A Note on Scoring Clones Given a Probe Ordering

Mudita Jain * Eugene W. Myers †

December 28, 1995

Abstract

We present an efficient algorithm for scoring clones given an ordering of probes under a schema proposed by Alizadeh et al. [1] in the context of physical mapping with unique probes. The algorithm runs in time linear in the number of blocks of ones in the underlying sparse incidence matrix. A sparse and efficient algorithm for this task is important as it appears to be a central task in most algorithms for physical mapping.

1 Introduction

The problem of physical mapping is to find an interval ordering of a collection of clones (fragments) of a DNA strand by inferring the overlap order from fingerprint data. One of the ways in which clones are fingerprinted is by recording hybridization data between a set of STS probes and the clones. Each STS probe is assumed to be specific enough that it hybridizes to a unique location in the underlying DNA sequence. Given m STS probes and n clones, the input for the problem is then presented as an $m \times n$ incidence matrix D, where d_{ij} is 1 if probe i hybridizes with clone j and 0 otherwise. Since the clones are intervals of the DNA strand, a permutation π of the rows/probes that gives the permuted matrix D_π the consecutive ones

*Dept. of Computer Science, University of Arizona, Tucson, AZ 85721 (e-mail: jainm@cs.arizona.edu). Fully supported by NLM grant LM-04960

†Dept. of Computer Science, University of Arizona, Tucson, AZ 85721 (e-mail: gene@cs.arizona.edu). Partially supported by NLM grant LM-04960

property, gives an ordering of the clones [2]. However, in practice, the measured data contains errors in the form of false positives (i.e. a 1 where there should be a 0), false negatives (i.e. a 0 where there should be a 1), and chimeras (i.e. two distinct intervals reported as one). Under these more realistic assumptions, the problem becomes one of finding the most-likely permutation π and set of error corrections that give D_π the consecutive ones property. Figure 1 shows an example of the actual layout versus the measured data.

	C1	C2	C3	C4	C5	C6
A	1	0	0	0	0	0
B	1	0	0	0	0	1
C	1	1	0	1	0	0
D	0	1	1	1	0	0
E	0	0	1	1	1	0
F	0	0	1	0	1	1

Actual data

	C1	C2	C3	C4	C5	C6
A	1	0	0	0	**1**	0
B	1	0	0	0	0	1
C	1	1	0	1	0	0
D	0	1	1	**0**	0	0
E	0	0	1	1	1	0
F	0	0	1	0	1	1

Measured data

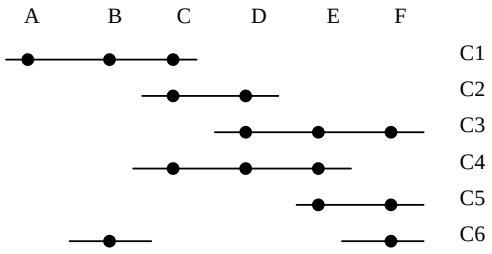

Figure 1: Actual vs. Measured Data. The clones are denoted by $C1$, $C2$, $C3$, $C4$, $C5$, $C6$, and A, B, C, D, E, F represent the probes. The layout on the right indicates the actual layout of the clones where $C6$ is chimeric. The false positives and negatives in the measured data incidence matrix are emboldened.

Correcting a false positive is to turn a 1 into a 0, correcting a false negative is to turn a 0 into a 1, and correcting a chimera is to split a column into two, or alternatively, to allow two blocks of consecutive 1's in a column of D_π. The goal is to correct D_π to the matrix that is most-likely to reflect the actual, perfect data. Assuming that each type of error occurs at a fixed rate, Alizadeh et al. [1] show that minimizing the objective function

$$C \cdot \chi(D_\pi) + P \cdot fp(D_\pi) + N \cdot fn(D_\pi)$$

maximizes the likelihood, Probability$(A|D_\pi)$, of A given D_π, where $\chi(D_\pi)$ is the number of chimera corrections, $fp(D_\pi)$ is the number of false positive corrections, $fn(D_\pi)$ is the number of false negative corrections, and C, P, and N are non-negative constants reflecting the error rates. Specifically, if false

negatives occur at fixed rate ϵ, false positives at rate δ, and chimeras at rate p, then these constants were shown to be the log-likelihood ratios: $C = -\ln \frac{p}{1-p}$, $P = -\ln \frac{\epsilon}{1-\delta}$, and $N = -\ln \frac{\delta}{1-\epsilon}$.

