A Partial Digest Approach to Restriction Site Mapping (Extended Abstract)

Steven S. Skiena*
Dept. of Computer Science
State University of New York
Stony Brook, NY 11794

Gopalakrishnan Sundaram
Dept. of Applied Mathematics
State University of New York
Stony Brook, NY 11794

Abstract

We present a new practical algorithm to resolve the experimental data of restriction site analysis, which is a common technique for mapping DNA. Specifically, we assert that multiple digests with a single restriction enzyme can provide sufficient information to identify the positions of the restriction sites with high probability. The motivation for the new approach comes from combinatorial results on the number of mutually homeometric sets in one dimension, where two sets of n points are homeometric if the multiset of $\binom{n}{2}$ distances they determine are the same.

Since experimental data contains error, we propose algorithms for reconstructing sets from noisy interpoint distances, including the possibility of missing fragments. We analyze the performance of these algorithms under a reasonable probability distribution, establishing a relative error limit of $r = \Theta(1/n^2)$ beyond which our technique becomes infeasible. Through simulations, we establish that our technique is robust enough to reconstruct data with relative errors of up to 7.0% in the measured fragment lengths for typical problems, which appears sufficient for certain biological applications.

*This work was partially supported by NSF Research Initiation Award CCR-9109289 and by New York Science and Technology Foundation grant RDG-90171.

Introduction

The genetic code of an organism defines a string on the alphabet $\{A, C, G, T\}$, representing the amino acids adenine, cytosine, guanine, and thymine. Restriction enzymes are chemicals which recognize and cut DNA molecules at particular patterns. For example, the enzyme *Eco*RI cuts at the pattern *GAATTC*. Exposing a DNA molecule to a particular restriction enzyme may cut it in one or more places, and measuring the length of the fragments provides information about where the restriction sites are located on the molecule. Reconstructing this information is called *restriction site analysis* and is a common laboratory technique.

Reconstruction can be difficult whenever an enzyme cuts the molecule in more than one place, which is usually the case for sufficiently long molecules. If we expose a linear molecule to an enzyme and wait for the reaction to complete, each of the n restriction sites are cut and so we obtain $n + 1$ fragments. Such an experiment is known as a *full* or *complete digest*. Any permutation of the fragments resulting from a complete digest is a possible interpretation of the molecule, and so further data is necessary to position the sites. To get additional information, typically multiple enzymes are used. Unfortunately, there is no way to guarantee that some constant number of enzymes k will be sufficient for unique reconstruction, since any unbroken run of three or more sites for a particular enzyme cannot be resolved.

If our sample contains a large number of identical linear molecules, as is obtainable by cloning, and we reduce the time of exposure so that not every site is cut, we would obtain instances of each of the $\binom{n}{2}$ fragments, correspond to fragments between every pair of sites. Such an experiment is called a *partial digest*. Finding the sites from the interpoint distances between pairs of sites, is known in the literature as the *turnpike reconstruction problem*. The combinatorial results of Skiena, Smith,

and Lemke [12], presented in Section suggest that only carefully constructed examples permit multiple reconstructions consistent with the results of a single partial digest. Thus only one partial digest should usually suffice for reconstruction.

The combinatorial results of [12] assume that the exact length of each fragment is presented, an entirely unrealistic assumption for experimental data. In practice, the lengths of DNA fragments are measured by means of differing forced diffusion speeds in gel electrophoresis. The observed length of each fragment may have significant relative error. Currently, 2% to 5% relative error is achievable [3]. Further, two fragments of approximately equal length are unlikely to be discriminated on the gel, so the multiplicities of distances are almost certain to be lost. Additional complications arise with small fragments, which can run off the end of the gel, and so be lost. Finally, on any particular molecule certain sites are less likely to be cut than other sites, so certain fragments might not occur with sufficient frequency to be measured.

Despite these complications, we believe that the combinatorial constraints inherent in partial digest data are sufficiently strong to facilitate reconstruction. To explore this idea, we have generalized the backtracking algorithm of [12] to deal with noisy data, both imprecise measurements as well as missing fragments. In this paper, we present our generalized algorithm for noisy data, analyze its performance, as well as results of experiments to assess its behavior.

We emphasize that the practical instances of this problem are quite small. With more than 5-10 restriction sites, it becomes increasing difficult to distinguish the resulting fragments with electrophoresis. The time complexity of an algorithm is much less important than minimizing the number of possible interpretations for the molecule, as a function of the number of experiments. By using partial digests, even with missing fragments, our methods perform extremely well.

