BIOINFORMATICS

Vol. 19 Suppl. 1 2003, pages i122–i129
DOI: 10.1093/bioinformatics/btg1016

Detecting protein sequence conservation via metric embeddings

E. Halperin[1,], J. Buhler[2], R. Karp[1], R. Krauthgamer[1] and B. Westover[2]*

[1]International Computer Science Institute and Computer Science Division, University of California, Berkeley, CA 94720 and [2]Department of Computer Science and Engineering, Box 1045, Washington University, One Brookings Drive, St. Louis, MO 63130, USA

Received on January 6, 2003; accepted on February 20, 2003

ABSTRACT

Motivation: Comparing two protein databases is a fundamental task in biosequence annotation. Given two databases, one must find all pairs of proteins that align with high score under a biologically meaningful substitution score matrix, such as a BLOSUM matrix (Henikoff and Henikoff, 1992). Distance-based approaches to this problem map each peptide in the database to a point in a metric space, such that peptides aligning with higher scores are mapped to closer points. Many techniques exist to discover close pairs of points in a metric space efficiently, but the challenge in applying this work to proteomic comparison is to find a distance mapping that accurately encodes all the distinctions among residue pairs made by a proteomic score matrix. Buhler (2002) proposed one such mapping but found that it led to a relatively inefficient algorithm for protein-protein comparison.

Results: This work proposes a new distance mapping for peptides under the BLOSUM matrices that permits more efficient similarity search. We first propose a new distance function on peptides derived from a given score matrix. We then show how to map peptides to bit vectors such that the distance between any two peptides is closely approximated by the *Hamming distance* (i.e. number of mismatches) between their corresponding bit vectors. We combine these two results with the LSH-ALL-PAIRS-SIM algorithm of Buhler (2002) to produce an improved distance-based algorithm for proteomic comparison. An initial implementation of the improved algorithm exhibits sensitivity within 5% of that of the original LSH-ALL-PAIRS-SIM, while running up to eight times faster.

Availability: The source of the code can be found at http://www.eecs.berkeley.edu/∼eran/projects/embed.

Contact: eran@eecs.berkeley.edu

Keywords: protein comparison, database indexing, metric embedding, Hamming space

*To whom correspondence should be addressed.

INTRODUCTION

Large-scale proteomic sequence comparison is a computationally challenging task in genome annotation. The rapid growth of biosequence databases demands fast filtering and indexing strategies to winnow a few meaningful similarities between proteins from millions or billions of irrelevant residues of sequence. These search strategies are typically informed by substitution score matrices from the PAM (Dayhoff *et al.*, 1978) and BLOSUM (Henikoff and Henikoff, 1992) families, which describe evolutionary models for scoring ungapped protein alignments.

Two principal types of strategy are used to accelerate large proteomic comparisons: seed- and distance-based methods. Seed-based methods are typified by BLASTP (Altschul and Gish, 1996; Altschul *et al.*, 1997), which initially seeks short words in a protein database that score highly when aligned to a given query sequence. Only sequences with high-scoring *seed matches* are explored further. Seed-based methods are highly sensitive provided short seeds (3–4 residues) are used, and they can be accelerated in practice by preindexing seed matches or considering only non-overlapping matches (Ning *et al.*, 2001; Kent, 2002). However, using short seeds incurs many spurious seed matches in the absence of a meaningful alignment. In contrast, if we may treat peptides as points in space, such that the distance between peptides decreases as their alignment score increases, then a variety of techniques from computational geometry can organize a protein database so as to limit the number of spurious hits to *any* query while maintaining high sensitivity. Examples of this distance-based approach include FLASH (Califano and Rigoutsos, 1993), SST (Giladi *et al.*, 2002), and LSH-ALL-PAIRS (Buhler, 2001), which exploits the randomized indexing strategy of Indyk and Motwani (1998).

A key problem in applying distance-based methods to accelerate proteomic comparison is the difficulty of

combining these methods with biologically meaningful score matrices. PAM and BLOSUM matrices do not easily map to distance functions that guarantee that higher-scoring pairs of peptides will be closer together[†]. Most distance-based tools for biosequence comparison therefore elect not to use score matrices at all. SST measures only the *Hamming distance*, or number of mismatches, between sequences, which is not suitable for protein, while FLASH compares amino acids using a reduced class alphabet that captures only a few of their key biochemical properties.