A primary focus of the work of Alizadeh et al. is how to intelligently explore the space of row permutations (probe orderings) in search of ones that permit the objective function to be minimized. So *given* a permutation π, a central subtask is to find a set of corrections to D_π that minimizes the objective function. Since the objective is additive, it suffices to determine how to optimally correct each column of D_π on an individual basis. Alizadeh et al. give an algorithm that is quadratic in the number of ones in a column. We improve this to linear time, and further generalize it to permit chimeras that are the union of up to k distinct intervals (called *k-meras*). We make no assumption about the distribution of the ordinality of chimeras by using the objective function

$$\Sigma_{c=1}^{k} C(c) \cdot \chi_c(D_\pi) + P \cdot fp(D_\pi) + N \cdot fn(D_\pi)$$

where $C(c)$ may be any score for c-meras and $\chi_c(D_\pi)$ is the number of corrected c-meras. On the other hand, we do not consider here the case where one knows that some probe incidences are at the ends of the clone involved, as in [1], in order to keep the treatment simple. We discuss at the end of this note how one can proceed to generalize our result to handle such additional probe information.

2 The Basic Linear Algorithm

We begin by developing an algorithm that is linear in the length m of a column V of D_π, and will derive our sparse algorithm in the next section. We use the following notation for regular expressions: $r|s$ denotes the union or disjunction of patterns r and s, rs denotes their concatenation, r^k denotes the concatenation of k copies of r, and r^* denotes the concatenation of zero-or-more copies. Also, w_i will denote the i^{th} of a string w.

We begin by viewing column $V = v_1 v_2 \ldots v_m$ as a string of 0's and 1's. Observe that our goal is to correct V to match the pattern $(0^*1^*)^k 0^*$ with minimal cost. Without chimera costs, the problem is clearly just an instance of the approximate regular expression pattern matching problem solvable in $O(mk)$ time [8]. Moreover, we claim without proof that the edit graph formalism developed in [8] can be modified to accommodate chimera costs. However, in order to ultimately derive a sparse algorithm, we proceed with an independent development.

Let B^c be the pattern $(0^*1^*)^c 0^*$ for $0 \leq c \leq k$. Further let $\Delta(V, \textit{pat})$ be the minimum cost of making false positive and negative corrections to V in order that it match pattern *pat* precisely. Formally,

$$\Delta(V, \textit{pat}) = \min_{w \in pat, |w|=m} (\Sigma_{i=1}^{m} \Delta(v_i, w_i)).$$

$$\Delta(a, b) = \begin{cases} N & \text{if } a = 0, b = 1 \quad \text{(false negative)} \\ P & \text{if } a = 1, b = 0 \quad \text{(false positive)} \\ 0 & \text{if } a = b \end{cases}$$

Informally, $\Delta(V, \textit{pat})$ is the minimum weighted Hamming distance between V and a word of length m in pat. By construction $\Delta(V, B^c) + C(c)$ is the cost of correcting V into a c-mera. Therefore, the minimal cost correction for column V is $\min_{c \in [1,k]}(\Delta(V, B^c) + C(c))$.

To compute $\Delta(V, B^c)$, we develop tandem dynamic programming recurrences for the cost of correcting every prefix of the column to B^c and also to the pattern $E^c = (0^*1^*)^c$ for $1 \leq c \leq k$. Let V_i represent the prefix $v_1 v_2 \ldots v_i$ of column V. The lemma below provides the needed recurrences.

Lemma 1

$$\Delta(V_i, B^k) = \begin{cases} 0 & \textit{if } i = 0 \\ \Delta(V_{i-1}, B^k) + \Delta(v_i, 0) & \textit{if } i > 0 \textit{ and } k = 0 \\ \min(\Delta(V_i, E^k), \Delta(V_{i-1}, B^k) + \Delta(v_i, 0)) & \textit{otherwise} \end{cases}$$

$$\Delta(V_i, E^k) = \begin{cases} 0 & \textit{if } i = 0 \\ \min(\Delta(V_i, B^{k-1}), \Delta(V_{i-1}, E^k) + \Delta(v_i, 1)) & \textit{otherwise} \end{cases}$$