We first discuss related work, both previous algorithms for restriction site analysis and the combinatorial results which support the partial digest approach. Then we present a backtracking algorithm for reconstructing sets from noisy data and missing fragments. Later, we analyze the performance of this algorithm on noisy data establishing the relative error limit of $r = \Theta(1/n^2)$ where our technique becomes infeasible. Finally, we present the results of experiments with an implementation of this algorithm.

Previous Work

Determining the location of the sites from restriction site data is a difficult algorithmic problem, which has been extensively studied [1, 2, 4, 5, 11, 9, 14, 16]. These approaches attempt reconstruction by using multiple complete digests, as opposed to the single enzyme method proposed in this paper. Fragment lengths from incomplete digests have been exploited by Sutherland and Partis [15] for the related problem of mapping antibody binding sites on a protein. They use linear programming techniques as opposed to our combinatorial search methods.

Our approach is based on the theory of homeometric sets [10]. The problem of reconstructing sets from interpoint distances dates back to the origins of X-ray crystallography in the 1930's [8, 7]. Two noncongruent n-point sets are *homeometric* if the multisets of $\binom{n}{2}$ distances they determine are the same. Let the function $H(n)$ denote the maximum possible number of mutually noncongruent and homeometric n-point sets which can exist in one dimension. For example, {0,2,3,5,7,9,10,13,16,17,18,22,28} and {0,2,7,9,13,14,15,17,18,19,22,25,28} are two homeometric sets on 13 points. Skiena, Smith and Lemke [12] show that

$$\frac{1}{2} \cdot n^{0.8107144} \leq H(n) \leq \frac{1}{2} \cdot n^{1.2324827}.$$

This fact demonstrates that incomplete digests dramatically reduce the number of interpretations of a complete digest.

The Reconstruction Algorithm

The polynomial factorization approach does not generalize to noisy data, since it would require a concept analogous to an approximate factorization. However, Skiena, Smith, and Lemke [12] presented a backtracking algorithm which, although it takes $O(2^n \cdot n \log n)$ time in the worst-case, generalizes readily into a practical algorithm for noisy data.

Here we review the backtracking algorithm. Let d_{ij} represent the distance between the ith and jth points on a line, ordered from left to right. The complete set of distances and their initially unknown assignments can be represented as a triangular matrix or *pyramid*. The set of distances d_{ij} such that $j - i = l$ defines row l of the pyramid. For convenience, we shall represent the input distances or *fragments* by a vector v sorted in increasing order, so that $v[i] \leq v[j]$, $1 \leq i < j \leq \binom{n}{2}$. A solution is an satisfactory assignment of the input distances to the positions of the pyramid. For a

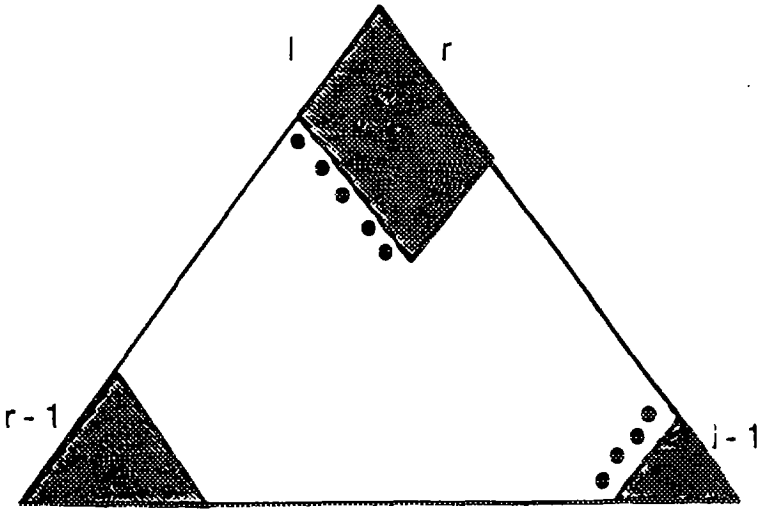

Figure 1: The Effect of Choosing $d_{1(n-l+1)}$.

solution, an input fragment is called a *base* fragment if it goes into the first row of the pyramid, and is called an *intermediate* fragment otherwise. The positions d_{1i}, $1 < i \leq n$ are called the *left* hand side of the pyramid and the positions d_{jn}, $1 \leq j < n$ are called *right* hand side of the pyramid.