Recent work by Buhler (2002) incorporated score matrices directly into distance-based search. That work mapped peptides to vectors so that the Hamming distance between two vectors was a simple affine function of the (ungapped) alignment score of their corresponding peptides. This *Hamming-space embedding* of peptides formed the basis of LSH-ALL-PAIRS-SIM, a similarity search algorithm with provably high expected sensitivity. However, the distance mapping used in Buhler (2002) induced only a small absolute separation of the distances between high-scoring versus low-scoring pairs of peptides, making it computationally expensive in practice to distinguish the two reliably. We therefore focus on the following still-open question: *Is there a distance function on peptides that provably captures the information in biologically meaningful score matrices, yet permits efficient application of distance-based methods to accelerate similarity search*?

We present an improved distance-based method for proteomic similarity search that utilizes nearly all the information in a substitution score matrix M. We first propose a mapping, motivated by work of Linial *et al.* (1997), from M to a distance function D on peptides of fixed length ℓ. We then use semidefinite programming followed by random hyperplane splitting to embed D into Hamming space with small distortion. As in Buhler (2002), peptides are mapped to vectors whose pairwise Hamming distances reflect their pairwise alignment scores under M. The new distance and embedding can replace those used previously in the LSH-ALL-PAIRS-SIM algorithm with only minor adjustments. The resulting search algorithm can perform all-pairs comparison of two large protein databases with sensitivity comparable to that reported in Buhler (2002) in as little as one-eighth the time.

The rest of the paper is organized as follows. In the next section, we introduce a distance function D for similarity matrices and observe that it is a metric (or nearly so) in practice. We then describe our embedding strategy, which consists of two phases. We first map the set of amino acids to vectors v_i on the unit sphere, such that the angle between each pair of vectors is approximately proportional to the distance between the corresponding residues. We compute the vectors v_i efficiently by semidefinite programming. We then use random hyperplane splitting to embed the vectors v_i into Hamming space, so that peptides at small distance under D are placed at small Hamming distance. The next section reviews the LSH-ALL-PAIRS-SIM algorithm for comparing proteomic databases and discusses the modifications needed to accommodate the new distance function D. Finally, we give empirical results on the modified algorithm's performance, then conclude by pointing out directions for future work.

[†] This guarantee was formalized in Buhler (2002) by the notion of *score simulation*.

ALGORITHMS

A distance mapping for similarity matrices

We first describe a mapping from an amino acid similarity matrix M to a distance function D. Let $\rho_i = M_{ii}$ be the score obtained by comparing residue i to itself. We define the distance D_{ij} on pairs of residues i, j as follows:

$$D_{ij} = \rho_i + \rho_j - 2M_{ij}.$$

Linial *et al.* (1997) applied a mapping similar to D_{ij} to pairs of peptides. We address this more general case by extending the definitions of both M and D to pairs of peptides of common length ℓ. Define the *self-score* $\rho(s)$ of a peptide s to be the score $\sum_{q=1}^{\ell} \rho_{s[q],s[q]}$ obtained by aligning s to itself without gaps. We define D on pairs of ℓ-mers to be

$$\begin{aligned} D(s,t) &= \sum_{q=1}^{\ell} D_{s[q],t[q]} \\ &= \rho(s) + \rho(t) - 2\sum_{q=1}^{\ell} M_{s[q],t[q]} \\ &= \rho(s) + \rho(t) - 2M(s,t). \end{aligned}$$

The function D is symmetric if M is symmetric, satisfies $D_{ii} = 0$, and is non-negative provided $M_{ii} \geq M_{ij}$ for residues $i \neq j$. The latter property is typically true of substitution score matrices, which consider identical residue pairs more likely to arise by conservation than by chance.

Our intent in the next section is to embed the distance D into a Hamming space with minimal distortion, i.e. to find a mapping ϕ from residues to $\{0, 1\}^T$ (for some dimension T) such that, for all pairs i, j of residues, the Hamming distance between $\phi(i)$ and $\phi(j)$ is approximately equal to D_{ij}. An isometric embedding (one with no distortion) is possible only if D is a *metric*, i.e. if it satisfies the triangle inequality $D_{ij} + D_{jk} \geq D_{ik}$. Using the definition of D, we can rewrite this inequality in terms of M as

$$M_{jj} + M_{ik} \geq M_{ij} + M_{jk}.$$

We checked a number of score matrices in the BLOSUM family to determine whether the above inequality holds for all i, j, k. We found that the inequality either holds (e.g. for BLOSUM-62) or is violated for only a small number of residue triples, and then only because $M_{jj} + M_{ik} = M_{ij} + M_{jk} - 1$. These violations may arise from the rounding used to make the BLOSUM matrices integer-valued. Our embedding construction should be robust to small violations of the triangle inequality, so these violations do not in practice pose a barrier to finding a low-distortion embedding of D.

