An Efficient Algorithm for Ordering Random Genomic Clones

Kyungsook Han
kshan@cs.rutgers.edu

Michiel Noordewier
noordewi@cs.rutgers.edu

Computer Science Department
Rutgers University
Piscataway, NJ 08855
Fax: 908-932-0537

July 2, 1992

Abstract

We propose an algorithm which efficiently constructs a physical map of DNA from a fingerprinted library of random clones. The algorithm clusters clones into contiguous regions using restriction enzyme digest data. Clones which contain overlapping genomic regions are detected and are grouped into an island in such a way that disjoint islands which can be linked by the clones are joined together automatically. This approach solves a common problem in fingerprinting – linking disjoint islands becomes more difficult and slow as mapping goes on. A map is produced in $O(km^2n)$ time where k is the number of cleavage sites of a clone, m is the number of clones in a library, and n is the number of libraries. Our approach provides a general and efficient means for constructing long-range physical maps in large-scale genome mapping.

key words: physical map, fingerprinting, restriction enzymes, DNA

1 Introduction

The molecular characterization of a genome involves the construction of a *physical map*, which is an ordered, overlapping set of clones. Such a set, also known as a "contig library", is a molecular map of an organism. A *genetic linkage map* is collinear with the physical map, but there is no constant scale factor that relates genetic and physical distances. However, alignment of the physical and genetic maps would allow, for example, the straightforward identification of the locus of any gene for which a probe is available. In addition, it would greatly assist in the isolation of genes defined only by a mutant phenotype and a recombination map position.

The most common strategy for physical map construction involves making a shotgun clone bank and "fingerprinting" randomly chosen clones by deriving information about restriction fragment lengths. The resulting fingerprints are compared and used as the basis to order an overlapping assemblage of clones. This assemblage is known as a "contig" [Staden, 1980], which corresponds to an "island" with at least two clones [Lander and Waterman, 1988].

Recently, efforts have been successful in making contig maps of the genomes of *E. coli* [Kohara *et al.*, 1987; Knott *et al.*, 1989], *B. subtilis* [Itaya and Tanaka, 1991], *C. elegans* [Coulson *et al.*, 1986] and *S. cerevisiae* [Olson *et al.*, 1986]. Each research program generated fingerprints in slightly differing ways, and produced maps based on the observed overlap of fingerprints in individual clones. As researchers focus attention on eukaryotic genomes, however, the combinatorial problem of contig alignment will pose a serious challenge for map assembly.

Although several methods have been proposed for physical map reconstruction, a fully automated method for map assembly remains problematical. Branscomb has suggested the use of statistical methods for contig assembly [Branscomb and Nelson, 1990]. Olson *et al.* [Olson *et al.*, 1986] have described a tree-searching algorithm which seeks restriction maps that are consistent with the fragment-size lists for complete digests of clones. Most of the emphasis, however, has been on detecting matches between pairs of clones [Myers and Huang, 1992].

Constructing a physical map from fingerprinted DNA fragments is not solved by conventional string-matching algorithms. Several related difficulties, both theoretical and practical, hinder such approach:

1. The information resolution of fingerprints is intrinsically limited by the fact that only a

small proportion of sequence information is sampled via restriction sites. This causes the number of solutions for a typical problem to grow exponentially in the number of clones.

2. The physical resolution of electrophoretic fingerprints may reflect errors of up to 5% of the fragment size. Therefore, uncertainty must be accounted for in the algorithm.

We propose an algorithm which constructs a physical map of DNA from a fingerprinted library of random clones. This approach provides a general and efficient means for constructing long-range physical maps in large-scale genome mapping. We have developed a representation and a method for alignment and comparison of incomplete sequences, and an error model to handle certain types of uncertainties. Section 2 will describe the method and the error model in detail, and a formal analysis is provided in section 3.

2 Methodology

In this section we describe the class of map construction problems that the system solves, the solution description language, and the problem solving approach.

2.1 The Problem

The kinds of problems that the system solves are:

Given

n clone libraries of genome or region of genome produced in the following fashion:

- Different libraries are obtained by partial digest of DNA with different restriction enzymes.
- The order of clones within a library is unknown.
- The approximate length of each clone is known.
- We can further determine a restriction map for each clone.

Find

a physical map for the molecule, consisting of overlapping sets of clones

The system is also able to randomly generate a set of DNA fragments corresponding to a hypothetical genome and construct a DNA map from that library.

Given

specifications for such a library (length of a DNA sequence, frequencies of bases, restriction enzymes, number of samples, number of clones, length of clones and fragments, etc.)

Find

- a library of random clones
- an ordered set of overlapping clones from that library (i.e. map)

The purpose of the hypothetical genomic library is to simulate several map-building strategies on that library and compare them. This has some advantages over a formal mathematical analysis because a simulation properly designed can show the effects of changing various parameters on map-building process without making too many simplifying assumptions [Sirotkin and Loehr, 1990]. Furthermore, the code used in simulation can be used on actual data.