The backtracking procedure repeatedly position the *largest remaining* unpositioned distance. Because we are placing the largest available distance in the pyramid, there will be only two possible locations to choose from. The key to making this procedure efficient is to immediately fill in the pyramid any values which are determined by our previous (non-deterministic) selections.

Assume that we have thus far placed l elements along the left-hand side of the pyramid and r on the right, as shown in Figure 1. All the positions in the shaded regions are determined by the $l+r-1$ distances d_{1j} and d_{in}, $(n-l+1) \leq j \leq n$, $1 \leq i \leq r$. The largest remaining distance must be associated with either $d_{1(n-l)}$ or $d_{(r+1)n}$.

Suppose that the backtrack procedure selects the left side to receive the next element. Thus $d_{1(n-l)}$ is assigned the largest remaining distance. This determines $d_{i(n-l)}$, $2 \leq i \leq r$, since $d_{i(n-l)} = d_{1(n-l)} - d_{1i}$, which are already in the pyramid, as well as $d_{(n-l+1)(n-j)}$, $0 \leq j \leq l-2$, since $d_{(n-l+1)(n-j)} = d_{1n} - d_{1(n-l+1)} - d_{(n-j)n}$. If any of these $l+r-1$ determined values is not an available distance, the partial reconstruction cannot be extended into a pyramid and so we backtrack. If all correspond to unused distances, we mark them as used and advance to the next level. Since the ith choice determines i values, $n-1$ choices are sufficient to determine any pyramid. On the ith choice, we will perform i binary searches in a sorted list of the distances to test whether the ones we need are available.

By assigning $d_{1n} = v[\binom{n}{2}]$, $d_{1(n-1)} = v[\binom{n}{2} - 1]$, and $d_{(n-1)n} = d_{1n} - d_{1(n-1)}$, the search is initialized with $l = 2$ and $r = 1$.

Example: Suppose the input fragment set is $\{1, 2, 3, 4, 4, 5, 5, 6, 8, 10\}$. Then $n = 5$, and $d_{15} = 10$. The largest remaining segment 8, must be placed in either d_{14} or d_{25}. If $d_{14} = 8$, then $d_{45} = 10-8 = 2$. The next largest segment 6 must be assinged to either d_{13} or d_{25}. If we choose d_{13}, then $d_{25} = 10-6 = 4$ and $d_{34} = 8-6 = 2$. But, there is only one fragment of length 2, which we assigned to d_{45}. Hence, given $d_{14} = 8$, we must choose $d_{25} = 6$. This implies $d_{12} = 4$. The next largest element is 5, which we assign to d_{13}, and subsequenctly place $d_{35} = 5$, $d_{23} = 1$, $d_{34} = 3$ and $d_{24} = 4$. The pyramid corresponding to this solution is

```
        10
      8   6
    5   4   5
  4   1   3   2
```

When the data contains noise, we must use interval arithmetic to correctly handle fragments subject to a given relative error r. Now each fragment f defines an interval $[f \cdot (1-r), f \cdot (1+r)]$. An interval is also associated with each position in the pyramid, which gives upper and lower bounds on the length of any fragment which can be placed in this slot, Specifically,

$$\max(d_{ik}) + \max(d_{kj}) \geq d_{ij} \geq \min(d_{ik}) + \min(d_{kj})$$

and

$$\max(d_{1n}) - \min(d_{1i}) + \max(d_{jn}) \geq d_{ij} \geq \min(d_{1n}) - \max(d_{1i}) + \min(d_{jn})$$

for all appropriate i, j, k. Here, the max and min functions are defined as follows: $\max(d_{i(i+1)}) = d_{i(i+1)}(1+\epsilon)$, $\max(d_{ik}) = \sum_{j=i}^{k-1} \max(d_{j(j+1)})$, $i < k < j$ and the min function defined analogously. An assignment A of the pyramid positions to the input fragments is said to be *globally consistent* if there exists a set of values for d_{ij} such that $d_{ij} \in [A(d_{ij}) \cdot (1-r), A(d_{ij}) \cdot (1+r)]$ and

$$d_{ik} = \sum_{j=i}^{k-1} d_{j(j+1)}$$

for $i \leq i < k \leq n$. Our implementation checks only local interval constraints, namely

$$\max(d_{i(j-1)}) + \max(d_{(j-1)j}) \geq d_{ij}$$

$\geq \min(d_{i(j-1)}) + \min(d_{(j-1)j})$

and

$\max(d_{i(i+1)}) + \max(d_{(i+1)j}) \geq d_{ij}$
$\geq \min(d_{i(i+1)}) + \min(d_{(i+1)j})$

during search, but then verifies the global consistency of any completed pyramid in $O(n^3)$ time.