A low-distortion Hamming-space embedding

We next describe how to map amino acids to vectors in a Hamming space $\{0, 1\}^T$ so as to nearly preserve the distances induced by D. In contrast to (Buhler, 2002), we do not require that the embedding be *isometric*, i.e. that two vectors lie at exactly the same distance as their corresponding residues under D. Instead, we try only to preserve the original distances to within a small multiplicative error. Such a small distortion of D should have only a minor effect on the sensitivity of a search algorithm using the embedding.

Bourgain (1986) showed that any finite metric on m points can be embedded into l_1 (and hence into a Hamming space) with distortion $O(\log m)$. Linial *et al.* (1995) devised a randomized algorithm that efficiently computes a variant of Bourgain's embedding. In practice, these embeddings frequently achieve their worst-case distortion bounds, or at least $\Omega(\sqrt{\log m})$ distortion, even for simple metrics such as a path, a complete binary tree, or a hypercube.

We have developed efficient embedding strategies based on semidefinite programming (Goemans and Williamson, 1995) that achieve low distortion in practice, with particular attention to avoiding the worst-case behavior of other methods. This section describes a heuristic strategy that works well with BLOSUM matrices.

The scaled hypercube embedding problem. Let (Σ, D) be a finite metric, with Σ a set of m elements and D their distance function. A *(scaled)* (T, γ)*-hypercube embedding* of (Σ, D) is a mapping $\phi : \Sigma \to \{0, 1\}^T$ such that, for some real scaling factor $\sigma > 0$ and all $i, j \in \Sigma$, we have $\sigma \leq \frac{d_H(\phi(i),\phi(j))}{D_{ij}} \leq \gamma \cdot \sigma$; throughout, $d_H(v, w)$ is the Hamming distance between vectors v and w (i.e. the number of positions in which v and w differ). We allow the scaling factor σ since it does not affect the 'complexity' of using the metric in many applications, including our intended one. In the *scaled hypercube embedding problem*, the input is a metric (Σ, D) and we wish to find a (T, γ)-hypercube embedding with both T and γ as small as possible. More precisely, if one of T or γ is given, we wish to minimize the other quantity.

A semidefinite programming heuristic. Our plan of action is as follows. First, we will construct a set of unit vectors $v_i \in \mathbb{R}^\delta$ (for some dimension δ) corresponding to the elements $i \in \Sigma$, such that the *angles* α_{ij} between vectors v_i and v_j are as nearly as possible proportional to the distances D_{ij}. Then, we will use the technique of *random hyperplane splitting* to construct a low-distortion Hamming-space embedding from the v_i's.

We initially seek to ensure that the angles α_{ij} roughly reflect the distances D_{ij}. In other words, for a distortion γ as small as possible and an arbitrary scaling factor σ_0, we ask that $\sigma_0 D_{ij} \leq \alpha_{ij} \leq \gamma \cdot \sigma_0 D_{ij}$ for all $i, j \in \Sigma$. Letting some inter-residue distances D_{ij} grow and others shrink may result in distortions canceling out when we sum the distorted distances over several residue pairs, as we do in a peptide-to-peptide comparison. Hence, we modify our low-distortion criterion to permit *two-sided error* as follows:

$$(1 - \gamma)\sigma_0 D_{ij} \leq \alpha_{ij} \leq (1 + \gamma)\sigma_0 D_{ij}.$$

Noting that $v_i \cdot v_j = \cos(\alpha_{ij})$ and that the cosine is monotone decreasing with its argument, we formulate our search for suitable vectors as the following optimization problem in $\mathbb{R}^\delta$:

$$\begin{array}{lll} \text{Min} & \gamma & \\ \text{s.t.} & v_i \cdot v_j \leq \cos\big((1-\gamma)\sigma_0 D_{ij}\big) & \forall i, j \in \Sigma \\ & v_i \cdot v_j \geq \cos\big((1+\gamma)\sigma_0 D_{ij}\big) & \forall i, j \in \Sigma \\ & v_i \cdot v_i = 1 & \forall i \in \Sigma \end{array}$$