2.2 Simplifying Assumptions

We have made the following assumptions to simplify the problem:

- The error in the measurement of the length of clones and fragments is proportional to the length and the error rate is less than 5%.
- The genomic library to be processed by the system represents unbiased samples from the genome.

  The task of creating a map requires sufficient clones to cover the entire genome. However, an unnecessarily large clone library will cause the system to do fruitless work. Lander and Waterman [Lander and Waterman, 1988] demonstrate that the library size required to produce a map is dependent on the minimum detectable overlap.

2.3 Solution Description Language

Typical string matching algorithms or sequence comparison/alignment algorithms are inapplicable or inadequate for this problem because: (1) most string matching algorithms require that

the complete sequence be known, which is not possible in this problem; (2) the size of alphabet in this problem is small (4 characters); (3) the enzyme recognition patterns upon which matches are determined are very short (typically 4, 6, or 8 characters) compared to the sequence.

We have developed a description language which facilitates alignment and comparison of incomplete sequences and an alignment method which detects an overlap of clones in linear time of the number of their restriction sites. The language is used both as the problem specification language and solution description language. The class of problems mentioned above is represented in the language and accepted as input to the system. The final DNA map constructed is expressed in the language together with a visual map. The following notations will be used to describe the basic solution objects and their relations.

1. $S_i = \{c_{i1}, c_{i2}, \ldots, c_{im}\}$
   S_i is a library of m clones.

2. $c_{ij} \in S_i$
   c_{ij} is a clone of a library S_i.

3. $c_{ij} = (f_1 \ldots f_k)$, or $c_{ij} = ((s_1 l_1 r_1) \ldots (s_k l_k r_k))$
   c_{ij} consists of fragments $f_1 \ldots f_k$. s_i, l_i, and r_i are the size, left end, and right end of a fragment f_i.

4. $I_k = (((c_{ij} p_a p_b)(c_{kl} p_c p_d)) \ldots ((c_{kl} p_e p_f)(c_{mn} p_g p_h)))$
   I_k is an island consisting of overlap of c_{ij} with c_{kl} and c_{kl} with c_{mn}. $p_a, \ldots, p_h$ represent overlapping portions in the clones.

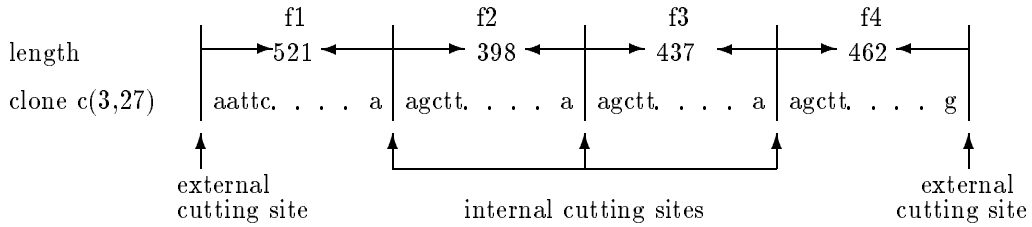
5. enzyme$(S_i) = ((e_1, \ldots, e_n)\ (e'_1, \ldots, e'_m))$
   Enzymes $e_1, \ldots, e_n$ are used to create the clones of a library S_i and enzymes $e'_1, \ldots, e'_m$ are used to cut the clones internally. $(e'_1, \ldots, e'_m)$ may be an empty list.

6. enzyme$(c_{ij}) = ((e_1, \ldots, e_n)\ (e'_1, \ldots, e'_m))$
   Enzymes $e_1, \ldots, e_n$ are used to create a clone c_{ij} and enzymes $e'_1, \ldots, e'_m$ are used to cut the clone internally.

7. length(c_{ij}) is the length of a clone c_{ij}.

8. length(f_j) is the length of a fragment f_j.

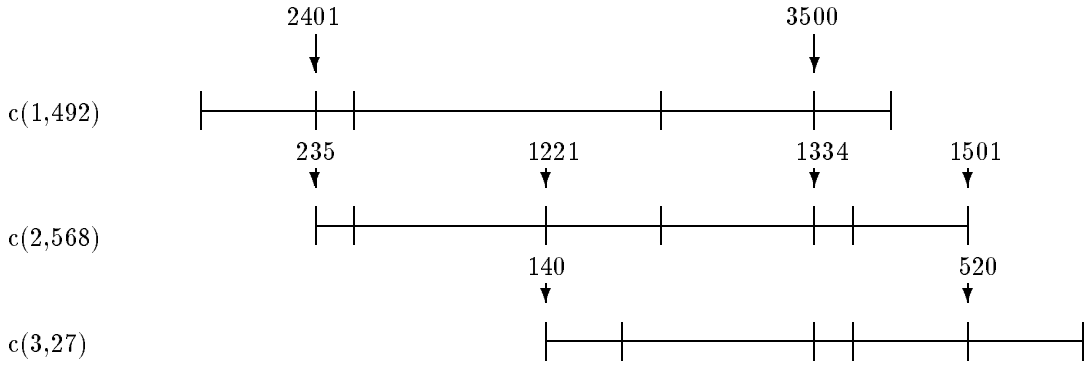
Suppose that a clone of 1818 bases is obtained with *Eco*RI and the clone is further cut into subfragments with *Hin*dIII as follows, where dots represent unknown bases. A clone is represented as an ordered list of triples where each triple corresponds to a fragment. A triple consists of size, left end, and right end of a fragment. The representation explicitly includes the sequence of cleavage sites of restriction enzymes instead of the names of restriction enzymes. This is because some restriction enzymes have the uncertainties in their recognition sites (e.g., gccnnnnnggc of Bgl1) and the name itself is not enough for matches. In the following example, clone c(3,27) has two external cleavage sites, three internal cleavage sites, and four fragments (f1, f2, f3, f4). c(3, 27) is just an identifier of a clone, meaning that it is the 27th clone of a library S_3.