Proof: By induction on k and i. The recurrence for $\Delta(V_i, B^k)$ follows directly from the pattern identity $B^k = E^k|B^k0$, and that for $\Delta(V_i, E^k)$ from the identity $E^k = B^{k-1}|E^k1$. For example, if $i, k > 0$ then $\Delta(V_i, B^k) = \Delta(V_i, E^k|B^k0) = \min(\Delta(V_i, E^k), \Delta(V_i, B^k0))$ and $\Delta(V_i, B^k0) = \Delta(V_{i-1}v_i, B^k0) = \Delta(V_{i-1}, B^k) + \Delta(v_i, 0)$. ∎

The recurrences permit each quantity to be computed in constant time from previously computed quantities. Since there are $(2k+1)(m+1)$ quantities, this leads directly to an $O(km)$ time, $O(k)$ working-space algorithm for evaluating the cost of column V. Note that in the context of Alizadeh et al. [1], k is fixed at 2 and our algorithm takes $O(m)$ time and constant working space.

Figure 2 shows the computed recurrences B^0, E^1, and B^1 for the example data in Figure 1 with the correct permutation of the probes, and $P = N = 1$.

	C1	C2	C3	C4	C5	C6
A	1	0	0	0	**1**	0
B	1	0	0	0	0	1
C	1	1	0	1	0	0
D	0	1	1	**0**	0	0
E	0	0	1	1	1	0
F	0	0	1	0	1	1

Measured data

0	0	0	0	0	0	0
1	1	0	0	0	1	0
2	2	0	0	0	1	1
3	3	1	0	1	1	1
4	3	2	1	1	1	1
5	3	2	2	2	2	1
6	3	2	3	2	3	2

B(0)

0	0	0	0	0	0	0
1	0	0	0	0	0	0
2	0	0	0	0	1	0
3	0	0	0	0	1	1
4	1	0	0	1	1	1
5	2	1	0	1	1	1
6	3	2	0	1	1	1

E(1)

0	0	0	0	0	0	0
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	0	0	0	0	0	0
5	0	0	0	1	1	0
6	0	0	0	1	1	1

B(1)

Figure 2: The computed recurrences B^0, E^1, and B^1, given the probe permutation of Figure 1.

3 The Sparse, Block-Based Algorithm

Typically the underlying incidence matrix D is sparse in that 10% or less of the entries in the matrix have value 1. To take advantage of this sparsity, note the simple observation that it is never advantageous to

begin or end a clone boundary in the middle of a "block" of consecutive zeroes or consecutive ones. This observation leads to the simple extension of our previous algorithm presented below. Note that the algorithm is sparse because the number of blocks is not more than twice plus one the number of ones in a row.

Let $V^{'} = v_1^{'} v_2^{'} \cdots v_{m'}^{'}$ be the homeomorphic image of V obtained by replacing every block of ones with a single 1, and every block of zeroes with a single 0. Further let l_i be the length of the i^{th} block. Now we can simply extend the recurrence of the previous section to compute $\Delta(V_i^{'}, B^k)$ and $\Delta(V_i^{'}, E^k)$ save that now we must weight every character replacement $\Delta(v_i^{'}, b)$ by the length l_i of the block as shown in the statement of the corollary below. With this weighting it is clear that $\Delta(V_i^{'}, X) = \Delta(V_s, X)$ where $s = \sum_{k=1}^{i} l_k$.

Corollary 1

$$\Delta(V_i^{'}, B^k) = \begin{cases} 0 & \textit{if } i = 0 \\ \Delta(V_{i-1}^{'}, B^k) + \Delta(v_i^{'}, 0) * l_i) & \textit{if } i > 0 \textit{ and } k = 0 \\ \min(\Delta(V_i^{'}, E^k), \Delta(V_{i-1}^{'}, B^k) + \Delta(v_i^{'}, 0) * l_i)) & \textit{otherwise} \end{cases}$$

$$\Delta(V_i^{'}, E^k) = \begin{cases} 0 & \textit{if } i = 0 \\ \min(\Delta(V_i^{'}, B^{k-1}), \Delta(V_{i-1}^{'}, E^k) + \Delta(v_i^{'}, 1) * l_i)) & \textit{otherwise} \end{cases}$$

It follows as in the previous section that the cost of the best correction for column V is $\min_{c \in [1,k]}(\Delta(V^{'}, B^c) + C(c))$. Thus given the B blocks of a matrix D_π, the recurrences above allow us to evaluate its score/likelihood in $O(kB)$ time.