The above optimization problem is not in a tractable form. However, if we fix the scaling factor σ_0, then the quantities $\cos(\sigma_0 D_{ij})$ are constants. Moreover, for sufficiently small σ_0, we can use the Taylor series expansion $\cos(h) \approx 1 - \frac{1}{2}h^2$ and the approximation (for small γ) $(1 + \gamma)^2 \approx 1 + 2\gamma$ to obtain $\cos((1 + \gamma) \cdot \sigma_0 D_{ij}) \approx 1 - \frac{1}{2}(1 + 2\gamma)(\sigma_0 D_{ij})^2$. Using these approximations, we obtain the following semidefinite[‡] program:

$$\begin{array}{lll} \text{Min} & \gamma & \\ \text{s.t.} & v_i \cdot v_j \leq 1 - \frac{(1-2\gamma)(\sigma_0 D_{ij})^2}{2} & \forall i, j \in \Sigma \\ & v_i \cdot v_j \geq 1 - \frac{(1+2\gamma)(\sigma_0 D_{ij})^2}{2} & \forall i, j \in \Sigma \\ & v_i \cdot v_i = 1 & \forall i \in \Sigma \end{array}$$

Semidefinite programs can be solved in polynomial time to within any desired precision (Alizadeh, 1995). We solve this program repeatedly for vectors of gradually increasing dimension δ, with $\sigma_0 = 1/\delta$, until we find a feasible solution whose worst-case distortion γ is at most a target

[‡] Technically, the program is not semidefinite because it contains a product of the variables σ_0 and γ. However, we can perform a binary search over γ, solving an SDP each time, to find the smallest value that gives a feasible solution.

value (say, 5%), or until increasing the dimension does not decrease the distortion. Once σ_0 is fixed, we can also optimize the *average* distortion by replacing γ in the constraints for pairs i, j with a variable γ_{ij} and optimizing the objective function $\sum_{i,j} \gamma_{ij}$.

Conversion to Hamming space. To convert our real-valued unit vectors to a Hamming-space embedding, we use the technique of *random hyperplane splitting*. We repeatedly cut the unit sphere with a random hyperplane passing through the origin, then separate the vectors v_i into two sets V^+ and V^- containing those vectors above and below the hyperplane, respectively. We perform T random splits, for a T to be determined. Our Hamming embedding is defined by the results of these splits as follows. For each element $i \in \Sigma$, we define a Hamming-space vector $\phi(i) \in \{0,1\}^T$. If the qth hyperplane places $v_i \in V^+$, we set $\phi(i)[q] = 1$; otherwise, $\phi(i)[q] = 0$.

Observe that a random hyperplane separates the vectors v_i and v_j with probability $\frac{\alpha_{ij}}{\pi}$. Since there are T random splits,

$$\mathbb{E}[d_H(\phi(i), \phi(j))] = \frac{T}{\pi}\alpha_{ij}.$$

By construction, the α_{ij} are proportional to the distances D_{ij} up to small distortion, so the expected Hamming distances between $\phi(i)$ and $\phi(j)$ have the same property. If we can produce a set of Hamming-space vectors $\phi(i)$ whose actual distances are guaranteed to be close to their expectations, then the vectors $\phi(i)$ represent a low-distortion scaled Hamming embedding of D.

Finally, we determine the dimension T of the embedding ϕ, i.e. the number of random hyperplanes to use, so as to ensure that the vectors $\phi(i)$ have distances close to their expectations. Each hyperplane splits the vectors v_i and v_j independently with probability proportional to $\alpha_{ij} \geq \sigma_0 D_{ij}$. By Chernoff bounds, if we use $\Omega(\frac{\log m}{\sigma_0 D_{ij}})$ hyperplanes, then with high probability, the Hamming distance between $\phi(i)$ and $\phi(j)$ is close to its expectation. We can therefore produce a low-distortion Hamming embedding whose dimension is of the order $\frac{\log m}{\sigma_0 \cdot \min_{i,j} D_{ij}}$. The final scaling factor for this embedding is $\sigma = T \cdot \sigma_0$.