The clone c(3, 27) will be represented as

```
c(3,27) = ((521 aattc a) (398 agctt a) (437 agctt a) (462 agctt g))
```

Suppose it is determined that the sequence from 2401 to 3500 of clone c(1,492) overlaps the sequence from 235 to 1334 of clone c(2,568) and the sequence from 1221 to 1501 of clone c(2,568) overlaps the sequence from 140 to 520 of clone c(3,27).



Then, the island is represented as a list of unordered pairs where each pair corresponds to an overlap between two clones.

```
(((c(1,492) 2401 3500) (c(2,568) 235 1334)) ((c(2,568) 1221 1501) (c(3,27) 140 520)))
```

2.4 Problem Solving Approach

When two clones A and B are compared, there are six possible relations between them (see Figure 1). During the alignment, we slide one clone forward against the other examining every possible restriction site alignment so as not to miss any potential overlap. On the other hand, the clone is never moved backward but is jumped forward as much as possible whenever it is considered safe in order to speed up the matching process. This is similar to the Boyer-Moore algorithm for string matching [Baase, 1988]. Several heuristics can guide which clones are to be considered first, which sites of clones are to be examined, and how much can be jumped during alignment.

Figure 1

The criteria for overlaps between clones should be strict enough so that the system produces few false positives. However, if the criteria are too stringent, we will miss many potential matches. According to Lander and Waterman's analysis [Lander and Waterman, 1988], clones which share at least 2 fragments are expected to have an acceptable low rate of false positive matches. This corresponds to an overlap of 20%. If a six-base-recognition enzyme is used, recognition sites for the enzyme would appear once in every $4^6 = 4$ Kbp and a clone of $80 - 100$

Kbp long would have about 20 – 25 fragments (19 – 24 internal cleavage sites). 20% of 20 – 25 fragments (19 – 24 internal cleavage sites) is roughly equal to 4 – 5 fragments or cleavage sites. Hence, our algorithm uses 4 cleavage sites as a cutoff value for claiming an overlap between clones.

Once a pair of overlapped clones is found, their superimposed sequence can be used in subsequent overlapping. For example, once A and B are found to overlap, the merged sequence C is:

```
A:  atgca.....c|  tgcat.....g|tgcat...g
B:                tgcat.....g|tgcat...g|tgcat...a
C:  atgca.....c|  tgcat.....g|tgcat...g|tgcat...a
```

Iterative pairwise alignment is often used for multiple sequence alignment and comparison. Though the iterative pairwise alignment is useful, it has a problem in that the final alignment is dependent on the choice of the pairwise alignments; for example, once an alignment among the first few sequences is decided, the alignment can never be modified with the addition of further sequences. Furthermore, there is no guarantee that the early matched sequences will be correctly aligned with subsequent sequences. We suggest a hierarchical method for multiple alignment and comparison, which is similar to the hierarchical agglomerative clustering method used in pattern recognition [Duda and Hart, 1973]. It is not completely order independent, but mitigates the problems mentioned above.

Figure 2 shows a global organization of tasks which must be accomplished by a map construction system. Tasks are represented in rectangles, and user-supplied inputs and outputs of the system are represented in ovals. Tasks in module A are for generating a hypothetical genomic library from a specification. The specification can be supplied by a user or a system, and the generated library is represented in an internal form required by module C. Module B is simply for representing clones in internal representation. Module C actually builds a map from a given library by detecting overlapping clones and clustering them into a contiguous segment. The tasks in module D process the map built in C for displaying the map in graphics or other forms. Figure 3 is a rough description of map building processes of the system (module C in Figure 2). **Figure 2** **Figure 3**

For a given clone of a reference library ($s(1)$ in Figure 3), clones of other libraries are examined and the best matching clone is selected in each library. If the best matching clone has not been attached to another island, it is appended to the current island. If it has already been attached to another island, an attempt is made to merge the two islands with respect to the common clone. If the two islands can be linked without violating consistency, they are merged together. If the two islands cannot be merged, the clone is appended to the better matching island.