Although there still exist only two possible positions for the largest fragment, the derived pyramid values are no longer unique, since multiple input fragments may lie in any given interval. To ensure that no solution is missed, we must continue the search with every possible combination of candidate fragments. Thus the nodes of the backtrack tree may have exponentially large outdegree, and the worst-case behavior of the algorithm may become significantly worse than $\Theta(2^n \cdot n \ln n)$. This suggests using a heuristic to select the fragment when there are multiple choices in a given interval, such as using the smallest or median fragment. Although this eliminates a potentially vast amount of redundant computation, often no solution will be found even though one exists.

Another approach to reducing redundant computation is grouping. The input data typically contains several pairs of fragments of such similar length that they are essentially interchangable. Since k pairs of interchangable elements increases the number of solutions (and execution time) by 2^k, in practice the data is preprocessed by replacing all fragments whose length are within a given grouping percentage of each other with their mean. Again, grouping provides a significant speedup, at the cost of potential missed solutions.

Analysis

Although the backtracking algorithm takes exponential time in the worst case, we expect better behavior in practice. When distance set, measured without error, arises from n real points in general position, i.e., no two points coincide, the incorrect position for the largest distance will (with probability 1) be pruned immediately, and so the procedure uses $\Theta(n^2 \log n)$ operations. In this section, we analyze the sensitivity of this algorithm to the relative error in the input.

To analyze the behavior of our reconstruction algorithm, we assume that inputs are generated according to the following probabilistic model. Let $[a, b]$ denote the closed interval from a to b. In the interval $[1, M]$ a set of integer points S are chosen according to a binomial distribution $B(M, p)$, where $p = n/M$. Thus the expected number of points in S is n. If we assume that DNA molecules are random strings, then $p \approx 1/4^k$ for a restriction enzyme which cuts at a pattern of length k.

In the backtracking procedure, the act of placing the kth largest fragment defines an interval for the value of $k - 1$ positions in pyramid. If any one of these intervals does not contain an available element of the input distance set, we immediately backtrack. If an interval contains multiple fragments, we must continue the search with all combinations of the multiple fragments. Thus there are two critical points in the algorithm; first, where the probability of a false match becomes high enough that the procedure backtracks infrequently, and second where intervals contain multiple fragments often enough that each tree node has high outdegree. By arbitrarily selecting one of the multiple fragments in the interval, using a preselection rule, such as the median, we avoid the effects of this second phenomenon. The following theorem gives the expected time of the preselection rule algorithm. For the details of the proof, refer to [13].

Theorem 1 *Let I be the set of fragments resulting from selecting points from the interval $[1, M]$ according to a binomial distribution, with $p = n/M$, where $M \gg n$. Assume that all base fragments in I are identified, and that each fragment in I is measured with a relative error of at most r. Then the expected time of the median selection algorithm on I is $O(n^3)$ when $r \leq \ln 2/(2n)$.*

The heuristic algorithm analyzed in Theorem 1 avoids the effects of multiple fragments per interval, at the cost of possibly missing solutions. For a more complete analysis of the backtracking algorithm, we need two quantities: the probability that an interval is devoid of fragments, and the expected number of fragments per interval. The following theorem states, when $r = o(1/n^2)$, the expected time of the backtracking algorithm is $O(n^3)$.