Limitations and empirical results. The above embedding construction is a heuristic, largely because we cannot guarantee that the distortion of ϕ is close to the minimum possible for the chosen dimension T. We have devised an alternative algorithm that does make such guarantees, using a different semidefinite program that is provably a relaxation of the scaled hypercube embedding problem. Using that algorithm, we can show that

LEMMA 1. *Given a finite metric that is embeddable in* $\{0,1\}^\delta$ *with distortion* γ*, it is possible to find efficiently an embedding in* $\{0,1\}^T$*, where* $T = O(\sqrt{\delta}\log m)$*, with distortion* $O(\gamma\sqrt{\delta})$.

We omit the alternate construction here because the heuristic described above gives lower distortions in practice.

We applied our embedding strategy to some commonly used protein scoring matrices using SDPPack package (Alizadeh *et al.*, 2002) for MATLAB to solve the semidefinite programs. The sizes of the semidefinite programs are fixed (roughly quadratic in the number of residues m), so solving them is a relatively moderate computational task in practice. The resulting embeddings are summarized in Table 1.

Table 1. Low-distortion embeddings of common score matrices

Matrix	SDP Dim	Total Dim	Max Dist	Mean Dist
BLOSUM-40	200	100000	10.0%	3.2%
BLOSUM-62	40	25000	6.6%	2.4%
BLOSUM-80	200	100000	3.9%	1.9%
PAM-40	200	100000	16.1%	8.7%
PAM-120	200	100000	30.1%	12.3%
PAM-250	200	100000	42.6%	13.1%

Dimension and distortion of embeddings computed for the distance equivalents of some common score matrices. *SDP Dim*: dimension $\delta = 1/\sigma_0$ used in semidefinite programming; *Total Dim*: total dimension T of final Hamming-space embedding.; *Max Dist*: largest percentage distortion of any distance in the matrix; *Mean Dist*: mean percentage distortion over all distances.

The maximum distortion listed for each matrix in the table applies only to the *worst-case* residue pair i, j. A more accurate view of total distortion is given by the *mean* distortion over all residue pairs, which is typically much less than that of the single worst-case pair. For BLOSUM-62 in particular, the mean distortion is quite low, only 2.4%.

We also investigated for BLOSUM-62 the effects of varying the dimension $\delta = 1/\sigma_0$ used in constructing score simulations. No improvement in distortion was observed for $\delta > 40$, while simulations based on semidefinite programs of lower dimension became increasingly distorted (as high as 45% mean distortion for $\delta = 20$). Increasing the number of hyperplanes used to approximate the SDP solution in Hamming space had negligible effect on these distortions. Hence, for BLOSUM-62, the simulation described in Table 1 seems to represent the lowest-distortion embedding obtainable with our method.

Application to similarity search

We now describe how we apply a low-distortion Hamming-space embedding of the peptide distance measure D to create a similarity search algorithm. We use a

modified version of the randomized LSH-ALL-PAIRS-SIM algorithm presented in (Buhler, 2002) for finding similar pairs of sequences under Hamming-embeddable metrics.

The LSH-ALL-PAIRS-SIM Algorithm. Let s be a peptide of length ℓ. Define the vector $\Phi(s)$ to be the concatenation of the mappings of its residues under the embedding ϕ. If ϕ has dimension T, then $|\Phi(s)| = \ell T$. For a pair of ℓ-mers s and t, we have that, since $D(s,t)$ is the sum of the distances between corresponding residues, the Hamming distance $d_H(\Phi(s), \Phi(t)) \approx D(s,t)$. Hence, *we may detect close pairs of ℓ-mers under D by finding pairs of Hamming-space vectors that differ by few substitutions*. The vectors $\Phi(s)$ are large (tens of kilobits), so a search algorithm should *not* store them explicitly for every ℓ-mer in the input.

LSH-ALL-PAIRS-SIM uses random projection (Indyk and Motwani, 1998; Buhler, 2001) to search a large collection of length-ℓ strings for pairs that differ in at most d positions. A *projection* π is a list of k indices between 1 and ℓ to be read from a string. For example, $\{1, 4, 5\}$ is a projection with size $k = 3$ for $\ell \geq 5$. Two vectors *match* under π if they agree in each position read by π.

Random projection detects pairs of strings within a specified distance threshold d as follows. First, choose p projections $\pi_1 \ldots \pi_p$ of size k, for k and p to be determined. For each ℓ-mer s in the input, compute $\pi_j(\Phi(s))$ for $1 \leq j \leq p$. Finally, compute distances $D(s,t)$ between each pair of ℓ-mers s, t whose vectors match under at least one π_j, and return all pairs at distance $\leq d$.