The remaining difficulty is that in order to evaluate D_π we must be able to deliver its blocks. Naively, this would require $O(nm)$ time. Recall that an overall algorithm for physical mapping will start with D, explore permutations π of D, and for each D_π will evaluate the score of all its columns. Before the overall algorithm begins take $O(mn)$ time to determine lists, U_j, of the positions of the 1s in each column j of D. When the evaluation of D_π is required for a specific permutation π, use bucket sort to simultaneously sort $\pi(U_j)$ for all j in $O(m + E)$ time, where $E = \Sigma_j |U_j|$, *i.e.*, the number of ones in D. Given the sorted ones-lists, one can

then deliver the blocks of each row in $O(E)$ additional time. Thus, with $O(mn)$ preprocessing, we may then evaluate the score/likelihood of any D_π in $O(m + E + kB)$ time. Since B is $O(E)$ it would be desirable if only $O(m + kB)$ time were taken. This is possible, but only in situations where the exploration of possible choices of π allows the $O(1)$ updating of block boundaries as π is being manipulated, for example, 2-OPT or other swapping strategies.

4 Discussion

We conclude by noting that because our algorithm is a specialization of an approximate regular expression pattern matching algorithm [8], it follows that other sequence comparison results apply. For example, the corrections achieving the best score can be obtained by the usual divide-and-conquer approach [6, 7]. It is also true that one can model any row property expressible as a regular expression over a finite scalar alphabet, and so, for example, can accommodate the end-clone hybridizations described in [1]. Efficiency improvements might be possible using path compression ideas [4, 10] and/or shortest path approaches [6, 3]. The efficiency of the presented algorithm(s) in practice and how they are best rendered to such practice is an open issue.

5 Acknowledgement

The authors wish to thank one of the referees whose comments lead to a significantly simpler exposition of the sparse algorithm.

References

[1] Farid Alizadeh, Richard M. Karp, Deborah K. Weisser, and Geoffrey Zweig. “Physical Mapping of

Chromosomes Using Unique Probes". In *Proc. 5th ACM-SIAM Symposium on Discrete Algorithms,* 1994.

[2] Kellogg S. Booth and George S. Leuker. "Testing for the Consecutive Ones property, Interval Graphs, and Graph Planarity Using PQ-Tree Algorithms". In *J. Comp. and Syst. Sci.*, vol.13, pp. 335-379, 1976.

[3] Ficket, J.W. "Fast Optimal Alignment". In *Nucleic Acids Research,* vol. 12, pp. 175-179, 1984.

[4] F. Hadlock. "Minimum Detour Methods for String or Sequence Comparison". In *Congressus Numerantium,* vol. 61, pp. 263274, 1988.

[5] Toru Mizukami, William I. Chang, Igor Garkavtsev, Nancy Kaplan, Diane Lombardi, Tomohiro Matsumoto, Osami Niwa, Asako Kounosu, Mitsuhiro Yanagida, Thomas G. Marr, and David Beach. "A 13kb Resolution Cosmid Map of the 14 Mb Fission Yeast Genome by Nonrandom Sequence-Tagged-Site Mapping". In *Cell,* vol.73, pp. 121-132, 1993.

[6] Eugene W. Myers. "An O(ND) Difference Algorithm and Its Variations". In *Algorithmica,* vol. 1, pp. 251-266, 1986.

[7] Webb Miller, and Eugene W. Myers. "Optimal Alignments in Linear Space". In *CABIOS,* vol. 4, pp. 11-17, 1988.

[8] Eugene W. Myers, and Webb Miller. "Approximate Matching of Regular Expressions". In *Bulletin of Mathematical Biology,* vol.51, pp. 5-37, 1989.

[9] Michael J. Palazzolo, Stanley A. Sawyer, Christopher H. Martin, David A. Smoller, and Daniel L. Hartl. "Optimized Strategies for Sequence-Tagged Site Selection in Genome Mapping". In *Proc. Natl. Acad. Sci. USA,* vol.88, pp. 8034-8038, 1991.

[10] Wu, S., Myers E., Manber, U. and W. Miller. “An O(NP) Sequence Comparison Algorithm”. In *Information Processing Letters*, vol. 35, pp. 317-323, 1990.