Theorem 2 *Let I be the set of fragments resulting from selecting points from an interval of length M according to a binomial distribution, with $p = n/M$, where $M \gg n$. Assume that all base fragments in I are identified, and that each fragment in I measured with a relative error of at most r. Then the expected time of the backtracking algorithm on I is $O(n^3)$ when $r = o(1/n^2)$.*

Experimental Results

We conducted our experiments on the bacteriophage λ [6], a common cloning vehicle with seven restriction sites for the enzyme *Hind*III. The distances between adjacent sites for this molecule are $\{23130, 2027, 2322, 9416, 564, 125, 6557, 4361\}$ which leads to the following set of 36 pairwise distances:

125, 564, 689, 2027, 2322, 4349, 4361,
6557, 6682, 7246, 9416, 9980, 10105, 10918,
11043, 11607, 11738, 12302, 12427, 13765,
14329, 14454, 16662, 18984, 21011, 21023,
23130, 23345, 25157, 25372, 27479, 36895,
37459, 37584, 44141, 48502

We perturbed the length of each of these fragments to simulate the effects of experimental error on the performance of our algorithm. Specifically, to simulate a relative error r, each fragment of length f was replaced with a random integer selected uniformly from the interval $[f \cdot (1-r), f \cdot (1+r)]$.

Thus each problem instance has an associated relative error r, and was run under a variety of different grouping factors. Increasing the grouping factor has the effect of dramatically reducing the search time while less dramatically reducing the number of solutions. For each pair of r and g, the execution time and number of solutions have been reported. An '∞' denotes pairs where the program was not run to completion.

By performing both a complete and an incomplete digest on clones of a given molecule, the base fragments of our pyramid can be instantly identified, reducing the search times substantially for a given problem instance. In Table 1, the base fragments have been identified to reduce the search. Under this model, problem instances with relative errors of up to 7.0% can be solved.

Missing fragments present a significant challenge to our algorithm. The most common cause of missing fragments occurs when two fragments of similar length are not resolved on the gel. In such a case, the approximate length of the fragments are known, but not the multiplicity. For the experiments reported in Table 2, all fragments clustered together within the given grouping percentage were replaced by one instance of the mean length of these fragments. Thus, the problems get harder with increased grouping. The number of missing fragments is reported within parentheses in Table 2. Problems with as many as eight missing fragments are successfully handled, and the number of satisfying solutions remains extremely small. Indeed, none of our experiments to date have resulted in two significantly different interpretations of the same data. The full paper [13] contains a more thorough presentation of our experiments.

The Implementation

In this section, we describe the current implementation of our algorithm, *turnpike.c*. The program is available by contacting either of the authors of this paper.

- Since each fragment represents a distance between two sites, the expected number of fragments is $\binom{n}{2}$ for some n. By the default, n is taken as the smallest integer such that $\binom{n}{2} \geq m$, where m is the number of input fragments. The default number of wildcard entries is $\binom{n}{2} - m$. This can be overridden by the -n option.
- As discussed below, each input segment can be annotated to signify if it should always be treated as either a base or intermediate fragment. By default, any fragment is free to occupy any position in the solution.
- In order to speed up the program and minimize the number of isomorphic solutions, the -g (or grouping) option permits the user to specify a grouping percentage g. This preprocesses the input segments, and replaces all fragments within a window of g% with mean of their lengths.
- Some rudimentary facilities exist to assess the relative quality of multiple solutions. The -r (or ranking) option computes two statistics for each solution. The *minimum relative error* gives the smallest relative error under which this solution would be generated. The *stable segments* sums over all positions in a pyramid the number of other solutions which place the same fragment in the same position. The -b option suppresses output of the pyramids.
- The implementation provides two ways to deal with missing fragments. The -W option treats all missing fragments as wildcards, meaning that they can take any value. The -M option assumes that all missing fragments are multiplicities of the input fragments, and so restricts their possible values to the given input. These options are mutually exclusive.
- To avoid the combinatorial effects of multiple fragments per interval, the -m, -s, and -h options implement heuristics which always select the median, smallest, and maximum overlap fragments (respectively). Of these, the -h option appears to give the best performance, although it still misses too many solutions to justify its use.

Example

To compile the program *turnpike.c*, do:

cc -O -o turnpike turnpike.c -lm

The usage string is echoed if no arguments are provided:

```
% turnpike
Usage: turnpike -g[grouping_pct] -r -m
-n -s -h -b -M -W data_file pct_err
```

Each line of the input file contains the length of one fragment, possibly starting with an annotation character. Fragments beginning with a '*' are associated with a complete digest, and hence will be placed only on the base level of a pyramid. Fragments beginning with a '&' denote intermediate fragments, which cannot be placed on the base level of a pyramid. Unannotated fragments may be placed on any level. The following example file *demo-input* contains 7 '*' fragments, 3 unannotated segments, and 23 '&' segments. Since 36 is the smallest value of $\binom{n}{2} > 33$, by default three wildcards are added.

```
*126 *570 *2339 &4332  *4385  *6616
&6723  *9414  9950 10193  10818  &10933
&11609  &11853  &12261  &12512  &13832
&14357  &14506 &16645  &18815  &20818
&21160  *23096  &23237  &24931  &25196
&27531  &37170 &37283  &37648  &44460
&48737
```

