as strong. It is possible, for example, that there exists a permutation which breaks an adjacency but the number of fragments in one row goes up while the number in all other rows goes down. The maximum number of fragments per row could even go down. Yet our simple procedure would not discover this fact. Thus even if the noise model has a cost function which is monotonic in number of fragments we may miss weak adjacencies. The chance of missing weak adjacencies is, however, a conservative property in our experiments. We will expect algorithms to correctly identify strong adjacencies so our algorithms will be penalized if we miss weak adjacencies.
Despite the above caveats about the notion of weak adjacencies, we have found it to be a useful concept. One interesting aspect of our definition of weak adjacencies is that unlike the number of multiple components and number of redundant probes it does not seem to correlate with error rate. See figures 8 and 9. Chimerism and false negatives seem to have little effect and false positives sometimes increase and sometimes decrease the number. Ideally, of course, none of the errors would effect it at all and thus we continue to look for better definitions.

6.4 Summary of data quality

As can be seen in Figures 3, 5, 7, 9, and 10, increasing the number of clones is an effective means of improving the data quality although it, of course, is experimentally expensive.

Error amounts	Min trial	Max Trial	Avg of Trials
none	41	63	52.3
10% chimerism	43	65	51.5
30% chimerism	45	59	53.2
10% false neg	40	60	50
30% false neg	31	61	48
.2% false pos	41	62	53.4
1% false pos	64	73	68.6
10% chi, 10% fn, .2% fp	45	60	54

Figure 8: Number of weak adjacencies for LANL-like data

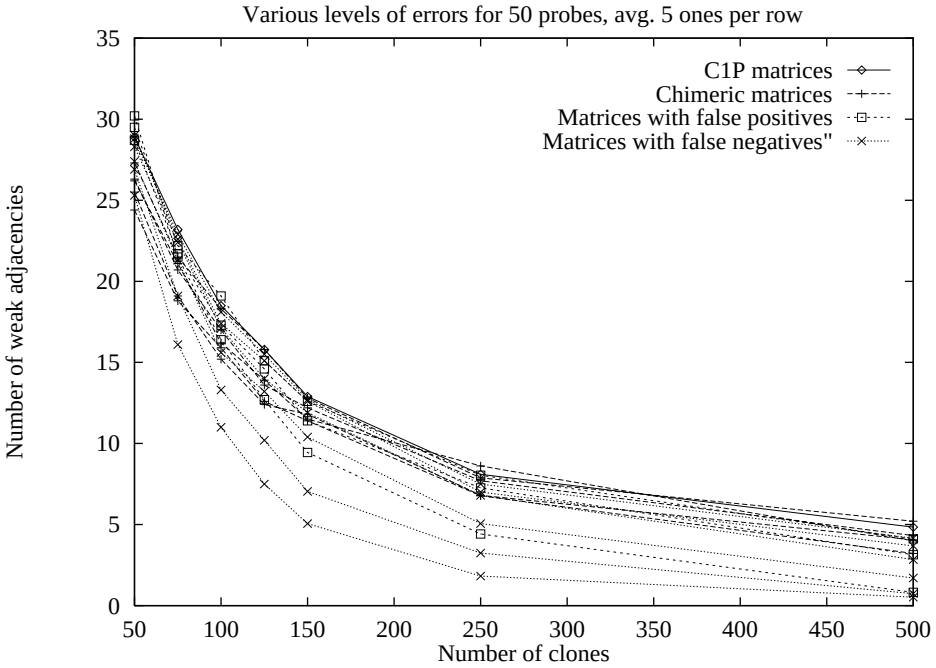

Figure 9: Weak adjacencies for varied data

One of our hopes is that by having algorithms which are effective on the strong parts of the data and which can identify the weak sections that effort can be placed into just improving the coverage on weak sections.

7 Combinatorial Optimizations as Models of Error-correction

7.1 Genomic reconstruction through parsimonious explanation

In the previous sections we have defined the physical mapping problem in terms of two processes: permutation (i.e. probe reorder) and noise removal (i.e. error repair.) One could, however, consider the process to consist of a single noise removal step where disorder is just another type of noise. Suppose there existed some objective function (which could be applied to ordered matrices) such that the function had value zero for the true map and some value greater than zero for any other order of the hybridization matrix. In this case, one could attempt to find the ordering of the matrix which minimized the function.
Unfortunately such a measure can exist only if one knows the genomic placement *a priori*. As we have seen, it is often the case that several orderings of the matrix could be the correct order depending on the actual genomic placement and the errors that occurred. The function could only take on its unique minimum value if it magically knew the genomic placement and the errors which occurred. However, it is possible that such a function can be approximated. In this section we discuss several functions which can be argued to approximate the magic measure and in Section 8 describe algorithms which attempt to minimize these functions.
Since the nature of the errors is poorly understood and they are stochastic in nature we might not even be able to recognize the magic function if we found it. We can, however, choose a candidate function based on some sound principles and then attempt to validate the function through empirical studies. As many errors occur at one time, measures that correlate with one or many errors are of interest. In the final analysis, however, one would hope to have measures that correlate with the structure of the errors as a whole.
Given such a measure μ defined for maps one can search for permutations π such the the value $\mu(A^\pi)$ is minimal, or close to minimal. Insisting on exact solutions is not justified biologically due to the type of data and its evolutionary nature. Moreover, the focus on approximations is mandatory: it is imposed by the computational intractability of computing exact solutions. The insistence on the relevance of "close-to-optimal" reflects the hypothesis of *parsimonious explanations*. Indeed, it is natural to search for minimal explanations, i.e., those based on the smallest amount of change.

7.2 Fragment-counting objective functions

Let us consider first measures based on fragment counting. For every permutation π the map A^π represents a fragmentation of the clones into fragments based on the blocks of consecutive ones in each row. One can measure various parameters of this fragmentation. Let $\sigma(A^\pi)$ be the total number of fragments. Let $\chi(A^\pi)$ be the maximum number of fragments in a clone. Let $\beta(A^\pi)$ be the number of clones that are "broken", i.e., split. Clearly all these functions have two nice properties. First, they have minimal value when the matrix is C1P.

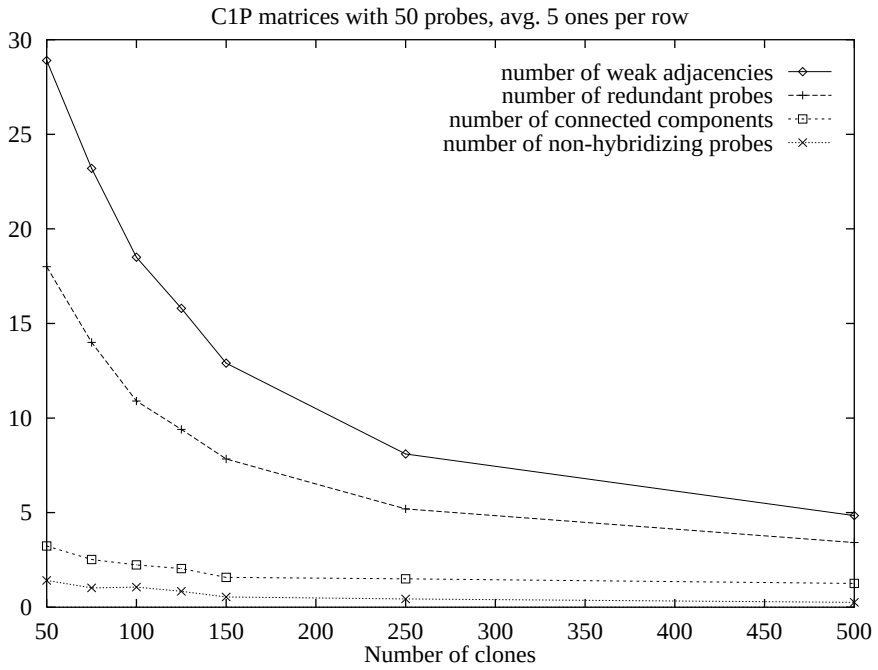

Figure 10: Symptoms of weak data for C1P matrices

Secondly, they are non-decreasing as the clone splitting increases (chimerism rate increases). Deletions, false positives, and false negatives can either increase or decrease the functions but will typically increase the measure. Decreases come only in the special cases when deletions or false negatives completely remove a fragment or when false positives join two fragments. (When the error rates get very high, eg. above .5, these special cases become more common. However, in these cases reconstruction will be extremely difficult by any means.) Intuitively, maps that have values close to minimal in these measures provide good approximations of parsimonious explanations.
Information known about genomic libraries supports the use of these objective functions. For example, it was observed that chimeric clones rarely have more than two inserts. Therefore, maps having a very small value of χ would be consistent with the observed data. If the rate of chimerism is known, maps with the β measure close to the chimeric rate, or the σ measure close to the number of clones plus the number of chimeric clones would provide a good fit for the extra knowledge about the library.
One disadvantage of these fragment-counting functions is that finding the optimum value on general 0/1 matrices (and on some sparse, hybridization-like matrices) has been shown to be NP-complete[2, 16]. However, NP-completeness for optimum need not be a barrier to us since we already know that we must look for properties of all approximate solutions rather than for a single optimum solution.

7.3 Other objective functions

7.3.1 Total inter-fragment gap length

One of the limitations of the measures based on fragment counting is the fact that they are insensitive to the size of the fragments or the size of the gaps between fragments. The distinction between the size of gaps is essential for distinguishing between chimerism and deletions. It is likely that chimeric clones have big gaps between their fragments, while deletions introduce small gaps in the clones. In this respect, false negatives and deletions could be treated together, and data sets exist where they are the dominant errors. The natural idea of keeping the gaps small led us to consider a measure γ which is the total inter-fragment gap length. Let us start first by defining the measure α to be sum over all rows of the number of columns between the first and the last 1 in the row. Also let δ be the density of the matrix, that is the total number of ones in the matrix. Now we define γ by $\gamma(A^\pi) = \alpha(A^\pi) - \delta(A^\pi)$.
Minimizing γ or equivalently α attempts to keep the gaps as small as possible. The function α resembles an objective function considered in numerical analysis for square matrices in connection with the storage of sparse matrices. Although our matrices are not square, the adjacency matrix of the clone cover graph is a closely related square matrix that can be used for the connection.
Minimizing γ obviously is inappropriate when large gaps are present. Therefore, in that case, dealing with chimerism takes algorithmic precedence.

7.4 Objective functions

There are two properties that it would be desirable for all the objective functions to satisfy.

1. **Conservative Extension of C1P** The optimization function takes on its best (typically minimal) value for C1P matrices.

2. **Monotonicity** The optimization function maps matrices corresponding to higher error rates to worse (typically higher) values.

Each of σ, χ, β, and α is indeed a conservative extension of C1P. Thus when these functions are applied to error-free data the optimal solutions will be C1P. As discussed above, each of σ, χ, β, and α is likely to be monotonic. The ways in which these functions break monotonicity for hybridization errors is a potential area for continued research. Do there exist simple functions which are monotonic for hybridization errors? Are they efficiently approximable?

8 Algorithms

8.1 The Huffman-greedy algorithm: Minimizing σ

8.1.1 Converting to TSP on the Hamming vector graph

The connection between minimizing the number of blocks of ones in a matrix and solving the Traveling Salesperson Problem[15] on the "hamming distance graph" of the matrix has been observed by many authors[1, 24]. In [18] we abstracted this connection to what we called the vector Traveling Salesperson Problem (vTSP). We repeat the definition of vTSP here for convenience.

Definition 20 [18] *An instance of the* vector-TSP *is an n vertex, vector-labeled, complete, undirected graph, $G = (V, E, cost_v)$ and a function $f : N^m \rightarrow N$ (where $cost_v$ is a function from edges to m-vectors over $\{0, 1\}$.)*
The sparseness *of a vTSP graph, $s(G) = \max_{e \in E}$(the number of ones in the vector cost_v(e)).*
The vector-cost *of a tour in G is the component-wise sum of the costs of the edges in the tour. The f-*cost *of a tour is f applied to the vector-cost.*
The vTSP *f*-optimization problem *takes as input a vTSP instance $I = (G, f)$ and returns a tour in G of minimal f-cost.*

The conversion from the problem of minimizing σ on a matrix to an instance of vTSP is as follows:

- Create a vertex corresponding to each column of the matrix.
- Label each edge (v_1, v_2) with the vector formed by taking the componentwise *exclusive or* of the columns corresponding to v_1 and v_2. The resulting vector has a one in each entry corresponding to a row in which the two columns differ and a zero in all other entries.
- Let reduction function, f, be the function which sums the entries of a vector. Thus, f applied to an edge gives the number of rows in which the two columns differ and f applied to the vector-cost of a tour gives the number of times a block of ones began or ended in the tour.

The tour found by minimizing the vTSP instance allows blocks of ones to wrap around the matrix (that is, start near the right side of the matrix and continue at the left side). To avoid this technical problem an additional column of all 0's can be added to the matrix and the tour opened up to a path from one neighbor of this column to the other. With this change the cost of the tour is exactly twice σ.
Because addition is distributive it is easy to see that using vector sum as the reduction function makes the vTSP equivalent to the standard scalar TSP. Therefore algorithms to optimize σ can be based on the classic approximation algorithms for TSP, "twice around a minimum spanning tree" and the Christofides' maximum matching improvement[15]. These algorithms have the advantage of giving provable guarantees on performance. For example, using Christofides' algorithm one is guaranteed that the ordering of the matrix produced has a value of σ which is no worse than $\frac{3}{2}$ times the optimal value of σ for any ordering (the simpler algorithm yields a guarantee of 2 times optimal.)
However, preliminary experiments (and the experience of other researchers on other TSP instances) show that these algorithms do not do much better than their guarantees and that other algorithms give better solutions in practice. Therefore we coded two variants of a greedy algorithm for minimizing TSP.
Similar algorithms were used extensively by [1] with the addition of a 2-OPT phase at the end.

Greedy sigma I Our first algorithm works as follows.

- Convert the matrix to a TSP graph.
- Starting with the first vertex add the closest vertex not already on the tour to the tour. Ties are broken randomly.
- Continue adding the closest (to the last vertex added) vertex which is not already on the tour until all vertices are used. Again, ties are broken randomly.
- Convert the tour to a matrix order by choosing the starting position which give the best value of σ. (A tour corresponds to n possible matrix orders depending on which vertex is made the left hand column.)

Although the reliance on randomness seems a disadvantage since the quality of the tour can depend highly on the random choices we found that by using multiple random orders that we could get both improved orders and information about the reliability of parts of the order.

Greedy sigma II, Huffman As a check against the dependence on randomness in the first algorithm we implemented a second algorithm patterned after the algorithm for creating Huffman codes. Although this algorithm still uses a randomness to break ties it is much less dependent on a start vertex. This algorithm works as follows.

- Convert the matrix to a TSP graph.

- Pick the nearest two vertices and make them adjacent on the tour. Ties are broken randomly.
- Continue choosing adjacent pairs of vertices which are not yet adjacent to two vertices and make them adjacent on the tour until all vertices have two neighbors on the tour.
- Convert the tour to a matrix order by choosing the starting position which give the best value of σ. (A tour corresponds to n possible matrices depending on which vertex is made the left hand column.)

8.2 The clone-cover algorithm

Although the reduction of σ optimization to TSP preserves the optimum value it is not clear that an approximation algorithm for TSP will fairly explore the region of near optimal solutions. In particular, the use of hamming distance produces a penalty for two adjacent columns having different values in a row but only indirectly rewards two adjacent columns which both have 1's in a row. We therefore created an algorithm which explicitly attempts to keep columns together which both have 1's in the same rows.
Rather than creating the hamming distance graph we create the clone cover graph described in Section 3. Similar ideas appear implicitly in D.R. Fulkerson and O.A. Gross[14] and Lander and Waterman[25].
Heuristically, it appears that one can minimize σ by maximizing the cost of a tour in the clone cover graph. If edges on the tour have high values then many 1's are placed adjacent to each other and there will be fewer overall blocks of ones. Maximizing the cost of a tour is, unfortunately, at least as difficult as minimizing the tour and therefore we once again resort to greedy heuristics.
Our third algorithm thus works as follows.