Task AlignClone tries every possible alignment of two clones and returns the best alignment position in the following way. Suppose we are looking for the best matching clone from the set S_j for the clone c_{ik} and the current candidate clone is c_{jl}. An alignment of the two clones is done by lining up at least one common cleavage site. How much the clone c_{jl} can be jumped against c_{ik} depends on the enzyme(S_i), enzyme(S_j), and the sizes of current fragments compared. Initially the two clones are lined up with respect to the last fragment of c_{jl} and the first fragment c_{ik}. To find the best alignment of clone c_{ik} and clone c_{jl}, c_{jl} is moved forward against c_{ik} (see Figure 4). **Figure 4**

In practice, many of the alignments in Figure 4 will be skipped. Since we always compare clones which share restriction enzymes, only certain alignments need be tried:

1. If enzyme(S_i) = ((A) (B)) and enzyme(S_j) = ((B) (A)), line up the terminal ends (external cleavage sites) of c_{ik} with internal cleavage sites of c_{jl} or the internal cleavage sites of c_{ik} with the terminal ends of c_{jl}.

2. If enzyme(S_i) = ((A) (B)) and enzyme(S_j) = ((A B) (C)), line up the terminal ends of c_{jl} with terminal or internal cleavage sites of c_{ik}.

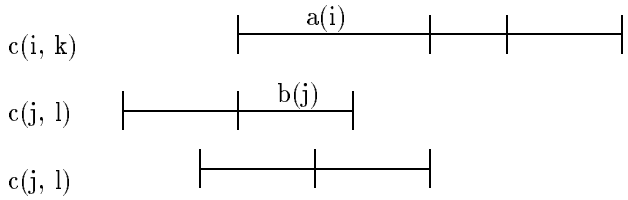
3. If enzyme(S_i) = ((A) (C)) and enzyme(S_j) = ((B) (C)), line up the internal cleavage sites of c_{ik} with the internal cleavage sites of c_{jl}.

4. If enzyme(S_i) = ((A B) (C)) and enzyme(S_j) = ((A) (B)), line up the terminal ends of c_{ik} with terminal or internal cleavage sites of c_{jl}.

Let a_i and b_j be the reference fragments of clone c_{ik} and c_{jl} of the last alignment. That is, the last alignment was done by lining up a_i and b_j. For the next alignment, move clone c_{jl} in

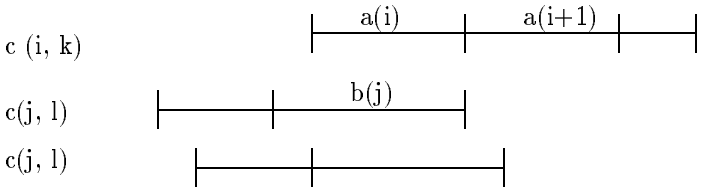
the following way:

1. If $a_i > b_j$, move c_{jl} to the right by [length(a_i) - length(b_j)].

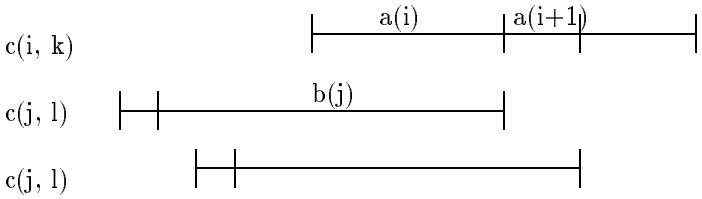


2. If $a_i < b_j$,

   a) if length(b_j) – length(a_i) < length(a_{i+1}), move c_{jl} [length(b_j) – length(a_i)] bases to the right.

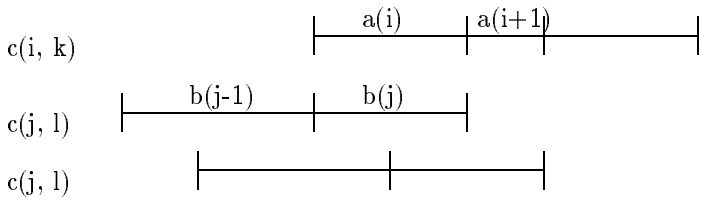


   b) if length(a_{i+1}) < length(b_j) – length(a_i), move c_{jl} to the right by length(a_{i+1}).

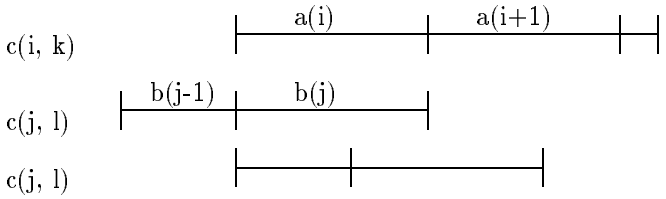


3. If $a_i = b_j$

   a) length(a_{i+1}) < length(b_{j-1}), move c_{jl} to the right by length(a_{i+1}).



b) length(b_{j-1}) < length(a_{i+1}), move c_{jl} to the right by length(b_{j-1}).