Suppose the positions of the p projections π_j are chosen independently at random with replacement[§]. Given bit vectors of length ℓT, the expected fraction ρ_{fn} of false negatives, i.e. of vector pairs at Hamming distance at most d that do not match under any π_j, is at most

$$\rho_{\text{fn}} \leq \left[1 - \left(\frac{\ell T - d}{\ell T}\right)^k\right]^p. \qquad (1)$$

If the embedding has a scaling factor σ different from one, the dimension T in the above inequality should be replaced by $T' = \frac{T}{\sigma}$. For fixed ℓT and d, we may choose many combinations of k and p so as to achieve a given false negative rate ρ_{fn} (e.g. 5%). Decreasing k requires fewer projections to ensure a low ρ_{fn} but also increases the rate of false positive matches between vectors at distance $> d$, which must be checked and discarded. As described in Buhler (2001, 2002), LSH-ALL-PAIRS-SIM chooses k and p that balance the empirically measured costs of performing p projections and processing the false positives arising from each.

[§] In practice, choosing positions without replacement is more efficient but slightly more complex to analyze.

A key point ensuring the practicality of our approach is that the vectors $\Phi(s)$ need never be represented explicitly in memory. Instead, we sample only k bits (less than 100 in practice) from each vector via the functions $\pi_j \cdot \Phi$. These functions can be computed implicitly on an ℓ-mer given only the vectors $\phi(i)$ for each residue $i \in \Sigma$.

Handling Composition Dependence in D. LSH-ALL-PAIRS-SIM is defined for arbitrary Hamming-embeddable distances. However, we must modify the algorithm to preserve a formal relationship between its distance threshold d and the biologist's choice of score threshold θ for similarity search. Recall that $D(s,t) = \rho(s) + \rho(t) - 2M(s,t)$, where $\rho(s)$ denotes the self-score of a peptide s. Ideally, there would exist a 1:1 mapping from scores to distances such that we could implement a search with score threshold θ by specifying a single distance threshold $\delta(\theta)$. For our D, however, the distance between peptides depends on *both* their score and their residue composition. In particular, the mapping from scores to distances is 1:1 only among pairs s, t for which $\rho(s) + \rho(t)$ is constant.

We address the composition dependence of D by *binning* the input ℓ-mers by self-score. Let C_1 and C_2 denote two sequence collections to be compared (e.g. the proteomes of two organisms). We first divide the ℓ-mers of each collection into bins so that the ℓ-mers in bin b have constant self-score ρ_b. We then separately compare each pair of bins b_1 from C_1 and b_2 from C_2 using LSH-ALL-PAIRS with distance threshold $d = \rho_{b_1} + \rho_{b_2} - 2\theta$, where θ is the desired score threshold. We compute the parameters k and p independently for each pair of bins to preserve the algorithm's guarantee of high expected sensitivity, up to the (small) distortion introduced by the embedding.

The computational overhead of binning can be substantial, rising quadratically with the number of possible self-scores for peptides in the input. To limit the number of bins, we group ℓ-mers with similar but not identical self-scores. A bin b covering a range of self-scores $[\rho_l, \rho_h]$ is treated as if all its ℓ-mers had the highest self-score ρ_h. Because higher self-scores translate into higher distance thresholds d for a given score threshold θ, and higher distance thresholds require more computation for a given ρ_{fn}, LSH-ALL-PAIRS may overestimate the amount of computation needed to detect matches involving some ℓ-mers in b. However, the target sensitivity ρ_{fn} is not compromised. In practice, dividing the range of possible self-scores for an ℓ-mer equally into 5-10 bins, even with the extra work caused by binning multiple self-scores, is substantially less expensive than restricting each bin to a single self-score value.

EXPERIMENTAL RESULTS

We implemented the algorithm of the previous section in C++ to investigate the performance of our distance

Table 2. Comparison of old and new LSH-ALL-PAIRS-SIM algorithms

Method	ℓ	θ	ℓ-mer Matches	Time (minutes)	Aligned Pairs
(Buhler, 2002)	10	40	48586	147.5	6000
	20	47	62301	587.2	7707
	30	50	61580	1260.8	7961
This work	10	40	46387	17.6	5876
	20	47	59862	168.1	7699
	30	50	58661	461.3	7959

Relative speed and sensitivity of LSH-ALL-PAIRS-SIM with our distance measure and embedding versus the technique described in Buhler (2002). Score thresholds were chosen for each ℓ to achieve an expected spurious match rate of $< 10^4$. *Aligned pairs*: number of pairs of proteins that were aligned at least once with E-value $\leq 10^{-10}$ after gapped extension. Times were measured on a 1 GHz Intel Pentium III machine.