- Convert the matrix to a clone cover graph.
- Starting with a random vertex, add the vertex whose weight to this vertex is greatest and which is not already on the tour to the tour. Ties are broken randomly.
- Continue adding the vertex, which is not already on the tour, whose weight to the last vertex added is greatest, until all vertices are used. Ties are broken randomly.
- Convert the tour to a matrix order by choosing the starting position which give the best value of σ.

8.3 The Fiedler-vector-spectral algorithm: Minimizing α

Concentrating on the *number* of blocks of ones in the matrix by optimizing σ does not capture the fact that two blocks separated by a single 0 could occur in experimental matrices due to several factors (eg. a false negative hybridization result or a deletion) which are highly unlikely to create two blocks separated by many 0s. Therefore we looked at methods which attempt to group all the 1's in each row together but don't demand that they form a single block. Formally, a row is scored by the number of columns from the first occurring 1 to the last occurring 1. The row 00010101010000000000 thus gets a score of 7 under the α

metric (compared to a best value of 4 and worst value of n) whereas it would get a score of 4 under the σ metric (compared to a best value of 1 and worst value of 4).
The α measure was inspired by the sparseness of the hybridization matrices. It seemed reasonable to attempt to keep the area containing ones as small as possible. Keeping this area small tends to minimize the number of false negatives and deletions in an unified way. For symmetric matrices, α is the so called "envelope" used in numerical analysis for handling sparse matrices. Minimizing the envelope is seen as a way to minimize the storage for sparse matrices which in turn speeds up computation. Although envelope minimization is NP-hard, various heuristic methods have been developed that often work well on numerical analysis applications.
Unfortunately, hybridization matrices are not symmetric, so standard envelope minimization techniques cannot be directly applied. We can, however, form a symmetric matrix my multiplying the hybridization matrix by its transpose. The resulting matrix has a nonzero structure which corresponds precisely to the edges in the clone cover graph.

8.3.1 Spectral methods

When the connection of with the "envelope" was made, a new algorithmic avenue came to light, namely the use of spectral methods.
Spectral methods are fundamental in numerical analysis. Fiedler[11, 12] developed the mathematics that was ultimately used to design powerful heuristics for arranging sparse matrices in such a way that all the entries are as close to the diagonal as possible. The first algorithmic uses of Fiedler's ideas for ordering problems were due to Juvan and Mohar[22]. More recent applications include work by Pothen and Simon [29] and Atkins, Boman and Hendrickson [3].
We employed a heuristic for optimizing α which is based on the work of our colleagues at Sandia who have developed the code for an independent project called Chaco[21] which partitions computational meshes for placement on parallel machines. When it became apparent that their code might be useful to us, they very kindly made modifications to allow its inclusion in our software suite. In fact, they became interested in the theoretical properties of spectral methods for this application and have proved several lovely theorems about it[3].
A full description of their code is beyond the scope of this paper and can be found in [21, 3] but we give here a sketch of its use for mapping.

The Fiedler-vector spectral algorithm

- Given an $m \times n$ matrix A construct the clone cover graph $CC(A)$. Let G_A be the adjacency matrix of the graph $CC(A)$.
- Construct the Laplacian L_A of G_A, i.e., $L_A = D_A - G_A$ where $D_A = (d_{ii})$ is a diagonal matrix with $d_{ii} = \sum_{j=1}^{n} A(i,j)$ and for $i \neq j, d_{ij} = 0$
- Let v_F be the n-dimensional Fiedler vector of L_A defined to be the eigenvector of L_A corresponding to the second smallest eigenvalue of L_A. v_F has only real entries.

- Sort the components of v_F. The sorting produces a permutation π_F of the the set $\{1, 2, ..., n\}$.
- Output the map A^{π_F}.

In recent work Atkins, Boman and Hendrickson [3] showed that a generalization of this algorithm has the remarkable property of solving the consecutive ones problem. That is, on C1P matrices, the ABH-algorithm finds the C1P permutations. The relevance of this beautiful theorem in the biological context is as follows. This algorithm offers an extra guarantee: it is *provably correct* on error-free instances. As the error-free set of inputs is quite complex, this is a highly non-trivial result and the first of this kind, to our knowledge.

8.4 The cycle-basis algorithm: Minimizing χ

Both our algorithms for optimizing σ and those for optimizing α consider measures which are the sum over all the rows of the matrix. For large matrices this means that one or two very badly ordered rows can be masked by lots of well ordered rows. In some cases this may be inevitable, but since we expect really bad clones to be filtered from the input we wanted to be able to look at measures which attempted to optimize the *maximum* behavior of any row rather than the *average* or sum behavior. The measure χ does exactly this.

8.4.1 Preliminaries

We start by considering a class of fragment counting objective functions for which the problem of finding optimal tours could be approximately reduced to the problem of computing spanning trees, both problems in vector-TSP graphs. Let us say that a function $f : N^m \to N$ is *monotone* if for every $a_1 \leq a_2 \in N^m, a_1 \leq a_2$ we have $f(a_1) \leq f(a_2)$. We say that f is *subadditive* if for every $a_1, a_2 \in N^m$ we have $f(a_1 + a_2) \leq f(a_1) + f(a_2)$.

Lemma 6 *Let A be an $m \times n$ matrix and $f : N^m \to N$ a monotone and subadditive function. Then the optimal f-cost of a tour of the vector-TSP instance (G_A, f) is at most twice the optimal f-cost of a spanning tree of G_A.*

Proof: The proof is a straightforward extension of the proof of the twice-around-a-tree heuristic for TSP[15]. ∎

As χ is monotone and subadditive we will concentrate now on computing an optimal f-cost spanning tree. Again, the problem turns out to be NP-complete.

8.4.2 The algorithm

We present an algorithm for computing a spanning tree whose χ-cost is close to optimal. The algorithm is based on local search. Informally, one starts with a spanning tree T and searches for better spanning trees in the neighborhood of T. If one such tree is found, one continues searching for improvements in its neighborhood and so on. When a tree T_0 is found such that no better trees exists in its neighborhood then T_0 is a locally optimal tree. We can show that T_0 is close to the globally optimal spanning tree T_{OPT}. To complete the informal presentation of the algorithm we have to describe the neighborhood of a spanning

tree and the criteria for choosing a better spanning tree. The neighborhood is the so called *cycle-basis*. For a spanning tree T and for every non-tree edge e one could obtain a graph with exactly one cycle by adding the edge e to the T. Then one could remove edges of T from the cycle. We collect only the resulting spanning trees (some edge removal might disconnect). When we do this operation for all edges not in T we obtain a collection of spanning trees that is the cycle basis neighborhood of T. As the χ value for spanning tree "is slow in reporting changes" we use as a more change-sensitive criteria for choosing better trees: The lexicographic order of the sorted vector-cost of the tree.

The cycle-basis algorithm

1. Given an $m \times n$ matrix A, construct G_A the n-node $vTSP$ graph of A. The nodes of G_A correspond to the columns of A, or to the probes. G_A is a complete graph with the edges labeled by m-dimensional vectors with 0/1 entries. If edge $(j1, j2)$ corresponds to columns C_{j1}, C_{j2} then its label is the the Hamming vector of C_{j1} and C_{j2}.
2. Construct a spanning tree T of G_A.
3. Repeat until no improvement
   - Add to the current tree T a non-tree edge
   - If an edge of T from the unique cycle formed by the added edge can be removed such that the resulting graph is a tree T', and T' has a better χ-value than T, then we have improvement and we set $T = T'$
4. Let T_0 be the resulting spanning tree.
5. Let π_0 be the permutation obtained by opening anywhere the tour obtained by the standard "walking twice-around" the tree T.
6. Output π_0

The performance of the algorithm is given next.

Theorem 4 *Given an $m \times n$ matrix A the cycle-basis algorithm outputs a permutation π_0 such that*

$$\chi(A^{\pi_0}) = O(\chi(A^{\pi_{OPT}}) + \log m)$$

where π_{OPT} is the χ-optimal permutation for A. The algorithm has worst-case running time $O(n^3 \log m)$.

8.5 The 2-OPT algorithm

It is common when using approximation algorithms for NP-hard combinatorial optimizations to use some sort of local search to fine tune the solution. One common local search method for permutation based algorithms is the 2-OPT algorithm. When performing a 2-OPT two permutations are consider to be neighbors if they differ by a single translocation. As we saw in Sections 3 and 6, the translocation is a natural operation for our algorithms.

Formally, the 2-OPT algorithm takes as input the $m \times n$ matrix A, and a permutation $\pi = (i_1, ..., i_n)$ and an optimization function f. Recall that a translocation "flips" sections of the permutation. A *segment* of π is given by two elements i_j, i_k such that i_j occurs before i_k in π. A *translocation* of the segment (i_j, i_k) in π is a permutation π' obtained from π by reversing the sequence of elements of π between and including i_j and i_k. Given a permutation π there are $\frac{(n-1)*n}{2}$ translocations for a permutation of size n (this set includes the original permutation π).

The 2-OPT algorithm

1. Given an $m \times n$ matrix A and permutation $\pi \in Perm[n]$.
2. Until no improvement is possible
   - Consider in turn all the translocations of π and among them choose the permutation π' for which $f(A^{\pi'})$ is minimal.
   - If $f(A^{\pi'}) < f(A^{\pi})$ then there is improvement and we set $\pi = \pi'$

In our experiments we used as our optimization function the function σ.

It is interesting to note that unlike the cycle basis algorithm, the local search of 2-OPT yields no guarantees either on running time or solution quality. The local optimum can be arbitrarily worse than the global optimum and it can take exponential (in n) time to converge to a local optimum. For this reason it is desirable to use 2-OPT only after a "good" starting solution has been found. If the starting solution is provably close to the optimum solution and a bound is placed on the number of steps of 2-OPT to be performed then the *combination* of an algorithm with 2-OPT will yield a solution which is provably close to the optimum and will not take unbounded time.

9 Experimental Design

We have argued both theoretically and experimentally that there are severe limitations on the ability of an algorithm to retrieve, from a hybridization matrix alone, the correct genomic order of the probes and/or clones along the genome. Nonetheless we have designed several algorithms and put them to the test. Although many algorithms have been proposed and even implemented for variants of the physical mapping problem (see Section 11 for more on related work) there has been little or no published attempts to give a rigorous evaluation of these algorithms across many situations. Typically, the algorithm is run on several specific large "real" data sets and the results are compared to the "best previous algorithm" or to "the published map of the data generators."

It is one of the goals of this paper to establish the beginning of a dialogue within the community concerning how to evaluate mapping algorithms. We have given some theoretical background but the success or failure of algorithms will always remain their practical use. We believe that understanding the practical use of an algorithm depends on answering the following questions:

1. What are some good *quantitative* measures of the goodness of a map?

2. How can fair comparisons of different algorithms be made? In particular what should the output of an algorithm be?

3. When an algorithm performs poorly, why did it perform poorly?

4. What are the requirements for a useful generator of synthetic examples?

5. How many random trials are necessary to produce reliable information about an algorithm?

6. What parameters are worth adjusting in studying an algorithm?

9.1 Measures of success

We do not yet have a totally satisfactory answer to the first two questions: how to measure the goodness of a map. We have, however, explored several measures. The first measure is a natural outgrowth of our theoretical discussion of the information available in a hybridization matrix. We ask how many of the adjacencies in the true order are identified by the algorithm. In retrospect, we now believe that we should have attempted to make our algorithms list only those adjacencies for which the algorithm has derived good evidence that the adjacency is in the true map. In this way the algorithm would be able to try to identify weak adjacencies as well as list strong adjacencies. However, in this study we created algorithms, which like most previous algorithms, attempted to produce a total order of the probes.
Thus, our first measure is *percent of true map adjacencies which are adjacent in the algorithm's order.*

Definition 21 *Given an algorithm $\mathcal{A}$ which produces an ordering π_A on an n probe hybridization matrix with true order π_0 the* total adjacency cost *is $\frac{1}{n-1}\sum_0^{n-2}\delta_i$ where $\delta_i = 1$ if $\pi_0(i)$ and $\pi_0(i+1)$ are not adjacent in π_A and $\delta_i = 0$ otherwise.*

We have argued that the algorithm should only be responsible for determining strong adjacencies. Thus our second measure counts only strong adjacencies which are missed. Of course, for data containing errors we must have some definition of a strong adjacency such as the one defined in Section 6.

Definition 22 *Given an algorithm $\mathcal{A}$ which produces an ordering π_A on a n probe hybridization matrix with true order π_0 the* strong adjacency cost *is $\frac{1}{n-1}\sum_0^{n-2}\delta_i$ where $\delta_i = 1$ if ($\pi_0(i), \pi_0(i+1)$ is a strong adjacency and $\pi_0(i)$ and $\pi_0(i+1)$ are not adjacent in π_A) and $\delta_i = 0$ otherwise.*

We are currently experimenting with some new measures. The number of adjacencies does not seem to completely capture how far from correct a solution is. For example, if two adjacent probes in the true order are flipped then two adjacencies will be incorrect. However, the adjacency cost will be the same if some large section of the genome is transposed in the solution. We therefore look at the average distance that adjacent probes from the true order are separated in the calculated order. A variant of this measure is mentioned in [8].

Definition 23 *Given an algorithm $\mathcal{A}$ which produces an ordering π_A on a n probe hybridization matrix with true order π_0 the* total distance cost *is*

$$\frac{1}{n-1}\sum_{0}^{n-2}|\pi_A^{-1}(\pi_0(i)) - \pi_A^{-1}(\pi_0(i+1))|.$$

Of course a variant of total distance can be defined in which only strong adjacencies are counted. In preliminary studies the total distance measure seems to give us a better understanding of the quality of an algorithm but we are not yet comfortable enough with our understanding of it to report results.

9.2 Parameters to vary

Perhaps the hardest part of an experimental study is deciding what to look at. Having settled on adjacency cost as our primary measure of success we still had to decide what parameters to vary and over what ranges.

It was clear that we wanted to vary the amount of chimerism, false negatives, and false positives. We chose, for this first experiment, to vary each independently so as to limit the number of possibilities and to make the results more clearly relatable to the input. We chose a range of 0 to 50% chimerism since the literature seemed to imply that a fair amount of chimerism was present in real data. We also expected from our earlier studies that algorithms could perform well even at these high levels of chimerism. Choosing a false negative rate was more difficult. Many pooling techniques are biased toward producing false negatives in order to reduce the amount of false positives. In fact, some groups believe false positives to be so costly that they apply pre-processing techniques to remove false positives from the matrices at the cost of increasing the number of false negatives. The fear of false positives was born out by our experience. Although we intended to use the same 1% to 32% range used for false negatives we found that some of algorithms took inordinant amounts of time to process more than 4% false positives and then gave very poor solutions.

It should be noted that the false positive and false negative rates are not equivalent measures since the false negatives can only occur at the few percent of matrix entries which record hybridizations while the false positives can occur anywhere else. In fact, we recommend that the false positive rate in future experiments be based on the expected number of false positives in a row. Thus if there are 50 probes and one wishes approximately one false positive per row then the rate must be 2%, whereas with 100 probes the rate must be 1%.

It was also clear that we wanted to vary the number of clones in the experiment. The term *coverage* has many definitions in the literature but always connotes the relative number of clones to probes. Defining coverage to be the relative number of clones to number of probes, we looked at coverages varying from 1 to 10. The coverage of 1 is almost certainly too low while the coverage of 10 is well beyond the level of effort likely to be made by any actual mapping team.

It was less clear what to do about the relative size of clones. Preliminary studies showed that minor variations did not seem to make much difference. However, we believe that the role of clone size deserves further study in the future. Even having decided to not explicitly vary clone size, we were left with a dilemma of how to account for the fact that differing amounts of errors led to differing apparent clone sizes even if generated clone size

was kept constant. When no effort was made to counterbalance the error effect we found that increased chimerism led to improved algorithm success. This counterintuitive result was due to the fact that the chimeric clones contained additional information (in the form of additional fragments). We thus decided to modify the input clone sizes so that the expected density of ones in the matrix was unchanged. This of course meant that with high chimera and false positive rates the size of clones decreased.
Each of the approaches, constant clone size or constant ones density, has is shortcomings. We arbitrarily settled on constant ones density but expect to report results on constant clone size in the future.