Executing the command *turnpike -g0.5 -W -r demo-data 2.0* runs the program using a relative error of 2.0%, a grouping factor of 0.5%, a default number of wildcards, and produces statistics to rank the relative quality of the solutions.

```
Number of Wild Card Entries: 3
Relative Error(Percent): 2.000000
Grouping Percent: 0.500000
INPUT DISTANCES:
126    570    2339    4332    4385    6616
6723   9414    9950   10193 10818   10933
11609  11853   12261   12512   13832   14357
14506  16645 18815   20818   21160   23096
23237  24931   25196   27531   37170   37283
37648  44460   48737

GROUPED INPUT
126    570    2339    4332    4385    6616
6723   9414    9950   10193 10818   10933
11609  11853   12261   12512   13832   14357
14506  16645 18815   20818   21160   23166
23166  24931   25196   27531   37367   37367
37367  44460   48737

SOLUTIONS & STATISTICS

Min. Pct. Rel. Error : 1.507813
Stable Segments 24

                    48737

                44460  25196

            37648  21160  23237

        37283  14506  18815  20818

    37170 14357  12512  16645 11853

  27531 13832  12261  10193  WCARD  10933

 24931   4332  11609   9950  WCARD  6723  10818

23096   2339  WCARD   9414    570    126   6616   4385

Time Taken for the 1 solution: 13.940000

Min. Pct. Rel. Error : 0.929688
Stable Segments 24

                    48737

                44460  25196

            37648 20818  23237

        37283 14506  18815  21160

    37170  14357  12512  16645  11609

  27531  13832  12261  10193   WCARD  10933

 24931   4332   11853  9950   WCARD   6723  10818

23096  WCARD   2339   9414    570    126   6616   4385

Time Taken for the 2 solution: 0.000000

Total Time :29.100000  secs
```

Conclusion

We presented a new practical combinatorial algorithm to resolve the experimental data of restriction site analysis. We analyzed the performance of our algorithm under a reasonable probability distribution and established a relative error limit of $o(1/n^2)$ upto which our technique is feasible. The conclusion to be drawn is that our technique of restriction site analysis via partial digests successfully reconstructed simulated data under realistic error tolerances. For typical problems, the algorithm can handle upto a relative error of 7%. The next phase of this work will involve reconstructing actual laboratory data.

Acknowledgements

We would like to thank Bill Chang, Scott Davey, and Tom Marr of Cold Spring Harbor Laboratory for supplying the data, and for their interest in

this work. Discussions with Joel Spencer helped with the probabilistic analysis.

References

[1] L. Allison and C. N. Yee. Restriction site mapping is in separation theory. *Comput. Appl. Biol. Sci.*, 4:97–101, 1988.

[2] B. Bellon. Construction of restriction maps. *Comput. Appl. Biol. Sci.*, 4:111–115, 1988.

[3] W. I. Chang and T. Marr. Personal communication. Cold Spring Harbor Laboratory, April 1992.

[4] T.I. Dix and D.H. Kieronska. Errors between sites in restriction site mapping. *Comput. Appl. Biol. Sci.*, 4:117–123, 1988.

[5] A. V. Grigorjev and A. A. Mironov. Mapping DNA by stochastic relaxation: a new approach to fragment sizes. *Comput. Appl. Biol. Sci.*, 6:107–111, 1990.

[6] T. Maniatis, E. F. Fritsch, and J. Sambrook. *Molecular Cloning, a laboratory manual.* Cold Spring Harbor Laboratory, Cold Spring Harbor NY, 1982.

[7] A. L. Patterson. Ambiguities in the x-ray analysis of crystal structures. *Phys. Review*, 65:195–201, 1944.

[8] A. L. Patterson. A direct method for the determination of the components of interatomic distances in crystals. *Zeitschr. Krist.*, 90:517–542, 1935.

[9] G. Rhee. *DNA restriction mapping from random-clone data.* Technical Report WUCS-88-18, Department of Computer Science, Washington University, St. Louis MO, 1988.