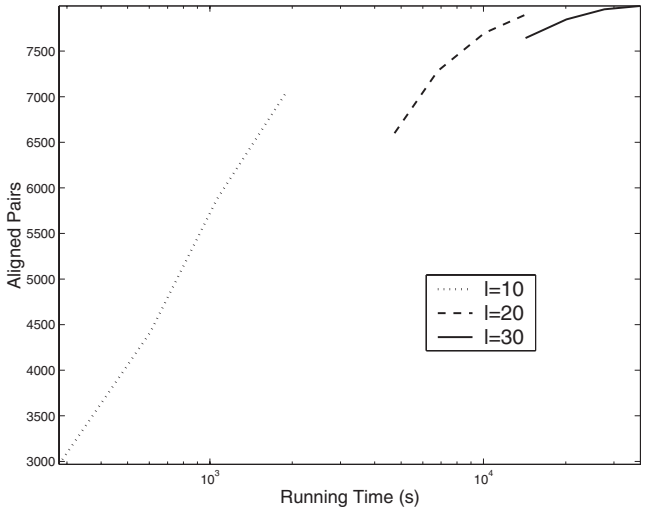

Fig. 1. Performance trade-offs achieved for the new LSH-ALL-PAIRS-SIM variant in bacterial protein comparison. We tested $\ell \in \{10, 20, 30\}$ and expected spurious match rates between 10^2 and 10^5. *Aligned pairs* is defined as in Table 2.

function and embedding. Experiments on a real proteomic comparison demonstrate that our new method runs significantly faster than the 'old' algorithm of (Buhler, 2002) with comparable sensitivity.

We tested LSH-ALL-PAIRS-SIM with the old and new distance measures and embeddings on a proteome-to-proteome comparison of the bacteria *E.coli* and *V.cholerae*. The total size of this comparison was roughly 1.35 million by 1.16 million amino acids. We used the BLOSUM-62 scoring function, the default used by NCBI BLASTP for protein comparison. For the old algorithm, the embedding was as described in Buhler (2002) (dimension $T = 17$), while for the new algorithm, we used our novel embedding (scaled dimension $T' = 50$). Further experimental details are given in the Appendix.

We used ungapped similarities found by each algorithm to initiate gapped extension using affine Smith-Waterman with BLAST's default gap penalties for BLOSUM-62. The resulting set of gapped alignments may fairly be compared to the output of BLASTP.

Table 2 shows the relative sensitivities and running times of LSH-ALL-PAIRS-SIM with the old and new embeddings. We tested the algorithm with peptide lengths ℓ of 10, 20, and 30; for each length, we chose a score threshold θ to produce no more than 10^4 expected spurious matches between unrelated peptides in the entire comparison. (Further performance characterization of the new method for other spurious match rates is shown in Fig. 1). For all values of ℓ, the new embedding yielded an algorithm whose sensitivity for raw ℓ-mer matches is within 5% of the old algorithm, while requiring substantially less running time (one-eighth as much for $\ell = 10$). The sensitivity for aligned pairs of high-scoring sequences after gapped extension is even closer to that of the old method: within roughly 2% for each ℓ tested.

The improved running time of the new distance function and embedding can be traced to its need for fewer projections, and hence fewer passes over the input sequences, to achieve (roughly) the target sensitivity level $\rho_{\text{fn}} \leq 0.05$. This reduction implies that the new method can more efficiently distinguish between pairs of ℓ-mers with high score and pairs with low score. The distance mapping of (Buhler, 2002) essentially had $D(s,t) = \ell T - M(s,t)$, which for large ℓ and/or T compressed the possible distances between ℓ-mers into a small range far from zero, even when $s = t$. In contrast, our mapping always has $D(s,s) = 0$ and utilizes a larger distance range, increasing the separation in distance between high-scoring and low-scoring ℓ-mer pairs. Hence, these two types of pairs are computationally easier to separate with high probability.