9.3 Timing

No experimental study would be complete without some measure of the time cost of the algorithms used. We ran all our experiments on a single dedicated Sparc 20 workstation so times should be equivalent. However, the effort put into optimizing the various algorithms varied greatly. The spectral algorithms were part of a separate project for graph bisection and have been carefully optimized. The Huffman and Clone Cover algorithms are relatively simple and, though little effort was put into optimization, are probably fairly efficient. The Cycle Basis Algorithm is quite complex and probably could be sped up considerably. The rudimentary 2-opt algorithm was coded in a naive manner and there are known to be considerably more efficient implementations. We hope to improve its efficiency considerably when we extend it to consider additional optimization criteria.
Given all these caveats the timing graphs which follow in the next section should be used mostly as a gauge of the amount of 2-opt necessary and as some indication of how the current implementations scale as the coverage and amount of error increases.

10 Experimental Analysis of Algorithms

10.1 General observations

All of the algorithms exhibit good trending behavior. That is, when more coverage is given the algorithms find more correct adjacencies. Furthermore, when the amount of error is increased the algorithms tend to do worse. One exception is that the use of 2-OPT seems to benefit from increased chimerism. We are not sure why this is so and continue to examine the detailed data in order to look for reasons.
We present our data in summary graphs. Each graph has cost (total adjacency cost, strong adjacency cost, or time) on the y axis. The x axis is more complicated. It represents both coverage and amount of error present. Each coverage is assigned an interval on the x axis and within that interval the amount of error is allowed to grow. Thus for chimerism the points at $x = 1, 1.5, 2, 2.5, 3, 5$, and 10 are for each amount of coverage with no chimerism and the points at $x = 1.25, 1.75, 2.25, 2.75, 3.25, 5.25$, and 10.25 are for each amount of coverage and 50% chimerism. Formally the value of the x axis is the coverage (i.e. # of clones / # of probes) plus $\frac{1}{200}$ times the percent chimerism.
Similarly for false negatives and false positives the value of the x axis is the coverage plus $\frac{1}{10}$ times the log base two of the percent false entries. No particular meaning is given to the

spacing of the error values – they are merely chosen to spread out the points.
For each algorithm there are nine graphs. The first three examine the effects of pure chimerism. The first graph looks at total adjacency cost and strong adjacency cost for the algorithm alone, while the second graph looks at the same costs for the solution of the algorithm followed by local search using 2-OPT. The third graph shows running time for the algorithm and running time for the 2-OPT portion alone. The next three graphs give analagous results for pure false positives while the last three give results for pure false negatives.
The algorithms examined are the spectral algorithm, which attempts to minimize the envelope of the hybridization matrix; the cycle basis algorithm, which attempts to minimize the maximum number of fragments in any row of the hybridization matrix; the huffman TSP algorithm, which attempts to minimize the total number of fragments in all rows of the hybridization matrix; the clone cover algorithm, which attempts to maximize the amount of adjacent overlap; and a random algorithm, which choses a random permutation.
For each experimental point (coverage and error rate) 50 random matrices were generated and all algorithms were run on the *same* matrices. Detailed data from the runs will be made available on the World Wide Web at the time of publication of this paper. The URL can be found through the home page at http://www.cs.sandia.gov/ dsgreen/main.html.

10.2 Spectral

Figures 11 through 19 show the results of the experiments using the spectral algorithm. It is interesting to note that increased coverage does not seem to significantly help the algorithm in the face of chimerism or false positives. In contrast, increased coverage does seem to allow the 2-OPT algorithm to find a better local optimum when applied to the spectral algorithm's solution. The fact that the spectral solutions with higher coverage do not appear better by the adjacency measure yet are better starting points for 2-OPT is evidence that the adjacency measure is not a perfect measure.
Increased coverage does improve the spectral algorithm's performance on false negatives. It is difficult to tell whether this effect is mostly due to the increased number of strong adjacencies or to increased information about all adjacencies. The spectral algorithm is so effective on false negatives that the 2-OPT algorithm can make little improvement, especially for rates less than 8%.

10.3 Cycle Basis Algorithm

Figures 20 through 28 show the results of the experiments using the Cycle Basis algorithm. Increased coverage seems to help the Cycle Basis algorithm most for correcting chimerism.

10.4 Clone Cover

Figures 29 through 37 show the results of the experiments using the clone cover algorithm. Increased coverage seemed to help the clone cover algorithm in all cases. We were surprised at the robustness of the clone cover algorithm since it doesn't explicitly optimize any parameter which we can directly relate to map goodness. However, the notion of strongly

linked probes is common in the biology literature and our experiments seem to justify its use.

10.5 Huffman

Figures 38 through 46 show the results of the experiments using the Huffman algorithm for greedy TSP minimization. As has been reported by us in a previous paper[18] and by other authors[1] the greedy minimization of the hamming distance TSP is quite effective. We do not give results for the Greedy Heuristic algorithm mentioned in Section 8 since the Huffman algorithm was marginally more effective and is less dependent on a good initial order. It was clear to us that it is very important to test greedy algorithms with a random initial ordering. Otherwise the greedy algorithms tend to conserve the input order and look artificially good when the data is presented in the correct order.

10.6 Random

Figures 38 through 46 show the results of the experiments using the 2-OPT algorithm on a random permutation. The results with just the random permutation are predictably bad. The results for the strong adjacencies can be used to track the number of strong adjacencies. When there are many weak adjacencies the random result can only be wrong on the remaining strong adjacencies. As the number of strong adjacencies increases there are more adjacencies on which the random result can fail.
The 2-OPT algorithm by itself is fairly effective. It, however, is very expensive. Even taking the relative efficiency of the implementations into account it is clearly a major benefit to give 2-OPT a good starting position. In addition, the dependency of 2-OPT on error rate is particularly bad. It seems that higher error rates yield more gentle landscapes thereby allowing 2-OPT to make many more changes before reaching a local minimum.

10.7 Comparison

In order to compare the various algorithms it is really necessary to examine each matrix in turn. We are building the machinary with which to do this but for now must settle on some summary data. In the two tables below we show the total and strong adjacency cost for each algorithm on a few extreme examples: coverage = 1 or 10 and error rate = min or max.

In both Figure 56 and 57 a star has been placed next to the minimum value for each case. Atkins *et al.* [3] have recently modified the spectral algorithm to always find a C1P ordering if one exists, so it is not surprising that even their early spectral algorithm gives the best results for the C1P case. It is apparent that there are many weak adjacencies in the coverage = 1 case and many fewer in the coverage = 10 case. Spectral is also very effective for false negatives. Although the total adjacency measures are not particularly encouraging the strong adjacency numbers (and the graphs above) show that for moderate levels of false negatives most of the information inherent in the matrix can be retrieved.

For chimerism there are several effective algorithms with both Huffman and spectral yielding slightly better starting points for 2-OPT. The fact that all algorithms miss less than 5% of the strong adjacencies even for 50% chimerism and coverage of 1 is quite encouraging. One might hope that with a combination of algorithms reliable information about the strong adjacencies can be retrieved.

The case for false positives is less rosy. Clone cover is surprisingly effective but it is disturbing that, even at the moderate levels of false positive tested, more coverage was actually a detriment toward finding strong adjacencies. This appears to be the result of there being more strong positives but in some sense the new strong positives are less strong and therefore harder to recover.

11 Related work

The literature contains extensive studies on physical mapping and a variety of algorithmic approaches have been proposed. Various software packages are available that offer a quite variate source of computational support for mapping. It is not our intent in this work to compare and contrast various approaches but merely to provide a framework in which this can be done in the future. Each of the efforts described below has had its own goals and measures of success.

The mapping software developed by the H. Lehrach's group [20, 27] contains a variety of algorithms including ones dealing specifically with chimerism (a rarity in the literature.) They make use of the TSP analogy and apply simulated annealing. As an output from their program they look for subsets of the clones which span the probes. Thus they are able to ignore some ill-conditioned clones. They also have a dechimaerising algorithm which uses ideas similar to our clone cover graph to remove clones which are apparently chimeric.

They have applied their algorithms to real data from *S.pombe.* For this organism it was possible to have a library with coverage equivalent to 47 genomes. This extremely high coverage allowed them to aim for a completely correct map. They found that they sometimes achieved the correct order for some of the chromosomes and were able to manual correct the others. They note that the minimum of their optimization function did not necessary

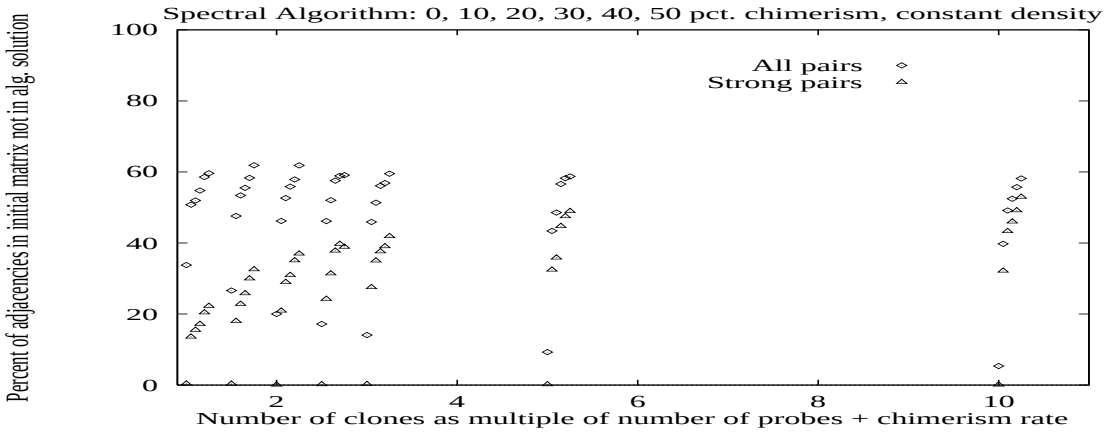

Figure 11:

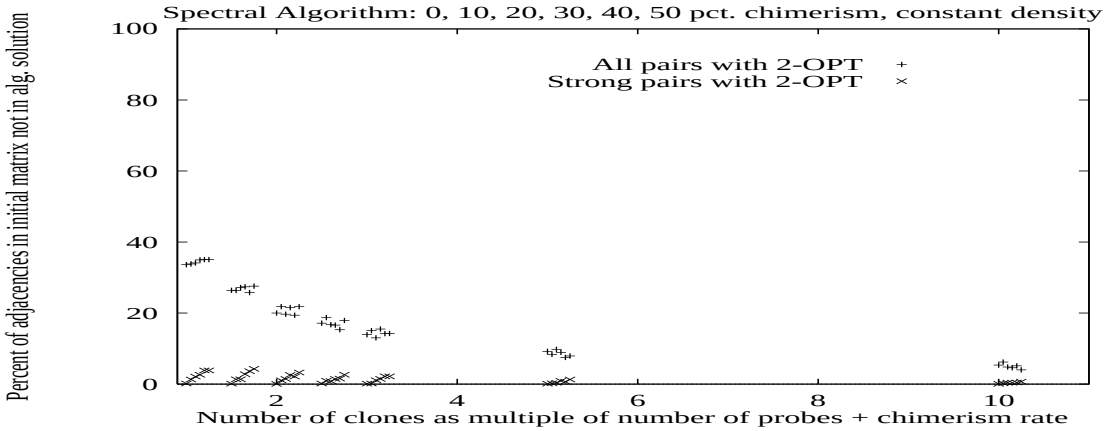

Figure 12:

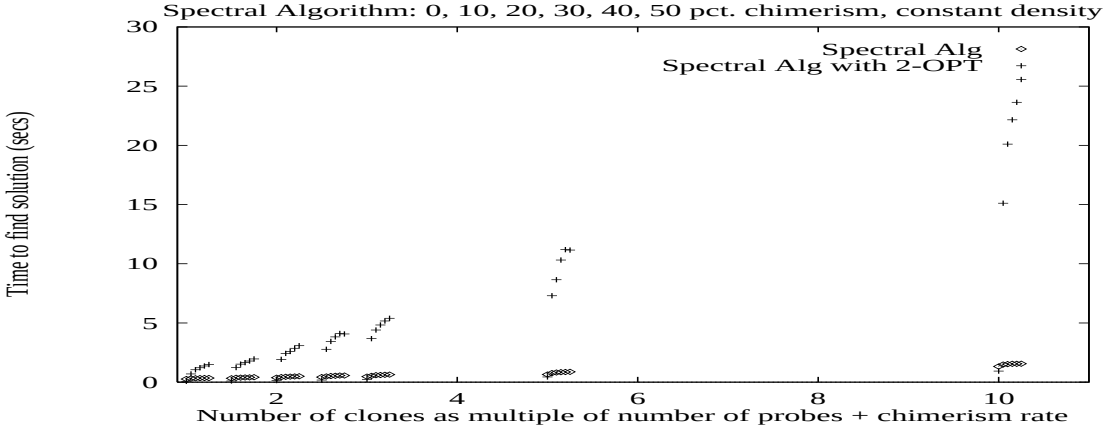

Figure 13:

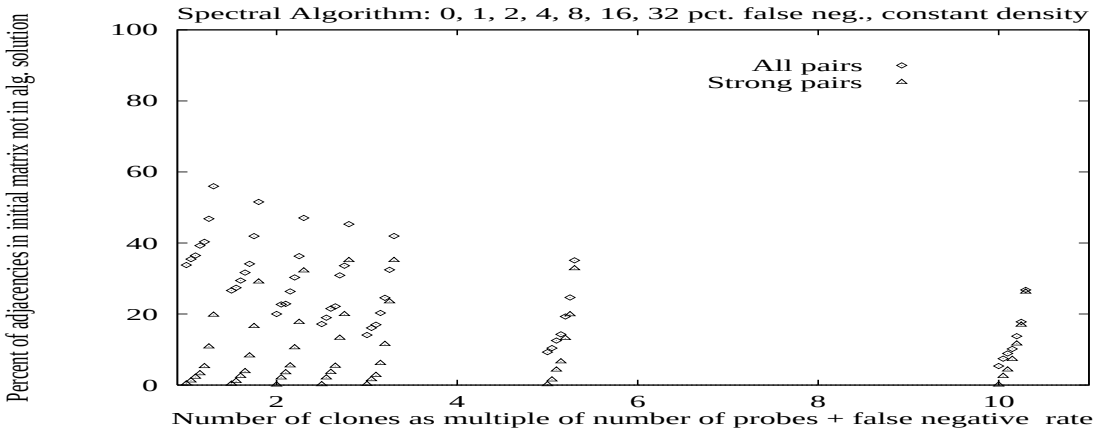

Figure 14:

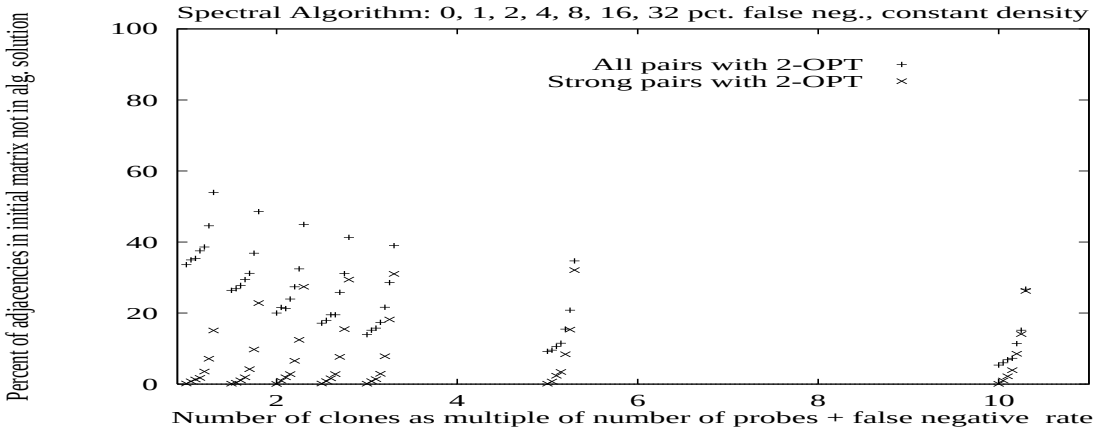

Figure 15:

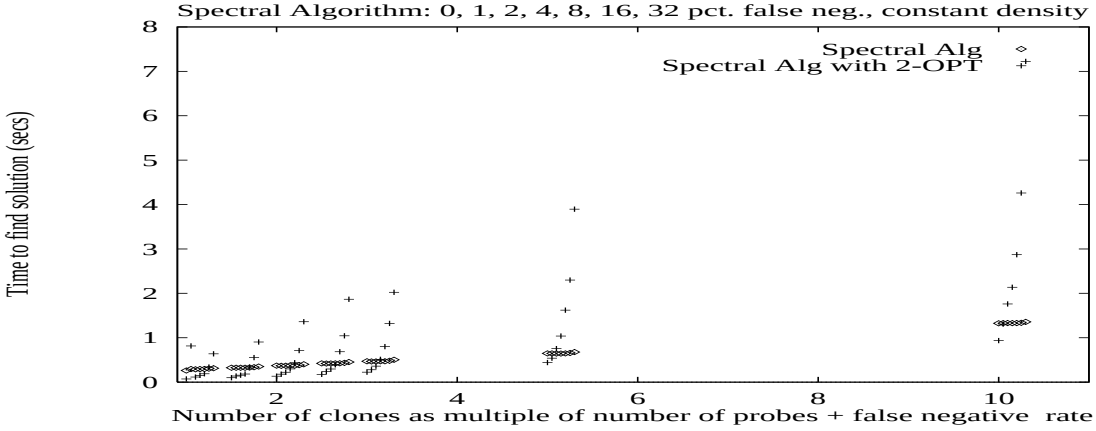

Figure 16:

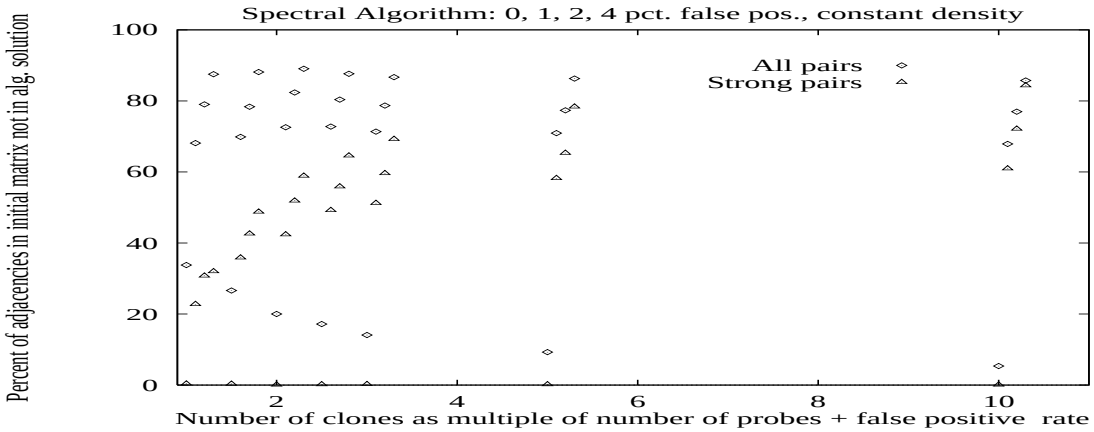

Figure 17:

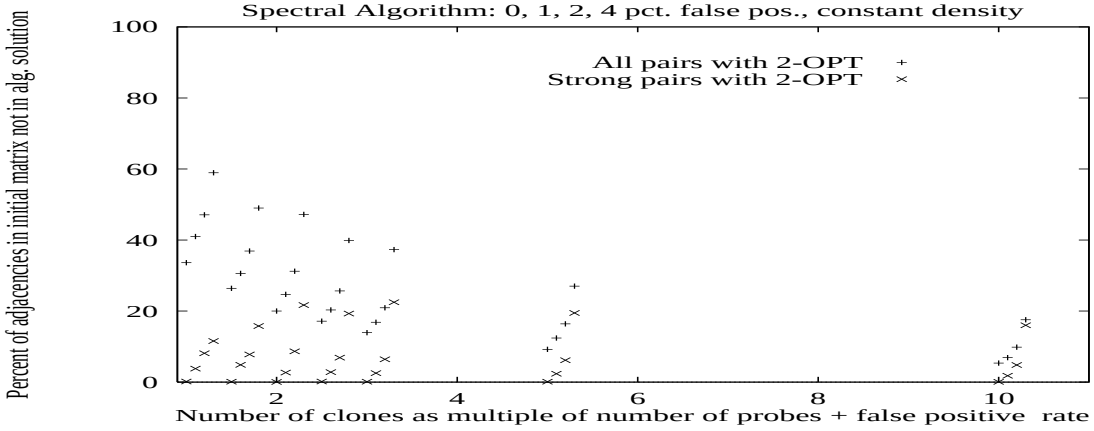

Figure 18:

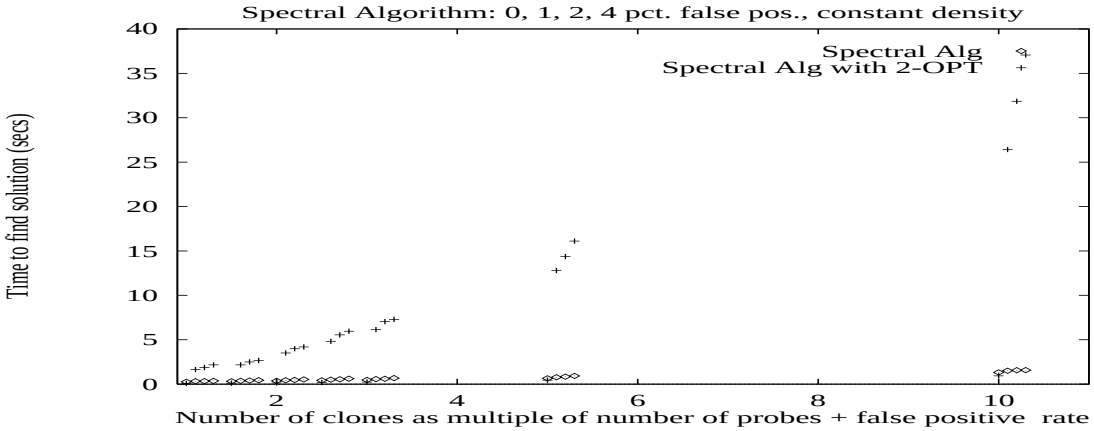

Figure 19:

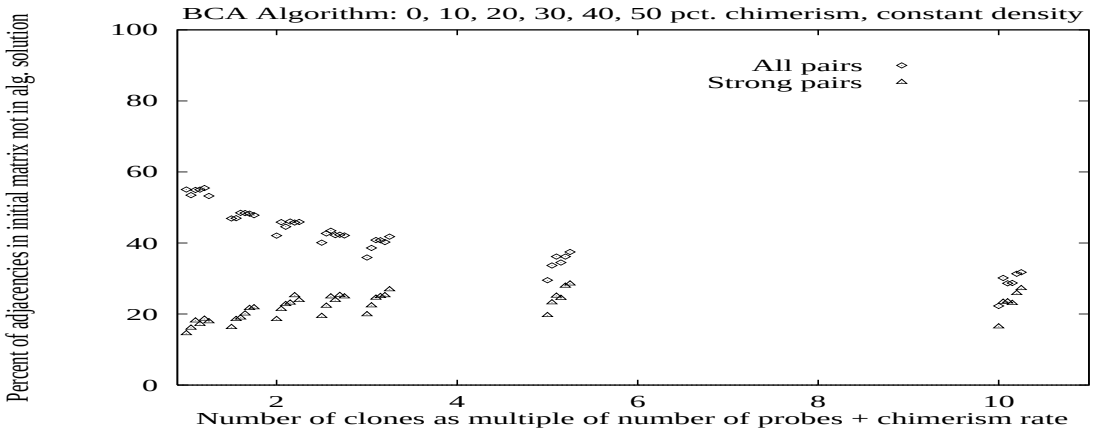

Figure 20:

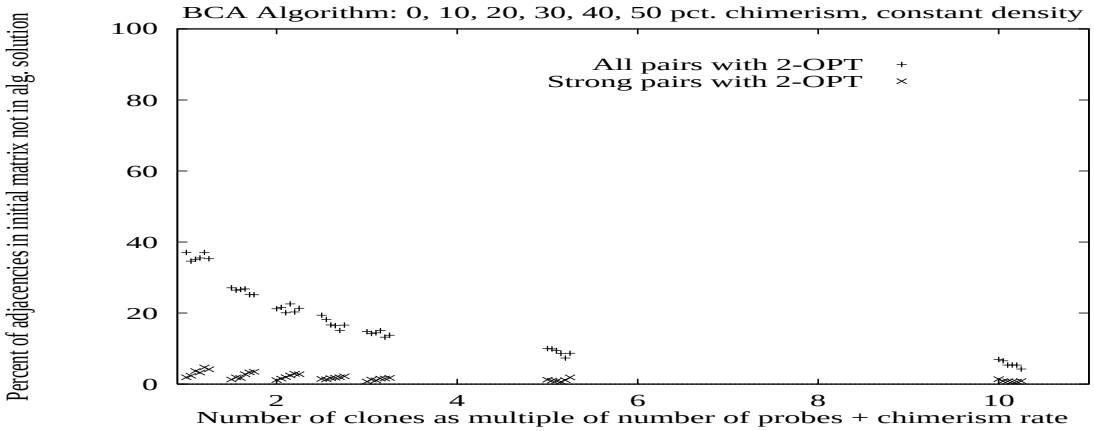

Figure 21:

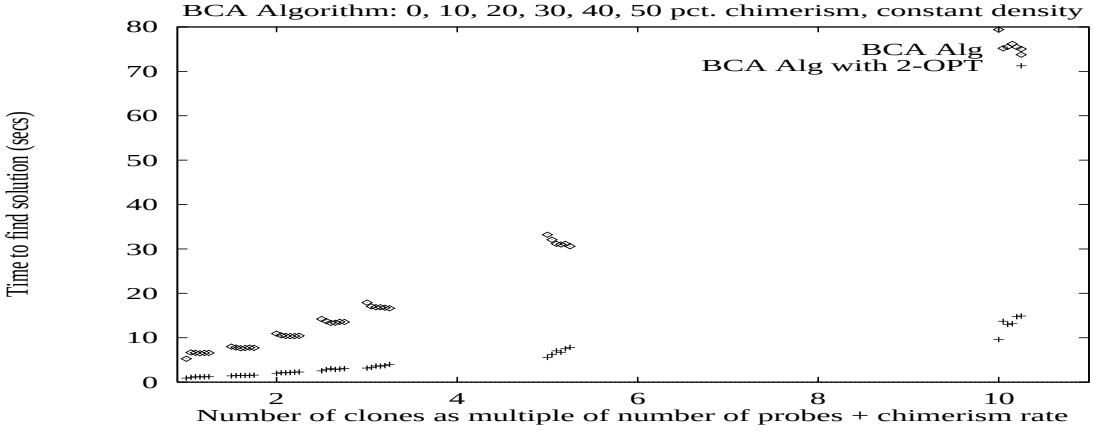

Figure 22:

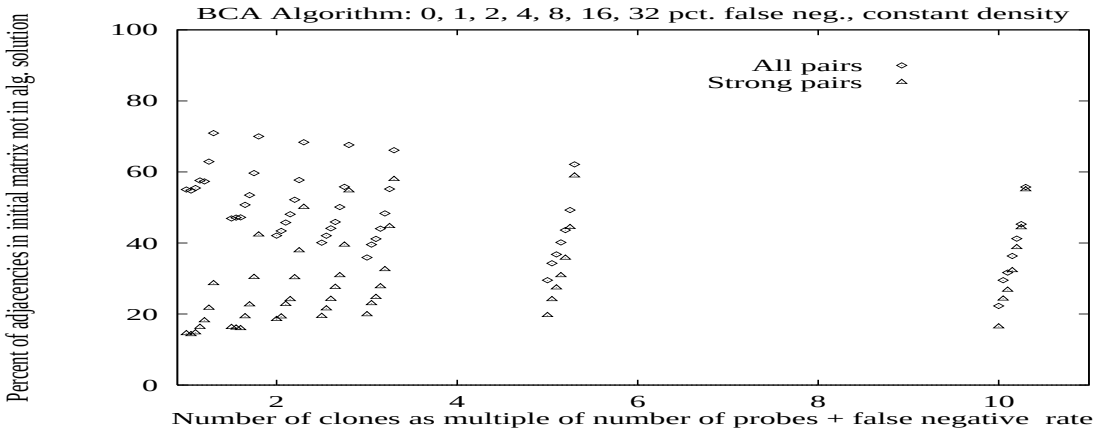

Figure 23:

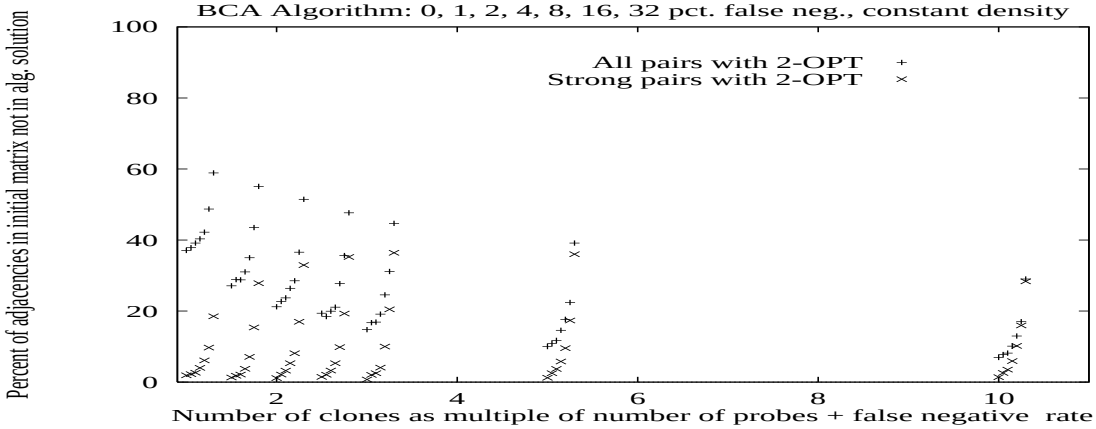

Figure 24:

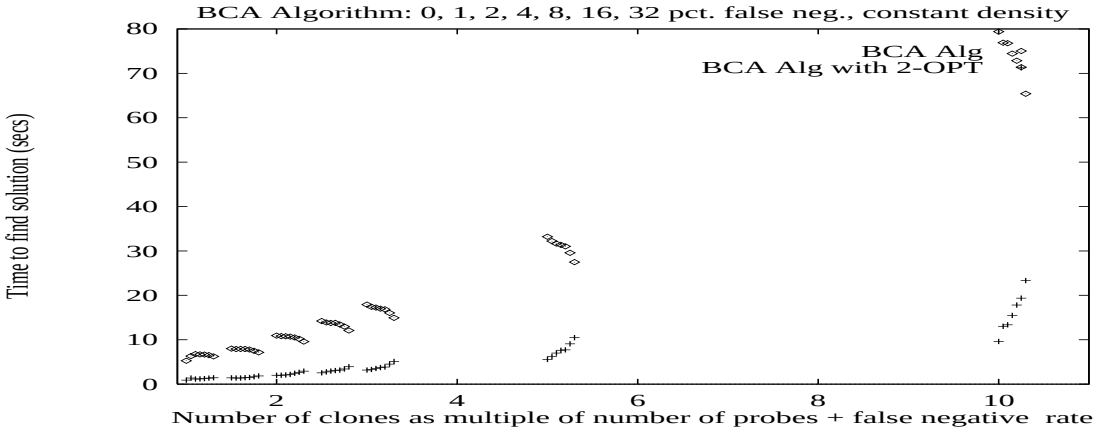

Figure 25:

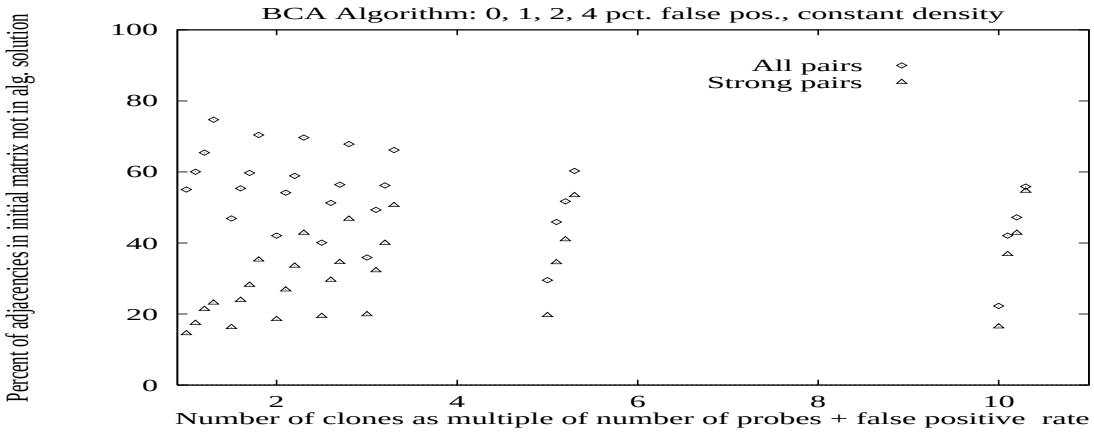

Figure 26:

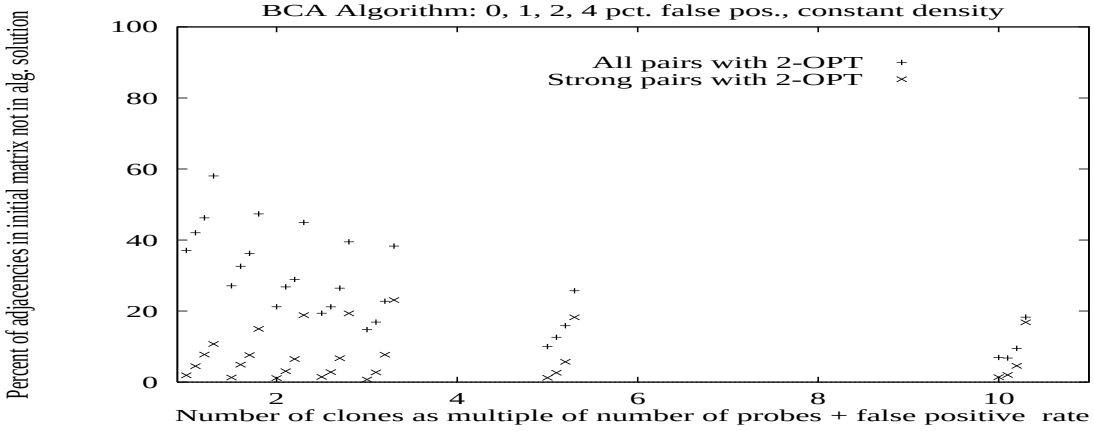

Figure 27:

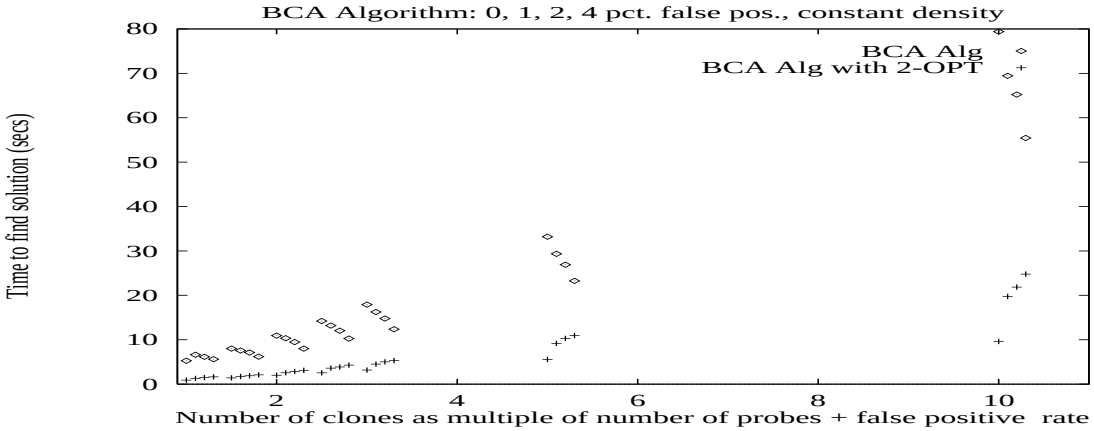

Figure 28:

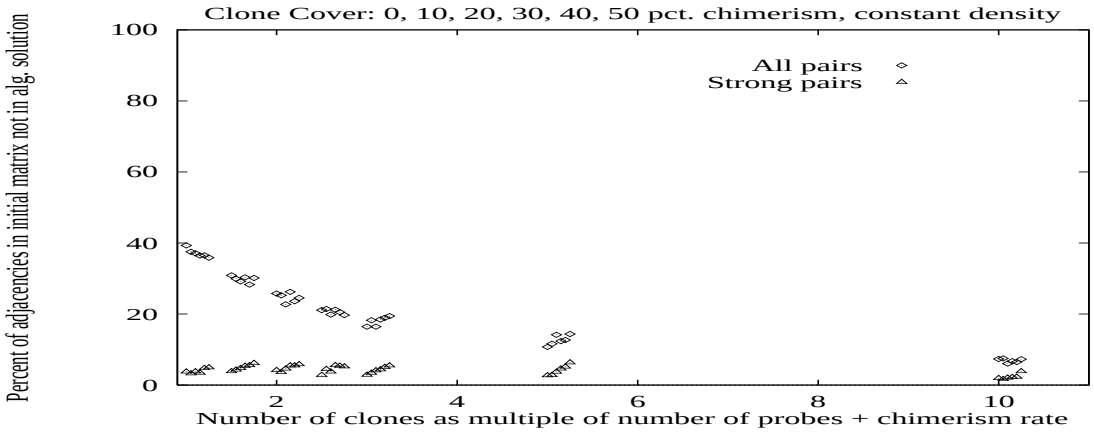

Figure 29:

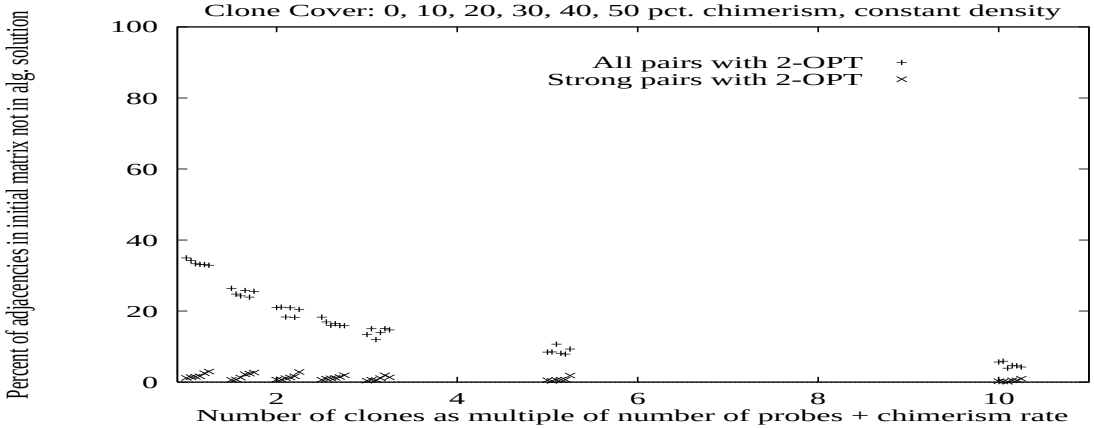

Figure 30:

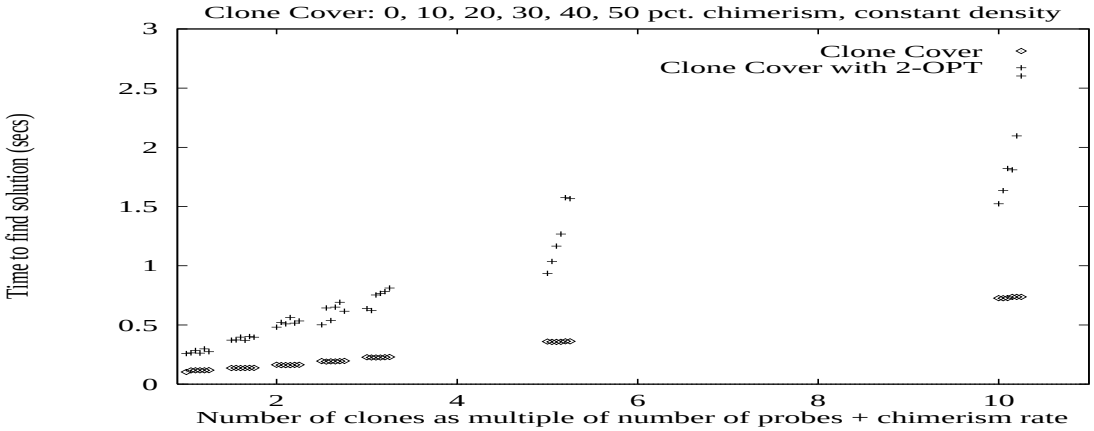

Figure 31:

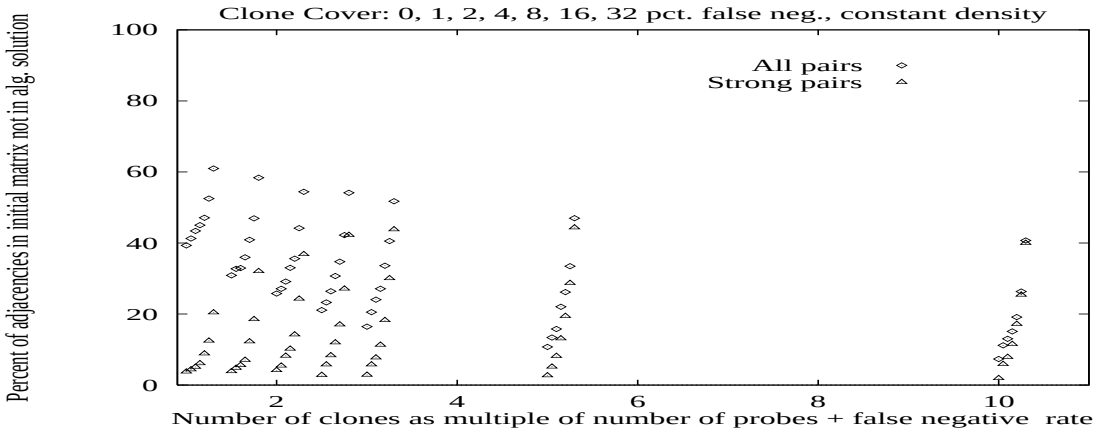

Figure 32:

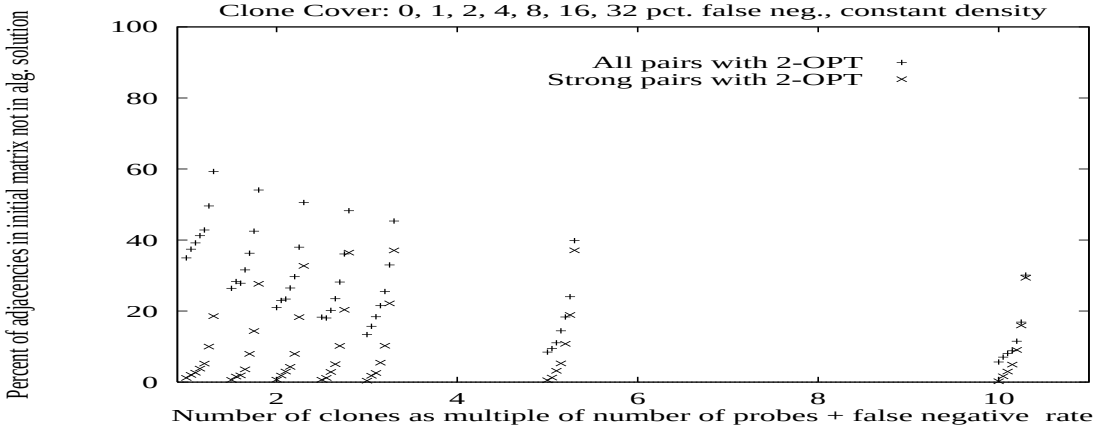

Figure 33:

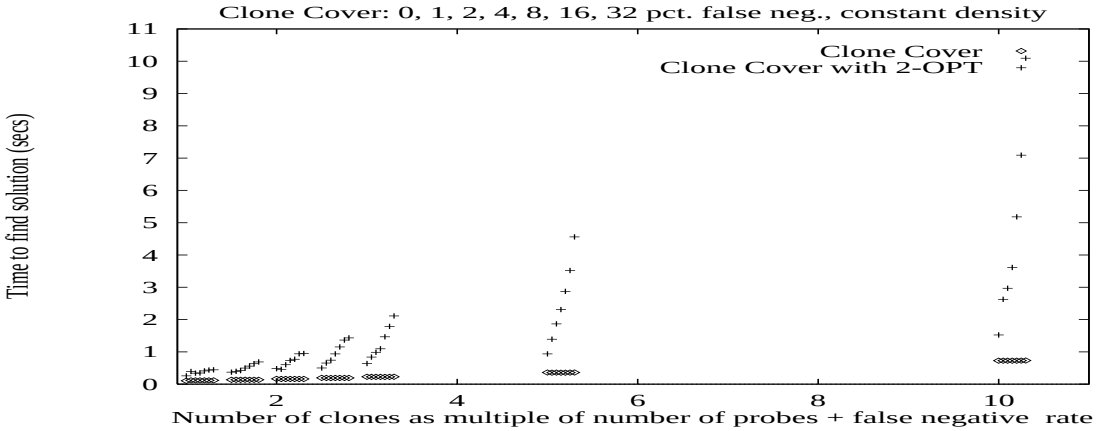

Figure 34:

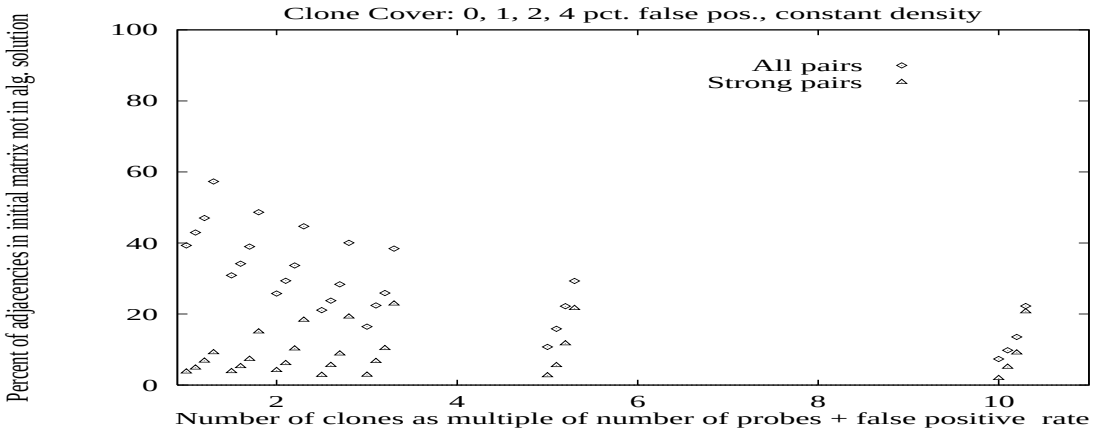

Figure 35:

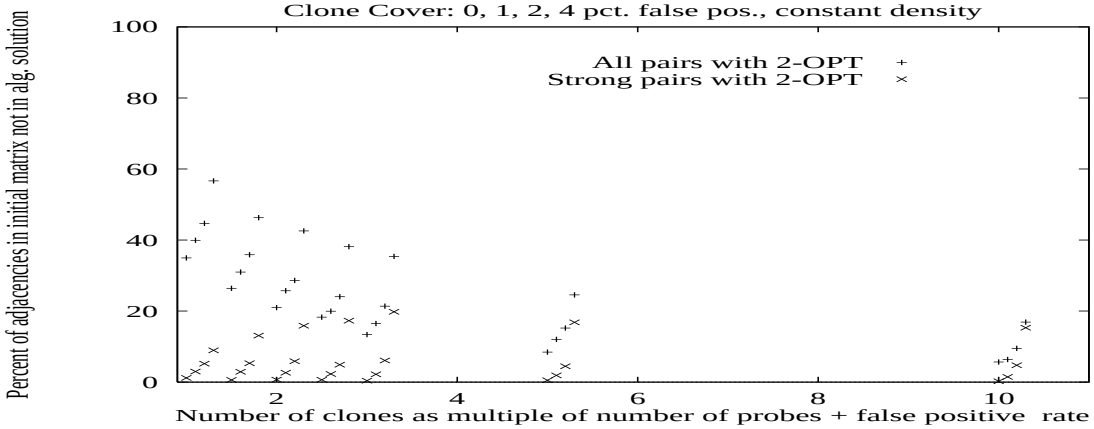

Figure 36:

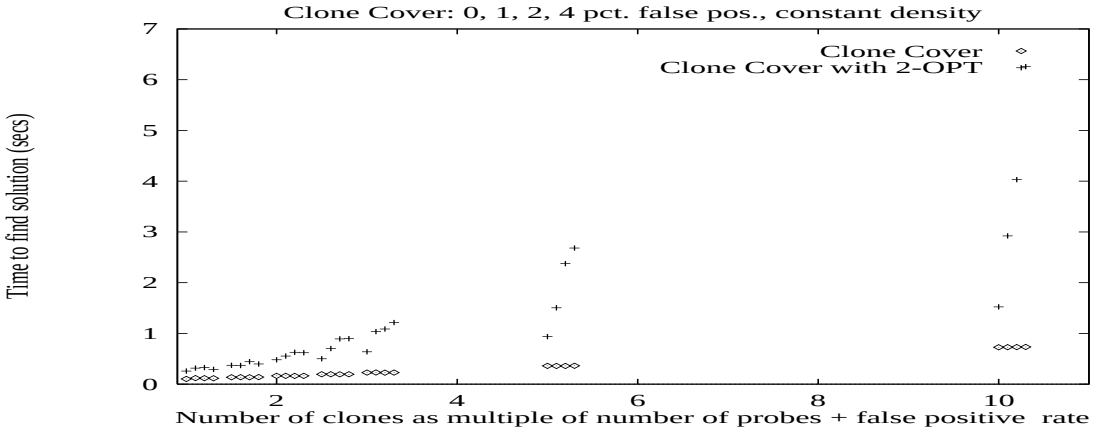

Figure 37:

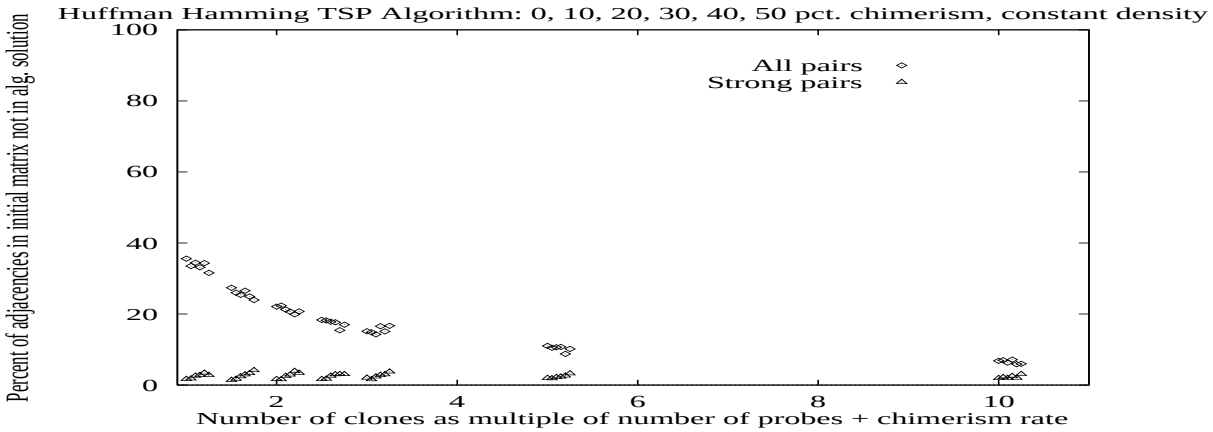

Figure 38:

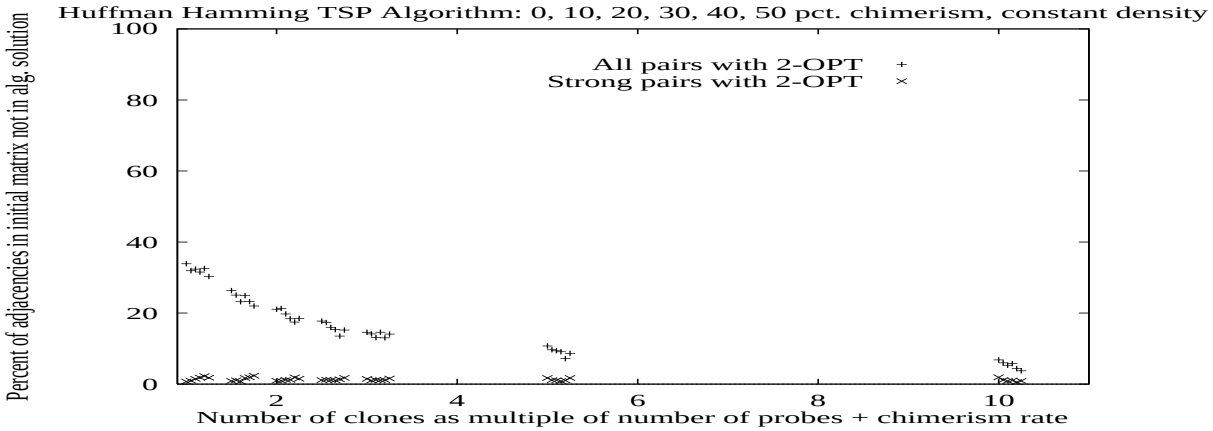

Figure 39:

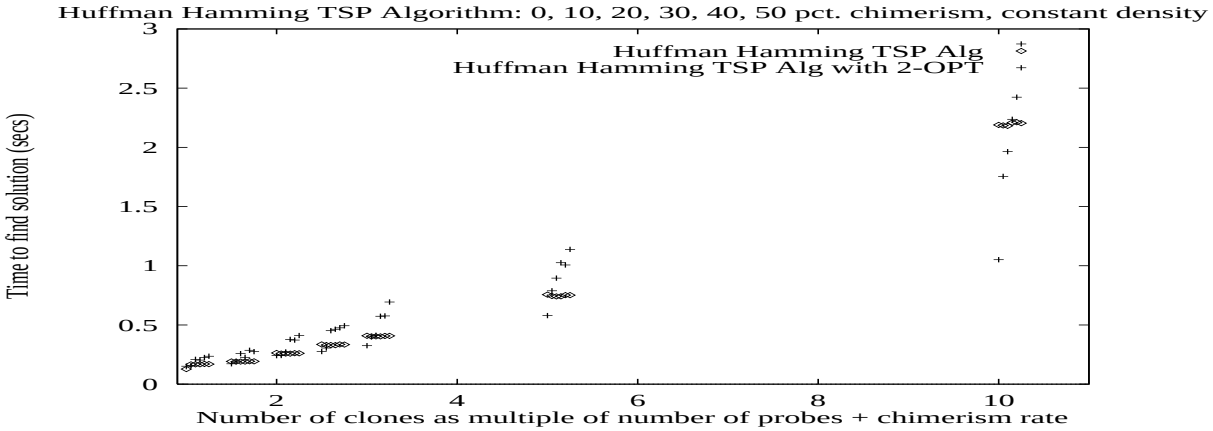

Figure 40:

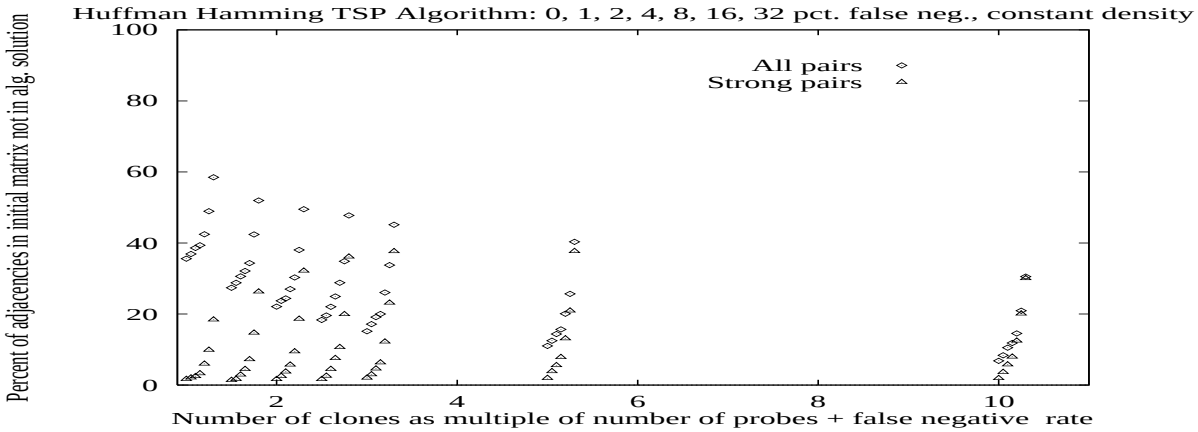

Figure 41:

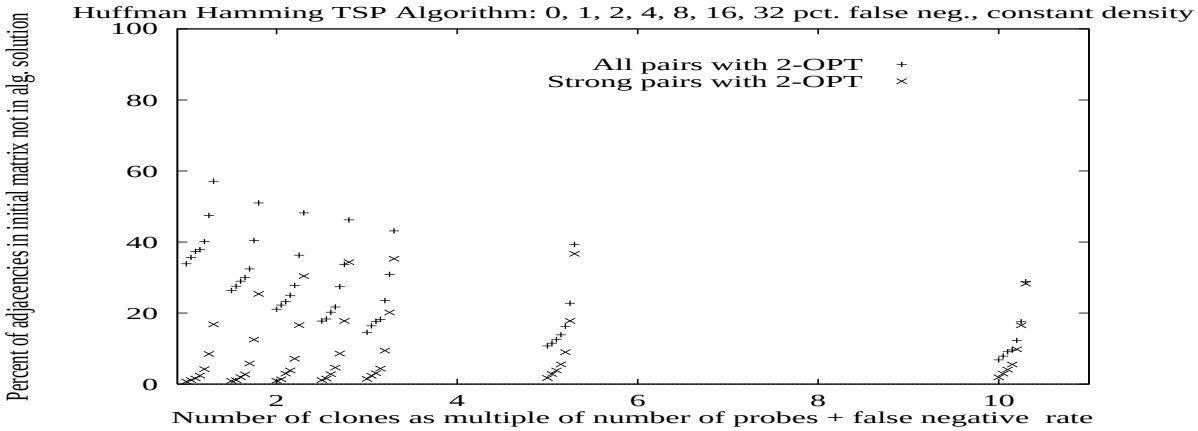

Figure 42:

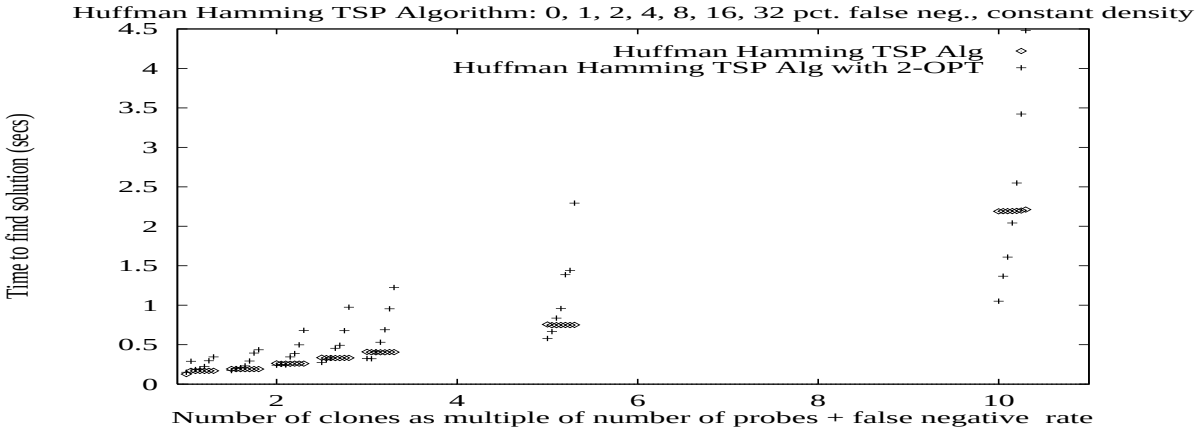

Figure 43:

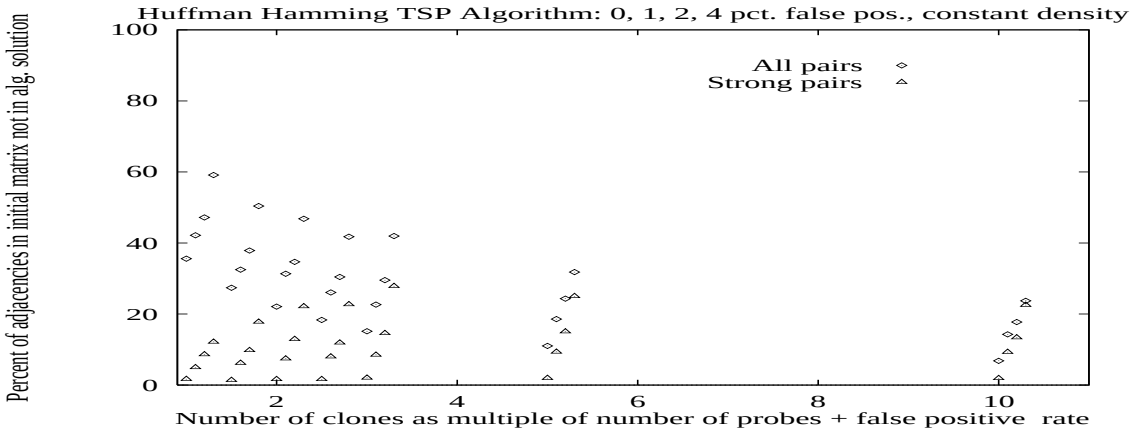

Figure 44:

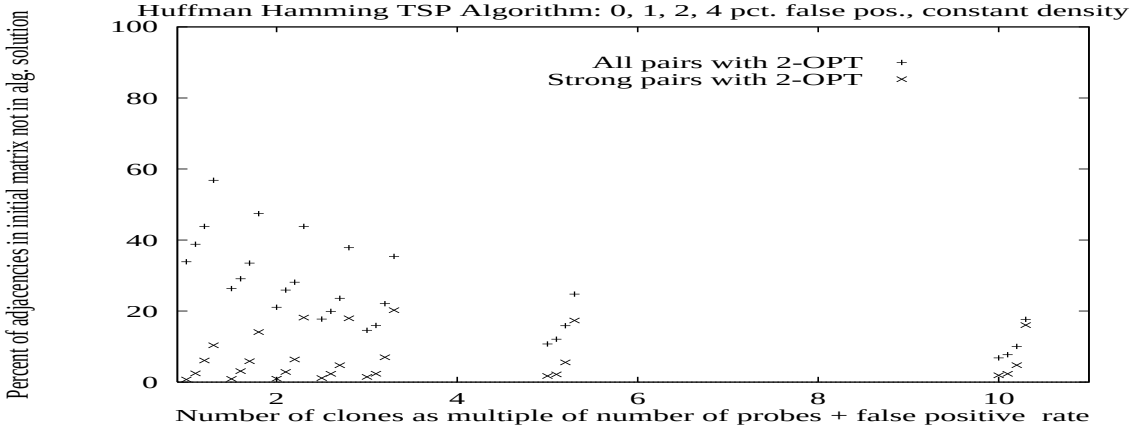

Figure 45:

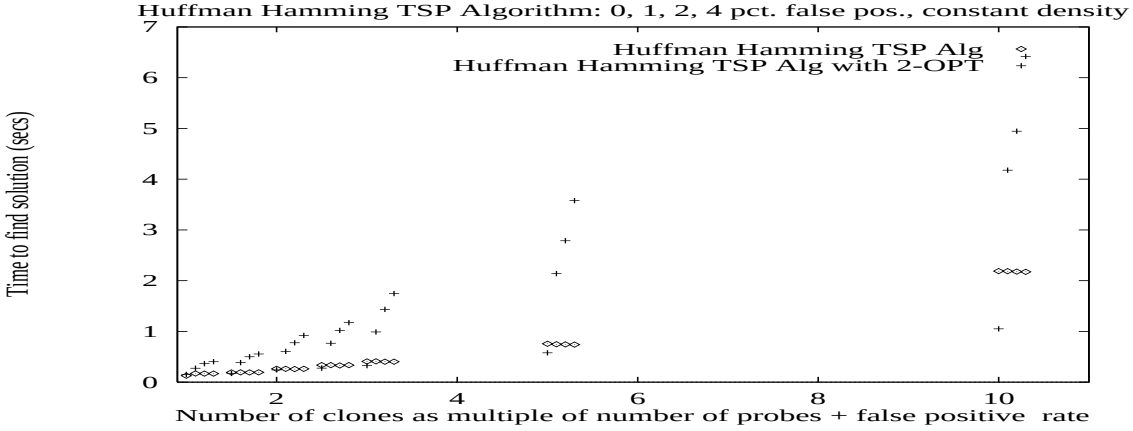

Figure 46:

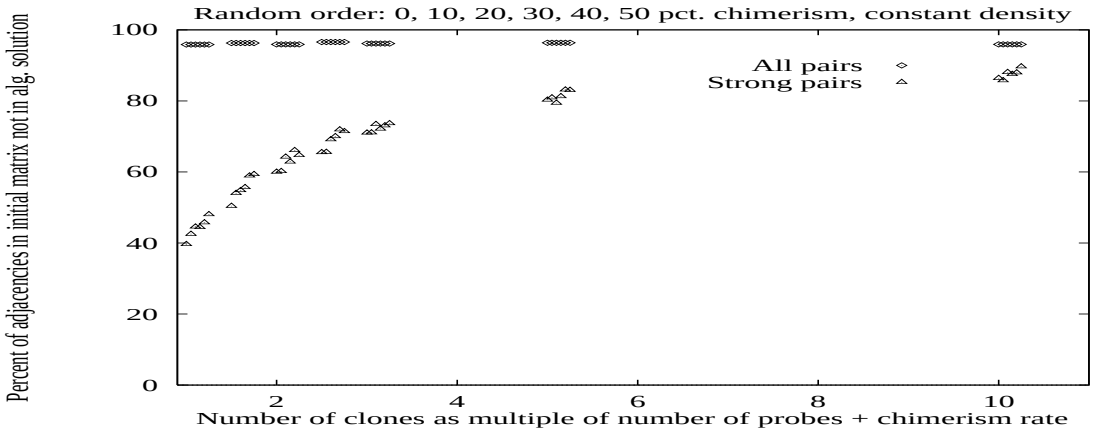

Figure 47:

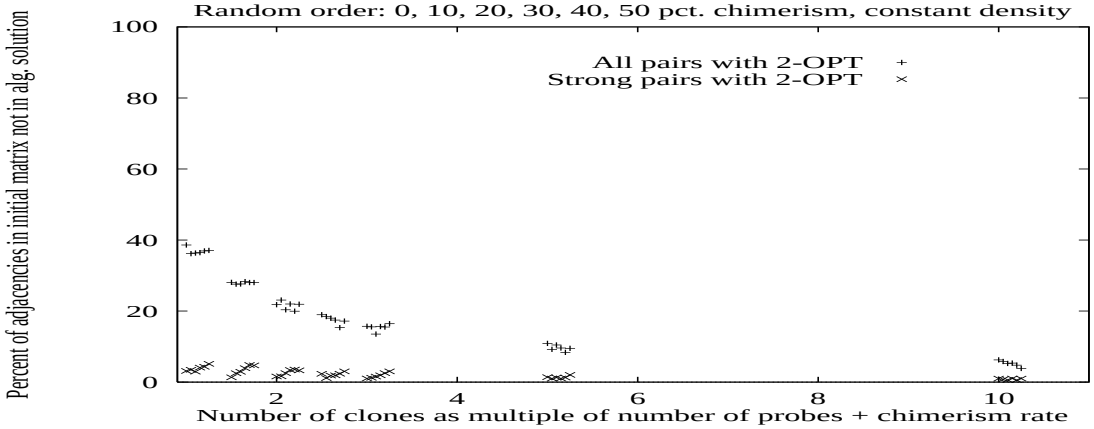

Figure 48:

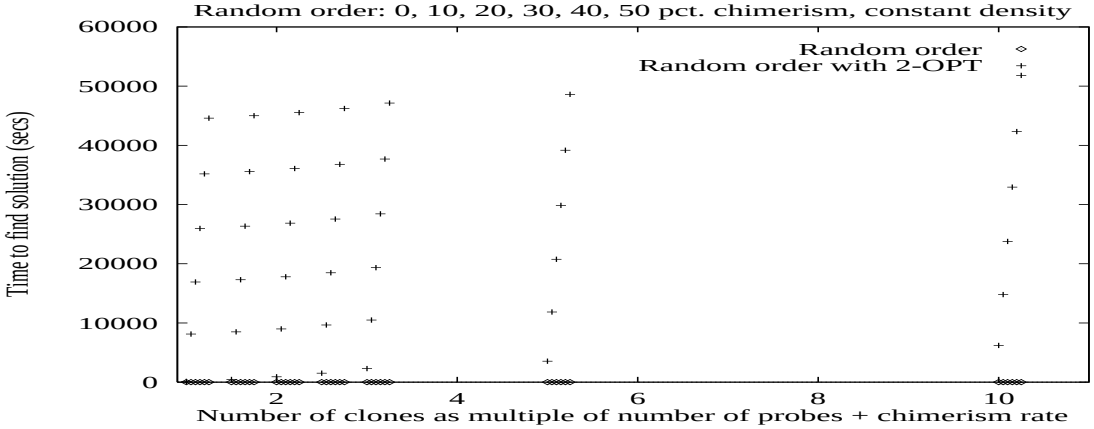

Figure 49:

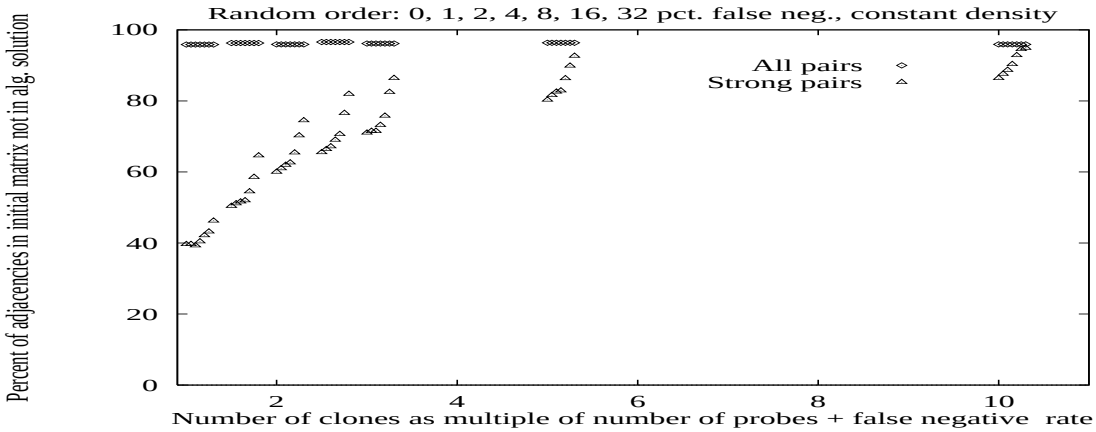

Figure 50:

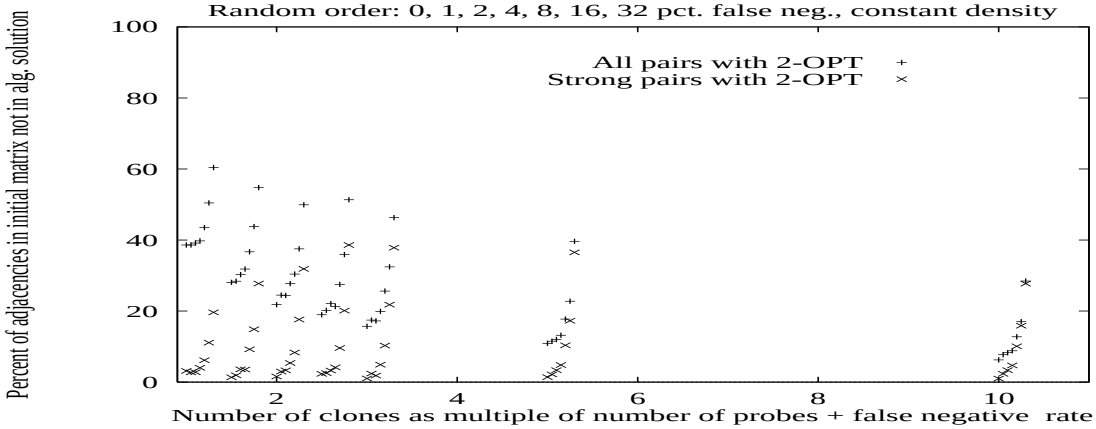

Figure 51:

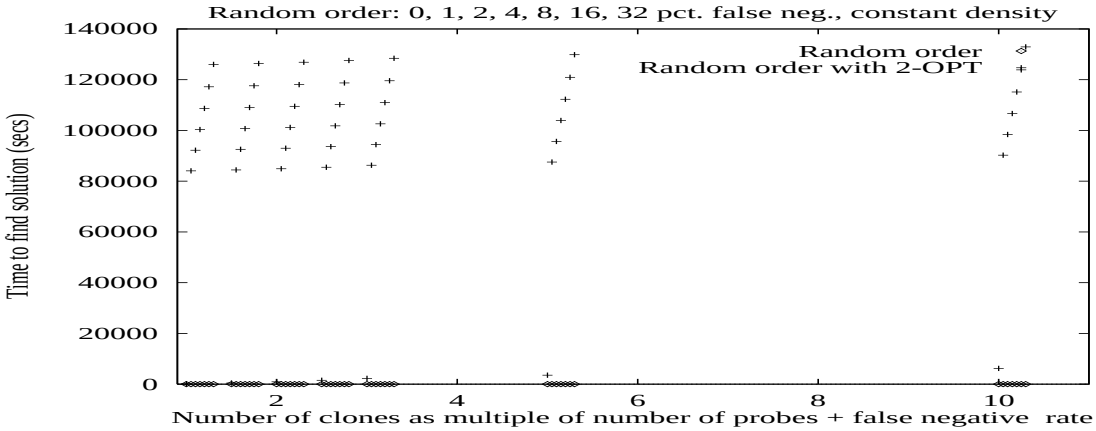

Figure 52:

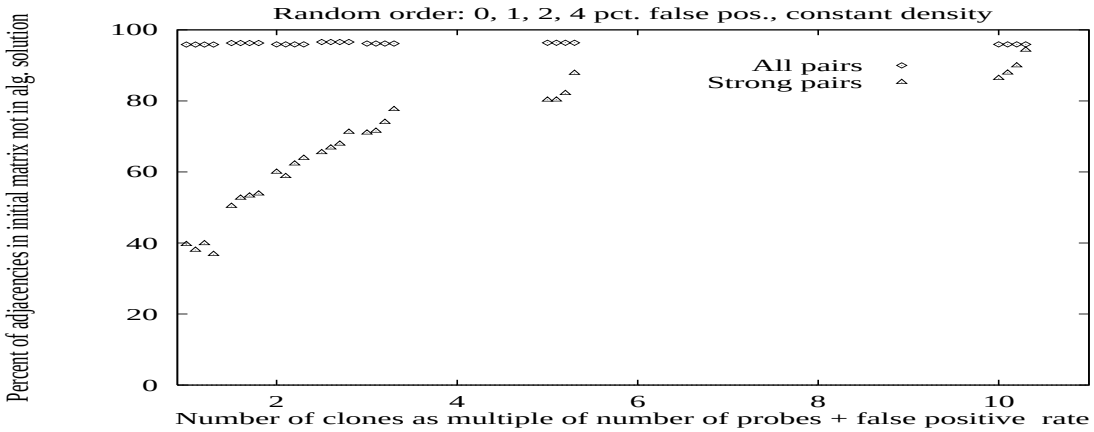

Figure 53:

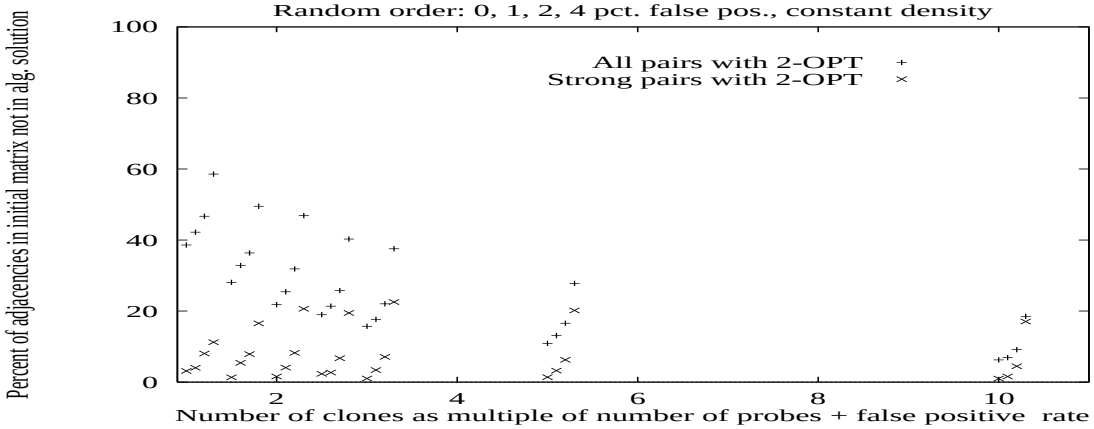

Figure 54:

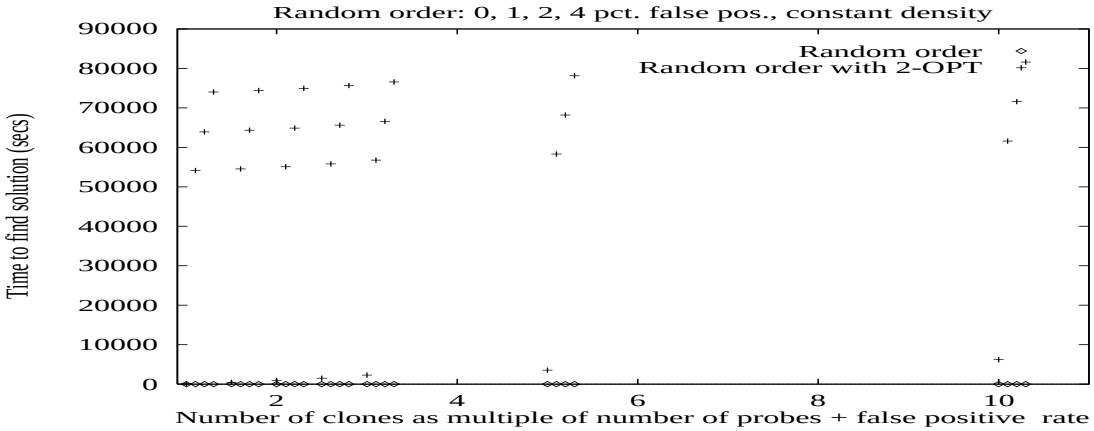

Figure 55:

	C1P	
	1/0	10/0
Spectral	33.7959	*5.34694
w/ 2-OPT	*33.6327	*5.34694
Cycle Basis	55.0612	22.2857
w/ 2-OPT	37.0612	6.93878
Clone Cover	39.3061	7.34694
w/ 2-OPT	34.9796	5.67347
Huffman	35.5918	6.81633
w/ 2-OPT	33.8776	6.81633
Random	95.8776	95.9184
w/ 2-OPT	38.6122	6.2449