Initially the two clones are lined up with respect to the last fragment of c_{jl} and the first fragment c_{ik} (which we call reference fragments). An alignment score is calculated by walking backward from the reference fragments, and by checking if the bases around the cleavage site match each other. The portion to the right of the reference fragments need not be examined since they are not overlapped. If there is a mismatch between two known bases at the cleavage site (i.e., both clones have a cleavage site at the position), further comparison is stopped and another alignment is tried. If it is a comparison between known bases and unknown bases (i.e., only one clone has a cleavage site at the position), that comparison does not contribute anything to alignment score. If there is a match between two known bases, the total score is increased for the alignment. Therefore, the score for an alignment is simply the number of cleavage sites whose positions and bases match. Mismatch between two known bases contributes an infinitely negative score so that the alignment with mismatch is not examined further. When a best matching clone is selected among others from a set, we claim an overlap only if the alignment score of the clones is at least 4.

If there is more than one disjoint island, the system examines boundary clones of islands and checks if they can be linked. If islands cannot be linked unambiguously with existing information, the system either requests more detailed restriction maps of boundary clones or just gives the incomplete map.

2.5 Error Model

Constructing a physical map from DNA fragments might look as simple as conventional string matching problems. However, the number of solutions to this problem grows exponentially in the number of clones involved, and uncertainties due to experimental errors make the problem more complicated. Although there exists only one true map for a given DNA molecule, the information from restriction enzyme digest may suggest many consistent maps, particularly when experimental errors are considered [Dix and Kieronska, 1988; Allison and Yee, 1988]. Sources of error in digest data include: (1) errors in the measurement of fragment lengths, (2) fragments of similar size, (3) missing fragments, (4) incomplete digestion.

Since errors can be accumulated as the map is built, we use rigorous error bounds for fragment lengths. We assume measurement errors of fragment lengths are within 5% of the length of the fragment being aligned. Hence, when we compare a fragment of length l_1 with a fragment of length l_2, we consider the two fragments match in their lengths if there exist l_1' and l_2' such that

$$l_1' = l_2'$$

$$(1 - 0.05) \times l_1 \leq l_1' \leq (1 + 0.05) \times l_1$$

$$(1 - 0.05) \times l_2 \leq l_2' \leq (1 + 0.05) \times l_2$$

For the matching of bases around cleavage sites, we require exact matching. We scan aligned clones to get an alignment score (i.e., we visit each cleavage site backward from the reference fragments), and check if the bases around cleavage sites match each other. If there is a mismatch between two known bases at the cleavage site (i.e., both clones have a cleavage site at the position), further comparison is stopped and another alignment is tried. If it is a comparison between known bases and unknown bases (i.e., only one clone has a cleavage site at the position), that comparison does not contribute anything to alignment score. If there is a match between two known bases, the total score is increased for the alignment. Therefore, the score for an alignment is simply the number of restriction sites whose positions and bases match. Mismatch between two known bases contributes an infinitely negative score so that the alignment with mismatch terminates as soon as possible.

When a best matching clone is selected among others from a set, we claim an overlap if the alignment score of the clones is at least 4. This number looks like an arbitrary choice, but was

actually determined based on Lander and Waterman's analysis [Lander and Waterman, 1988] as mentioned earlier.

3 Evaluation

Efficiency improvements in our approach come from:

1. We consider constraints in each task, which intelligently reduces the search space.
2. Clones are compared only at their cleavage sites.
3. Disjoint islands are merged together while clones are attached to an island.

For the purpose of efficiency analysis, we assume that the algorithm is given total mn clones (m clones from each of n libraries) and that the maximum length of of a clone is l. (a library with less than m clones can be padded with empty strings.)

$$S_1 = \{c_{11}, c_{12}, \ldots, c_{1m}\}$$

$$S_2 = \{c_{21}, c_{22}, \ldots, c_{2m}\}$$

$$\ldots$$

$$S_n = \{c_{n1}, c_{n2}, \ldots, c_{nm}\}$$

In general, comparing m sequences of length l takes $O(l^m)$ time. Finding the best matching clone for c_{1j} from a library S_i will take $O(l^m)$ time since all the m clones of S_i should be compared with c_{1j}. Finding the best matching clone for c_{1j} from each of the $n-1$ libraries will take $O(nl^m)$ time. Since we have m clones in the library S_1, the total time to find the best matching clone for every clone of S_1 from each library takes $O(mnl^m)$ time.

In our approach, comparing two clones can be done in $O(k)$ where k is the number of cleavage sites of the two clones. Recall that $k \ll l$. Finding the best matching clone for c_{1j} from a library S_i will take $O(km)$ time. Finding the best matching clone for c_{1j} from each of the $n-1$ libraries will take $O(kmn)$ time. The total time for finding the best matching clone for every clone of S_1 from each library will take $O(km^2n)$ time.