[10] J. Rosenblatt and P. Seymour. The structure of homometric sets. *SIAM J. Alg. Disc. Methods*, 3:343–350, 1982.

[11] L. Allison S. Ho and C. N. Yee. Restriction site mapping for three or more enzymes. *Comput. Appl. Biol. Sci.*, 6:195–204, 1990.

[12] S. S. Skiena, W. D. Smith, and P. Lemke. Reconstructing sets from interpoint distances. In *Proc. Sixth ACM Symp. Computational Geometry*, pages 332–339, 1990.

[13] S. S. Skiena and G. Sundaram. A partial digest approach to restriction site mapping. *submitted to Bulletin of Mathematical Biology.*

[14] M. Stefik. Inferring DNA structures from segmentation data. *Artificial Intelligence*, 11:85–114, 1978.

[15] R. A. Sutherland and M. D. Partis. Using fragment lengths from incomplete digestion by multiply cleaving enzymes to map antibody binding sites on a protein. *Comput. Appl. Biol. Sci.*, 8:1–4, 1992.

[16] P. Tuffery, P. Dessen, C. Mugnier, and S. Hazout. Restriction map construction using a 'complete sentences compatibility' algorithm. *Comput. Appl. Biol. Sci.*, 4:103–110, 1988.

Reconstruction Time in Seconds

rel. error	*grouping percentage* 0.0%	0.5%	1.0%	1.5%	2.0%	2.5%	3.0%
0.0%	0.05						
0.5%	0.03						
1.0%	0.04	0.06					
1.5%	0.35	0.05					
2.0%	0.53	0.22	0.03				
2.5%	1.81	0.63	0.07				
3.0%	5.55	3.03	1.51	1.47			
3.5%	7.66	2.18	0.73	0.38			
4.0%	24.17	7.0	1.39	0.24	0.16		
4.5%	67.31	35.76	20.62	1.27	0.27		
5.0%	∞	172.41	30.3	7.01	0.88	0.38	
5.5%	∞	∞	108.72	5.42	5.41	0.86	
6.0%	∞	∞	114.45	109.74	32.51	11.63	4.87
6.5%	∞	∞	∞	∞	161.19	99.1	14.76
7.0%	∞	∞	∞	∞	265.12	266.83	78.58

Number of Solutions

rel. error	*grouping percentage* 0.0%	0.5%	1.0%	1.5%	2.0%	2.5%	3.0%
0.0%	1						
0.5%	1						
1.0%	1	1					
1.5%	1	1					
2.0%	1	1	0				
2.5%	1	1	0				
3.0%	2	1	1	1			
3.5%	2	2	2	1			
4.0%	2	2	2	1	1		
4.5%	2	2	2	2	0		
5.0%	∞	3	2	2	0	0	
5.5%	∞	∞	3	1	1	0	
6.0%	∞	∞	2	2	0	0	
6.5%	∞	∞	∞	∞	0	0	0
7.0%	∞	∞	∞	∞	3	3	0

Table 1: Performance when provided with 36 of 36 fragments, and with all of the base fragments identified.

Reconstruction Time in Seconds

rel. error	*grouping percentage* 0.0%	0.5%	1.0%	1.5%
0.0%	0.03 (0)			
0.5%	0.04 (0)			
1.0%	0.05 (0)	0.1 (3)		
1.5%	0.26 (0)	0.13 (5)		
2.0%	0.32 (0)	3.13 (5)	0.29 (11)	
2.5%	1.08 (0)	14.52 (3)	0.77 (8)	
3.0%	2.92 (0)	∞ (4)	239.41 (6)	55.31 (7)
3.5%	4.04 (0)	23.59 (2)	240.63 (6)	∞ (8)
4.0%	13.34 (0)	65.59 (2)	∞ (6)	∞ (9)
4.5%	37.01 (0)	133.01 (1)	∞ (4)	∞ (8)

Number of Solutions

rel. error	*grouping percentage* 0.0%	0.5%	1.0%	1.5%
0.0%	1			
0.5%	1			
1.0%	1	1		
1.5%	1	1		
2.0%	1	1	0	
2.5%	1	2	1	
3.0%	2	∞	3	0
3.5%	2	2	2	∞
4.0%	2	2	∞	∞
4.5%	2	2	∞	∞

Table 2: Performance when provided with single copies of multiple fragments.