	Chimerism only			
	1/.1	1/.5	10/.1	10/.5
Spectral	50.7755	59.6327	39.7551	58.1633
w/ 2-OPT	33.8367	35.0612	6.20408	4
Cycle Basis	53.4694	53.2245	30.1633	31.7551
w/ 2-OPT	34.6122	35.3061	6.57143	4.2449
Clone Cover	37.5102	35.8776	7.5102	7.26531
w/ 2-OPT	34.2041	32.898	5.87755	4.2449
Huffman	33.5102	31.5918	6.97959	6
w/ 2-OPT	*31.9184	*30.2857	6	*3.7551
Random	95.8776	95.8776	95.9184	95.9184
w/ 2-OPT	36.1633	37.0612	*5.79592	3.83673

	False positives only			
	1/.01	1/.04	10/.01	10/.04
Spectral	68.1224	87.5102	67.8776	85.7551
w/ 2-OPT	40.9796	58.9388	6.89796	17.551
Cycle Basis	60.0408	74.6939	42.0816	55.8776
w/ 2-OPT	42.0408	58.0408	6.77551	18.2857
Clone Cover	42.9388	57.3061	9.79592	22.2449
w/ 2-OPT	39.9184	*56.6531	*6.36735	*16.898
Huffman	42.1633	59.1429	14.2857	23.7143
w/ 2-OPT	*38.8163	56.8163	7.71429	17.6735
Random	95.8776	95.8776	95.9184	95.9184
w/ 2-OPT	42.2041	58.5714	6.89796	18.5306

	False negatives only			
	1/.01	1/.32	10/.01	10/.32
Spectral	35.4694	55.9592	7.42857	26.7755
w/ 2-OPT	*34.9796	*53.9592	*6	*26.7347
Cycle Basis	54.7755	70.9388	29.5102	55.7551
w/ 2-OPT	37.8367	58.898	7.7551	29.0204
Clone Cover	41.2653	60.9796	11.1837	40.6939
w/ 2-OPT	37.4286	59.3061	7.02041	30.1224
Huffman	36.898	58.4898	8.36735	30.5306
w/ 2-OPT	35.6735	57.102	7.87755	28.8163
Random	95.8776	95.8776	95.9184	95.9184
w/ 2-OPT	38.6122	60.4082	7.71429	28.449

Figure 56: Total adjacency cost(for various coverage/error rates)

	C1P	
	1/0	10/0
Spectral	0.367347	0.0816327
w/ 2-OPT	*0.163265	*0.0816327
Cycle Basis	14.5306	16.449
w/ 2-OPT	1.91837	1.38776
Clone Cover	3.7551	1.91837
w/ 2-OPT	1.1836	0.244898
Huffman	1.67347	1.91837
w/ 2-OPT	0.734694	1.91837
Random	39.6327	86.4082
w/ 2-OPT	3.14286	0.97959

	Chimerism only			
	1/.1	1/.5	10/.1	10/.5
Spectral	13.5918	22.2449	32.1633	52.9796
w/ 2-OPT	1.30612	3.83673	*0.2449	*0.69388
Cycle Basis	16.0816	17.8776	23.3878	27.2245
w/ 2-OPT	2.32653	4.16327	0.81633	0.85714
Clone Cover	3.22449	4.93878	1.63265	3.83673
w/ 2-OPT	1.38776	2.97959	0.16326	0.93878
Huffman	1.79592	2.77551	2.16327	3.06122
w/ 2-OPT	*0.93878	*1.79592	1.14286	0.89796
Random	42.5306	48.0408	85.7551	89.6735
w/ 2-OPT	3.46939	5.14286	0.77551	0.97959

	False positives only			
	1/.01	1/.04	10/.01	10/.04
Spectral	22.7755	32	60.9388	84.3265
w/ 2-OPT	3.79592	11.551	1.79592	16
Cycle Basis	17.4286	23.1429	36.8163	54.6531
w/ 2-OPT	4.4898	10.7347	2	16.8163
Clone Cover	4.85714	9.18367	5.10204	20.6939
w/ 2-OPT	2.97959	*8.93878	*1.46939	*15.3061
Huffman	4.97959	12.1633	9.30612	22.5306
w/ 2-OPT	*2.44898	10.3673	2.32653	16.0408
Random	38.0408	36.8571	87.9184	94.3673
w/ 2-OPT	4.04082	11.2245	1.59184	17.0612

	False negatives only			
	1/.01	1/.32	10/.01	10/.32
Spectral	1.22449	19.7551	2.53061	26.2449
w/ 2-OPT	*0.77551	*15.102	*1.02041	*26.2449
Cycle Basis	14.2449	28.6122	24.2857	55.0612
w/ 2-OPT	2.2449	18.5306	2.57143	28.3673
Clone Cover	4.36735	20.4082	5.87755	40
w/ 2-OPT	2.04082	18.5714	1.63265	29.3878
Huffman	2.08163	18.4082	3.63265	30.0408
w/ 2-OPT	1.14286	16.8163	2.93878	28.2857
Random	39.6735	46.2449	87.5918	94.898
w/ 2-OPT	2.73469	19.6327	2.32653	27.7143

Figure 57: Strong adjacency cost (for various coverage/error rates)

correspond to the correct order.
Another prominent software package that is based on simulated annealing is the package developed by [9, 8]. A simulated annealing heuristic is used to construct the Hamming distance TSP tour. They look at several measures of success including the size of the largest contig, the number of contigs, and the error within a contig. The emphasis on separate contigs is a version of our concept of weak adjacencies, the measure of error within a contig is related to our notion of average probe distance. They applied their algorithm to simulated data which included errors which we called unclone-able regions and false positives. They also applied their algorithm to the genome of *A. nidulans*.
The Berkeley software library developed by R. Karp's group [1] also employs simulated annealing to the TSP graph. In addition they propose several heuristics for reducing the amount of false positives in the data, dealing with repeated probes, and pooling schemes. They also include analysis of data in which probes are built from the ends of clones.
Zhang *et al.* look at data which contains clone-clone hybridization data rather than clone-probe data. They use a classic algorithm for finding a diameter of a graph in order to estimate an ordering of the clones. In order to deal with false hybridization they apply several heuristics. For the library of cosmids and YACS for human chromosome 13 to which they applied their techniques there was a three to five times coverage. They do not give any quantitative measure of how successful their algorithms were.

12 Summary and Future Directions

Although we feel that we have moved forward a great deal from our earlier work in physical mapping, we are sure that much remains to be done. At a minimum, there remain open questions in the creation of generators of synthetic data, the creation of algorithms based on combinatorial optimizations, the consideration of algorithms based on other techniques such as maximum likelihood, the inclusion of data of other types besides hybridizations, the modeling of errors and inclusion of other error types, the definition of more detailed measures of algorithm success, and the construction of more detailed experiments.

Synthetic Data We discussed the question of generators in some depth in Section 5 but a few points are worth reiterating. No one is likely to produce a generator which actually captures all of the intricacies of the experimental mapping process. The key is to attempt to include facets of the process which are likely to have large effects on the evaluation of algorithms. Toward this goal we have looked at the distribution of probe positions, the size of clone fragments, and the amount of simple errors. We believe that it will be important to further examine the role of the distribution of fragment sizes, the correlation of various error types, and the correlation of errors to fragment position.

Algorithms In our discussion of algorithms in Section 8 we noted that it was desirable to find optimization functions which are conservative extensions of the C1P problem and which are monotonic in the error rate. We continue to look for better such functions. In addition, it is clear to us that the use of multiple algorithms to increase the reliability of the results and allow the handling of multiple error types will be crucial. We are exploring

ways to combine the outputs of multiple algorithms and to use them in iterative schemes which progressively improve the solution.
The use of the 2-OPT refinement opens up many questions. We used 2-OPT to locally search for solutions with the lowest value of σ. However, we could equally well have used any other (or combination) of our optimization functions. It would be interesting to see whether there is an advantage to having the 2-OPT use a different (or the same) function as the primary algorithm.
We also expect that new research avenues will develop based on combinatorial optimizations of multiple objective functions. The area of multiple objective optimization is an active area of research in combinatorial optimization and we expect that the efforts for providing computational support for mapping will reinforce the need for new algorithmic tools and methods.

New data and error types There will always be new experimental techniques with corresponding new types of information and errors. It is our hope that some of the mechanisms developed in this paper will be applicable to the resulting mapping problems. One of the greatest challenges to our approach is to allow the biologist to include side information. For example, if it is known from a break point analysis that two sets of probes are separate from each other it would be nice to allow the algorithm to make use of this information.

Experimental Analysis We learned many lessons from performing the experiments of Section 10. An important lesson was that summary statistics is not always enough. It is sometimes important to be able to directly compare the solutions of all algorithms on a single input instance. It would be nice to have additional tools for comparing how close two permutations are to each other. These tools might also provide us with additional candidates for measuring relative algorithm success.
A second lesson we learned is that it is very difficult to hold everything constant except one factor. For example, varying the error rates changed the density of the resulting matrix. It is important to keep a record of all parameters, even ones not being studied in order to be able to look for later correlations.
We were not particularly careful about implementation efficiency. This sometimes meant we had to limit the size of our examples and/or the number of trials examined. In both cases this meant that we were not able to reach as high a level of confidence in our output numbers as we would have liked. We computed standard deviations, minimums, and maximums, as well as averages. The averages seem to be reliable measure but we would like to make the study more rigorous. In addition, the use of run time as a measure of algorithm efficiency is dubious. We would like to re-examine the use of 2-OPT in order to count the number of optimization steps. We suspect that it will be interesting to count not only the total number of steps until local optimum is found but also the number of steps until a solution which is within a threshold of the local optimum is found.

Theory Our theory of physical mapping in the presence of errors is still in its early stages. We hope to extend the theory to uncover intrinsic limitations of distinguishing noise from signal in physical mapping. In particular, we are currently use only very simple noise

models. Determining how to properly weight the effects of different error sources will be challenging.
Maximum likelihood approaches implicitly include a noise model. We hope that by making the noise model explicit that we can determine when enough information is available to make maximum likelihood effective.

Integration into mapping projects We have presented the problem of physical mapping as occurring after the biological experiments are completed. This is probably not the most efficient use of mapping algorithms. It would be preferable if the mapping software became integrated into the mapping process. For example, the software should be able to help the biologists know which sections of the data "need work". The hope is that the integrated approach will lead to improved pathways to successful maps. The question of how to make this integration happen will undoubtedly be one of the major issues for mapping software in the next several years.

Acknowledgments

It is a pleasure to thank Eric Lander for suggesting this research area to us, for his technical contributions and his support. Ernie Brickell and Fred Howes' support have been crucial to this effort. Bruce Hendrickson, James Park, Cindy Phillips, and Mike Sipser provided helpful reviews, comments, and valuable technical contributions. Norman Doggett and Manny Knill shared with us both their data of chromosome 16 and their ideas about physical mapping with us. We would also like to thank Leslie Goldberg, Jim Orlin, David Torney, and Mike Waterman for fruitful discussions.
Jon Atkins and Paul Goldberg contributed proofs that certain optimization functions were NP-hard. Bruce Hendrickson and Rob Leland not only gave us access to their code for envelope minimization (from their Chaco software) but made modifications to it in order to interface it to our codes. Cathy McGeoch, Henry Shapiro, and Bernard Moret gave us many helpful tips on experimental analysis.

References

[1] F. Alizadeh, R. Karp, L. Newberg, and D. K. Weiser. Physical mapping of chromosomes: a combinatorial problem in molecular biology. In *Proceedings of the 4th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 371–381, 1993.

[2] J. Atkins. Testing whether genomic matrices have chimeric number two is np-complete. Manuscript, Sandia Labs, 1994.

[3] J. Atkins, E. Boman, and B. Hendrickson. A spectral algorithm for the seriation problem. Technical Report SAND94–03082, Sandia National Laboratories, Albuquerque, NM, 1994.

[4] K. Booth and G. Lueker. Testing for the consecutive ones property, interval graphs and graph planarity using PQ-tree algorithms. *J. Comput. Syst. Sci.*, 13:333–379, 1976.

[5] T. A. Brown. *Gene cloning*. Chapman and Hall, second edition, 1990.

[6] I. Chumakov et al. Continuum of overlapping clones spanning the entire human chromosome 21q. *Nature*, 359:380–387, 1992.

[7] D. Cohen, I. Chumakov, and J. Weissenbach. A first-generation physical map of the human genome. *Nature*, 366:698–701, 1993.

[8] A. Cuticchia, J. Arnold, and W. Timberlake. The use of simulated annealing in chromosome reconstruction experiments based on binary scoring. *Genetics*, 132:591–601, 1992.

[9] A. Cuticchia, J. Arnold, and W. Timberlake. Ods: ordering dna sequences – a physical mapping algorithm based on simulated annealing. *CABIOS*, 9(2):215–219, 1993.

[10] A. Cuticchia, J. Arnold, and W. E. Timberlake. The use of simulated annealing in chromosome reconstruction experiments based on binary scoring. *Genetics*, 132:591–601, 1992.

[11] M. Fiedler. Algebraic connectivity of graphs. *Czechoslovak Math. J.*, 23(98):298–305, 1973.

[12] M. Fiedler. A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory. *Czechoslovak Math. J.*, 25(100):619–633, 1975.

[13] S. Foote, D. Vollrath, A. Hilton, and D. C. Page. The human Y chromosome: Overlapping DNA clones spanning the euchromatic region. *Science*, 258:60–66, 1992.

[14] D. Fulkerson and O. Gross. Incidence matrices and interval graphs. *Pacific J. of Mathematics*, 15(3):835–855, 1965.

[15] M. Garey and D. Johnson. *Computers and Intractability*. Freeman and Co., 1979.

[16] P. Goldberg. The generalized consecutive ones property is NP-complete. Technical report, Sandia National Laboratories, Dec. 1992. also included in *Four strikes against physical mapping of DNA* by Goldberg, Golumbic, Kaplan and Shamir, Tel Aviv University Tech Report 287.

[17] E. Green, H. Reithman, J. Dutchik, and M. V. Olson. Detection and characterization of chimeric yeast artificial-chromosome clones. *Genomics*, 11:658–669, 1991.

[18] D. Greenberg and S. Istrail. The chimeric mapping problem: algorithmic strategies and performance evaluation on synthetic genomic data. *Computers chem.*, 18(3):207–230, 1994.

[19] D. Greenberg and C. Phillips. In preparation.

[20] A. Grigoriev, R. Mott, and H. Lehrach. An algorithm to detect chimeric clones and random noise in genomic mapping. *Genomics*, 22:282–486, 1994.

[21] B. Hendrickson and R. Leland. The Chaco user's guide, version 2.0. Technical Report SAND94–2692, Sandia National Laboratories, Albuquerque, NM, October 1994.

[22] M. Juvan and B. Mohar. Optimal linear labelings and eigenvalues of graphs. *Disc. Appl. Math.*, 36:153–168, 1992.

[23] E. Knill. Lower bounds for identifying subset members with subset queries. In *Proceedings of the 6th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 369–377, 1995.

[24] L. T. Kou. Polynomial complete consecutive information retrieval problems. *SIAM J. Comput.*, 6(1):67–75, 1977.

[25] E. S. Lander and M. S. Waterman. Genomic mapping by fingerprinting random clones: A mathematical analysis. *Genomics*, 2(3):231–329, 1988.

[26] P. Little. Mapping the way ahead. *Nature*, 359:367–368, 1992.

[27] R. Mott, A. Grigoriev, E. Maier, J. Hoheisel, and H. Lehrach. Algorithms and software tools for ordering clone libraries: application to the mapping of the genome *schizosaccharomyces pombe*. *Nucleic Acid Research*, 21(8):1965–1974, 1993.

[28] D. Nelson and B. H. Brownstein. *YAC Libraries: A User's Guide*. W. H. Freeman and Co., 1993.

[29] B. Parlett and D. Scott. The Lanczos algorithm with selective orthogonalization. *Math. Comp.*, 33:217–238, 1979.

[30] D. Torney. Mapping using unique sequences. *J. Mol. Biol.*, 217:259–264, 1991.

[31] D. Vollrath, S. Foote, A. Hilton, L. Brown, P. Beer-Romero, J. Bogan, and D. C. Page. The human Y chromosome: a 43-interval map based on naturally occurring deletions. *Science*, 258:52–59, 1992.