When does our algorithm outperforms the above method?

$$O(km^2n) < O(mnl^m)$$

After factoring out mn from both terms

$$O(km) < O(l^m)$$

$$km = o(l^m)$$

Since $km = o(l^m)$, our algorithm is always better than the method mentioned above for any value of m, l, and k. Then, the next question is if the method has an optimal efficiency. Assuming that each clone has k cleavage sites in it, it should take at least $\Omega(kmn)$ time to find overlapping sets because it should examine every cleavage site of every clone at least once. Therefore, $O(km^2n)$ may not be optimal. However, we do not know yet if $\Omega(kmn)$ is the achievable lower bound to this problem. We should also be aware that the complexity analysis given above is the worst case complexity. The average complexity may be much lower.

4 Conclusion and Future Research

We have described a strategy for rapidly building a physical map of a large number of random clones. Using this strategy, many parts of current process of physical map construction can be automated. We did not assume the terminal clones of the genome are known. However, if this information is provided, the performance of map building process can be improved.

Currently we use a simple error model, where only a few uncertainties associated with measurement errors and ambiguities of recognition sites of restriction enzymes are considered. The value itself for error bounds can be changed (e.g., from 5% to 3%) easily, but more work (including an extensive testing on actual/generated data) remains to be done to get a more accurate error model which can also handle other types of uncertainties. For the system to be practical, actual implementation issues at the program level would be important, too. Since any system will deal with a huge amount of data, efficiency will be greatly affected by implementation.

Although a current implementation uses the representation described in section 2.3, a sequence can be represented in a simple numeric coding scheme for efficiency purposes. Each

individual base of a sequence is given a rank b_i and the sequence is represented by a vector $\mathbf{b} = (b_1 \ldots b_n)$. The vector is then mapped to an integer $S(\mathbf{b})$ as follows:

$$S(\mathbf{b}) = \sum_{i=1}^{n} (b_i - 1)4^{i-1} + 1$$

where g = 1, t = 2, a = 3, and c = 4.

Dumas and Ninio [Dumas and Ninio, 1982] use a similar coding scheme, but their coding scheme has a problem that there is an ambiguity among sequences of g's of different lengths. For example, g = 1, gg = 1, ggg = 1, and ggg ... g = 1. We can use a slightly different coding scheme which associates a different sequence with a unique integer.

$$S(\mathbf{b}) = \sum_{i=1}^{n} b_i \cdot 4^{n-i}$$

where a = 1, c = 2, g = 3, and t = 4. For example, the following clone with 4 internal fragments

```
((520 tcgac a) (489 agctt a) (634 agctt a) (453 agctt g))
```

will be represented as

```
((520 1206 1) (489 500 1) (634 500 1) (453 500 3))
```

where

tcgac $= 4 \cdot 4^4 + 2 \cdot 4^3 + 3 \cdot 4^2 + 1 \cdot 4^1 + 2 \cdot 4^0 = 1206$

agctt $= 1 \cdot 4^4 + 3 \cdot 4^3 + 2 \cdot 4^2 + 4 \cdot 4^1 + 4 \cdot 4^0 = 500$

a = 1

g = 3

Using this coding scheme does not change the asymptotic complexity of $O(km^2n)$, but the constant factor of $O(km^2n)$ becomes smaller. Comparison of bases around each cleavage site is done only once instead of s times where s is the number of bases around a cleavage site, which is usually 4, 6, or 8. Bit-wise operation on the number can allow even faster manipulation.

5 References

Allison, L. and Yee, C. N. (1988) Restriction site mapping is in separation theory. *CABIOS*, 4(1):97–101.

Baase, S. (1988) *Computer algorithms: introduction to design and analysis.* Addison-Wesley Pub. Co., Reading, Mass.

Bell, G. I. (1990) The human genome: An introduction. In *Computers and DNA: the proceedings of the Interface between Computation Science and Nucleic Acid Sequencing Workshop*, Addison-Wesley Publishing Company, 7:3–19.

Branscomb, E. W. and Nelson, D. O. (1990) Statistical Considerations of Physical Mapping by Contig Assembly. In *Proceedings of the Second Conference on Mathematical Approaches to DNA.*

Committee on Mapping and Sequencing the Human Genome (1988) *Mapping and Sequencing the Human Genome.* National Academy Press, Washington, D.C.

Coulson, A., Sulston, J., Brenner, S., and Karn, J. (1986) Toward a physical map fof the genome of the nematode, *Caenorhabditis. Proc. Natl. Acad. Sci. USA*, 83:7821–7825.

Dix, T. I. and Kieronska, D. H. (1988) Errors between sites in restriction site mapping. *CABIOS*, 4(1):117–123.

Duda, R. O. and Hart, P. E. (1973) *Pattern classification and scene analysis.* Wiley, New York.

Dumas, J. and Ninio, J. (1982) Efficient algorithms for folding and comparing nucleic acid sequences. *Nucleic Acids Research*, 10(1):197–206.

Itaya, M. and Tanaka, T. (1991) Complete Physical Map of the *Bacillus subtilis* 168-Chromosome Constructed by a Gene-Directed Mutagenesis Method. *Journal of Molecular Biology*, 220: 631–

648.

Karlin, S., Ost, F., and Blaisdell, B. E. (1989) Patterns in dna and amino acid sequences and their statistical significance. In *Mathematical Methods for DNA Sequences*, CRC Press, Inc., pp. 133–158.

Kohara, Y., Akiyama, K., and Isono, K. (1987) The Physical Map of the Whole E. coli Chromosome: Application of a New Strategy for Rapid Analysis and Sorting of a large Genomic Library. *Cell*, 50:495–508.

Knott, V., Blake, D. J., and Brownlee, G. G. (1989) Completion of the detailed restriction map of the *Escherichia coli* chromosome by the isolation of overlapping cosmid clones. *Gene*, 17: 5901–5912.

Lander, E. S. (1989) Analysis with restriction enzymes. In *Mathematical Methods for DNA Sequences*, CRC Press, Inc. pp. 35–52.

Lander, E. S. and Waterman, M. S. (1988) Genomic Mapping for Fingerprinting Random Clones: A mathematical Analysis. *Genomics*, 2:231–239.

Myers, E. W. and Huang, X. (1992) An $O(N^2 log N)$ Restriction Map Comparison and Search Algorithm. *Bulletin of Mathematical Biology*, 54(4):599–618.

Olson, M. V., Dutchik, J. E., Graham, M. Y., Brodeur, G. M., Helms, C., Frank, M., MacCollin, M., Scheinman, R., and Frand, T. (1986) Random-clone strategy for genomic restriction mapping in yeast. *Proc. Natl. Acad. Sci. USA*, 83:7826–7830.

Sirotkin, K. and Loehr, J. J. (1990) Simulation and analysis of physical mapping. In *Computers and DNA: the proceedings of the Interface between Computation Science and Nucleic Acid Sequencing Workshop*, Addison-Wesley Publishing Company, 7:241–257.

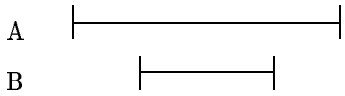
Staden, R. (1980) A new computer method for the storage and manipulation of DNA gel reading

data. *Nucleic Acids Research*, 8:3673–3694.

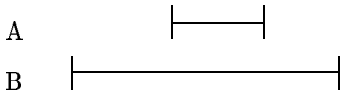
Stephens, J. C., Cavanaugh, M. L., Gradie, M. I., Mador, M. L., and Kidd, K. K. (1990) Mapping the human genome: Current status. *Science*, 250:237–244.

Tavare, S. and Giddings, B. W. (1989) Some statistical aspects of the primary structure of nucleotide sequences. In *Mathematical Methods for DNA Sequences*, CRC Press, Inc., pp. 117–132.

1. perfect overlap

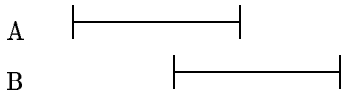


subfragment (A) = B

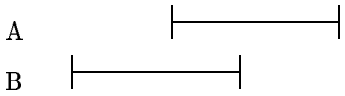


subfragment (B) = A

2. partial overlap

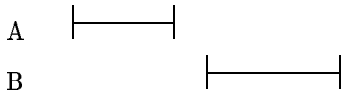


suffix (A) = prefix (B)

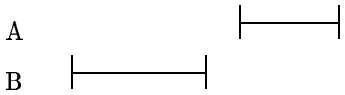


suffix (B) = prefix (A)

3. no overlap

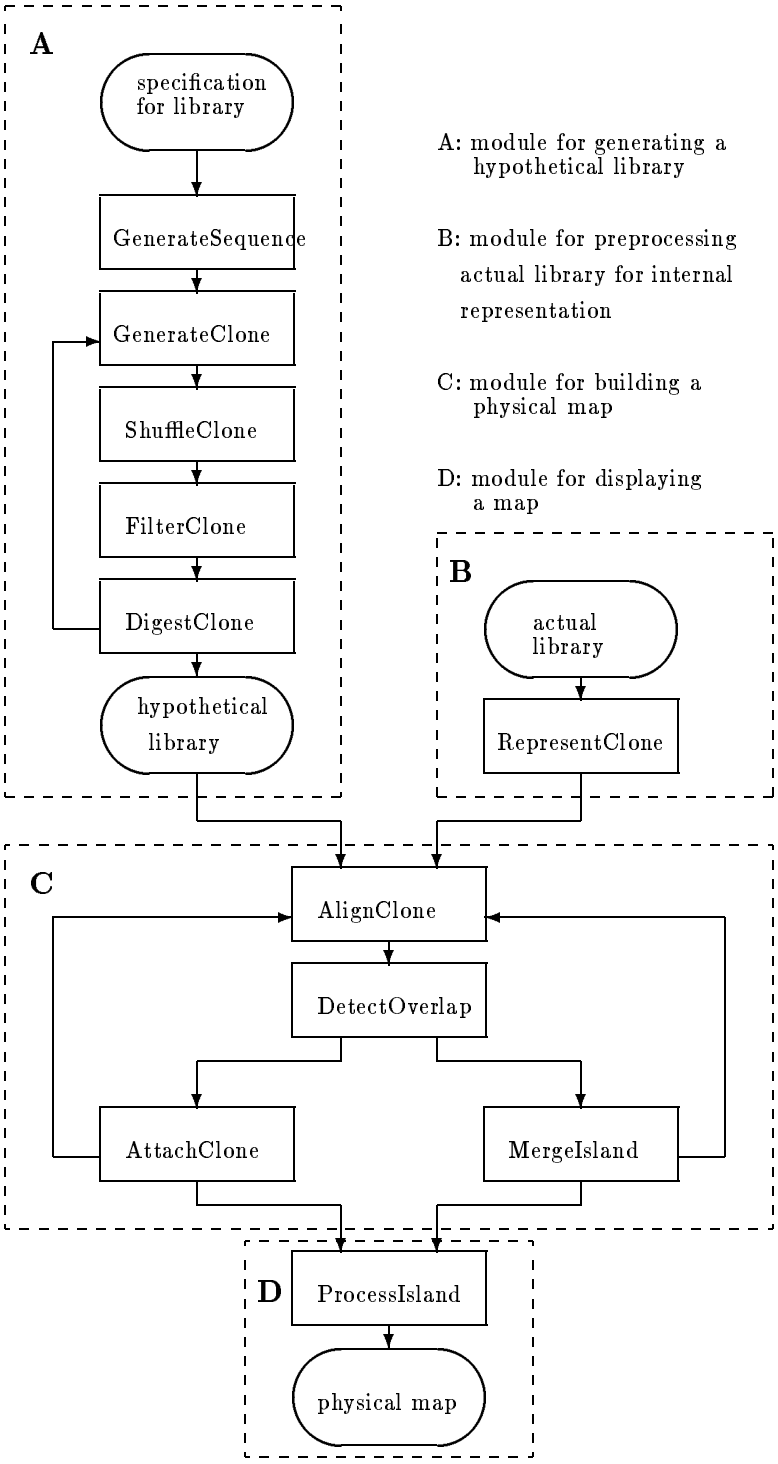


A precedes B



B precedes A

Figure 2


A
specification for library
GenerateSequence
GenerateClone
ShuffleClone
FilterClone
DigestClone
hypothetical library
A: module for generating a hypothetical library
B: module for preprocessing actual library for internal representation
C: module for building a physical map
D: module for displaying a map
B
actual library
RepresentClone
C
AlignClone
DetectOverlap
AttachClone
MergeIsland
D
ProcessIsland
physical map

```
for each clone c(1,i) in library s(1)
    island(i) = MakeIsland(c(1,i))
    for each library s(j)
        if ExistsCommonEnzyme(island(i), s(j))
            for each clone c(j,k) in library s(j)
                score(k) = AlignClone(c(1,i), c(j,k))
            bestmatch = SelectClone(score,k)
            if UnAttached(bestmatch)
                island(i) = AttachClone(island(i), bestmatch)
            else if Attached(bestmatch) and Consistent(island(i),islandof(bestmatch))
                island(i) = MergeIsland(island(i), islandof(bestmatch))
            else
                AttachClone(SelectIsland(island(i), islandof(bestmatch)), bestmatch)

while more than one disjoint islands
     foreach island(i) and island(j)
        if Consistent(island(i),island(j))
          MergeIsland(island(i), island(j))
```

Figure 4

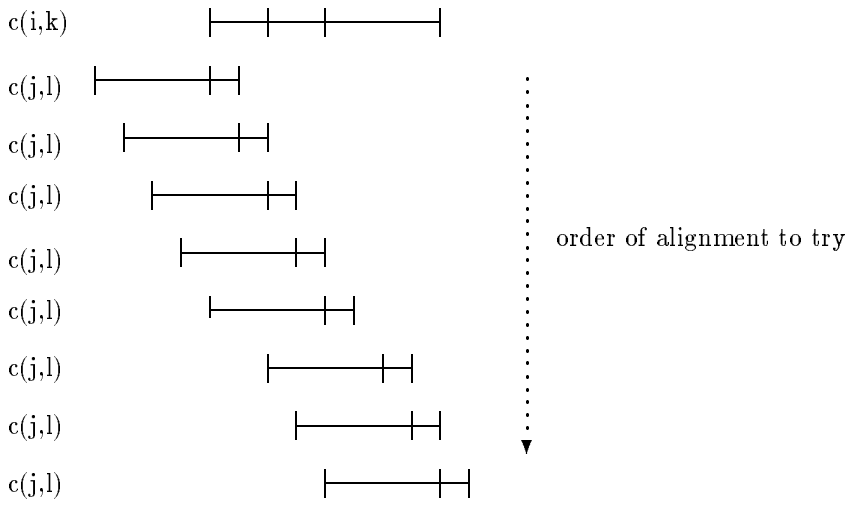


Figure Legends

Figure 1 overlapping relations between two clones

Figure 2 organization of tasks

Figure 3 Top-level tasks for module C in figure 2

Figure 4 When clone c(j,l) is aligned with clone c(i,k), c(j,l) is always moved forward with respect to c(i,k) so that at least one common cleavage site can be lined up.