Finally, we compared the performance of our distance-based search algorithm to that of the seed-based NCBI BLASTP. The new algorithm's improved running time significantly narrows the gap between LSH-ALL-PAIRS-SIM's speed and that of BLASTP, which found significant alignments ($E \leq 10^{-10}$) between 9613 pairs of proteins in about 10 minutes. Further speed improvements might be achieved by carefully lowering the effective dimension T' of the embedding at the cost of small extra distortion in D. The ability to trade slightly higher distortion for speed is a new feature of our approach not found in the exact isometry of (Buhler, 2002).

CONCLUSIONS AND FUTURE WORK

We have proposed a new mapping of the BLOSUM protein similarity matrices into distance metrics that can be efficiently embedded with little distortion in Hamming space. These metrics, combined with the LSH-ALL-PAIRS-SIM framework, yield a new distance-based algorithm to detect similarities in protein databases. The new algorithm improves substantially on its predecessor in efficiency, particularly for small ℓ, without sacrificing sensitivity.

Our framework for generating and using embeddings from amino acid score matrices suggests a number of avenues for future study. Firstly, we can modify the embedding strategy or the distance function D for improved performance. For example, replacing D with D^n for $n > 1$ could increase the distance gap between low- and high-scoring pairs of peptides, making these pairs easier to distinguish. Secondly, because our embedding construction works (with increased distortion) on distances that violate the triangle inequality, it also extends to the PAM matrices, for which D incurs a larger number of such violations. Table 1 gives some sample embeddings for common PAM matrices, but we must still carefully explore our algorithm's performance with these embeddings in practice. Most importantly, we can seek more desirable points in the trade-off between distortion of the original score matrix M and speed/sensitivity of the resulting search algorithm. FLASH, for example, represents an extreme of this trade-off that accepts high distortion in exchange for fast running times even for $\ell = 100$. Exploring the continuum of distortion versus efficiency should lead to more practical search tools that incorporate the precise distinctions of score matrices yet can blur them in a controlled way to produce fast indexing and filtering schemes for protein databases.

ACKNOWLEDGEMENTS

The first, third and fourth authors were supported in part by NSF grants CCR-9820951 and CCR-0121555 and DARPA cooperative agreement F30602-00-2-0601.

REFERENCES

Alizadeh,F. (1995) Interior point methods in semidefinite programming with applications to combinatorial optimization. *SIAM J. Optimization*, **5**, 13–51.

Alizadeh,F., Haeberly,J.-P.A, Nayakkankuppam,M.V., Overton,M.L. and Schmieta,S. (2002) SDPack version 0.9 beta for Matlab 5.0: semidefinite quadratic linearly constrained programs, http://www.cs.nyu.edu/cs/faculty/overton/sdppack/sdppack.html

Altschul,S.F. (1996) Local alignment statistics. *Meth. Enzymol.*, **266**, 460–480.

Altschul,S.F., Madden,T.L., Schäffer,A.A., Zhang,J., Zhang,Z., Miller,W. and Lipman,D.J. (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res.*, **25(17)**, 3389–3402.

Bourgain,J. (1986) The metrical interpretation of superreflexivity in Banach spaces. *Israeli J. Math.*, **56(2)**, 222–230.

Buhler,J. (2001) Efficient large-scale sequence comparison by locality-sensitive hashing. *Bioinformatics*, **17**, 419–428.

Buhler,J. (2002) Provably sensitive indexing strategies for biosequence similarity search. *Proceedings of the 6th Annual International Conference on Computational Molecular Biology (RECOMB02)*. pp. 69–76.

Califano,A. and Rigoutsos,I. (1993) FLASH: a fast look-up algorithm for string homology. *Proceedings of the 1st International Conference on Intelligent Systems for Molecular Biology (ISMB93)*. pp. 56–64.

Dayhoff,M.O., Schwartz,R. and Orcutt,B.C. (1978) A model of evolutionary change in proteins. *Atlas of Protein Sequence and Structure*, **5**, 345–352.

Giladi,E., Walker,M.G., Wang,J.Z. and Volkmuth,W. (2002) SST: an algorithm for finding near-exact sequence matches in time proportional to the logarithm of the database size. *Bioinformatics*, **18**, 873–877.

Goemans,M.X. and Williamson,D.P. (1995) Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, **42**, 1115–1145.

Henikoff,S. and Henikoff,J. (1992) Amino acid substitution matrices from protein blocks. *Proc. Natl Acad. Sci. USA*, **89**, 10915–10919.

Indyk,P. and Motwani,R. (1998) Approximate nearest neighbors: towards removing the curse of dimensionality. *Proceedings of the 30th Annual ACM Symposium Theory of Computing (STOC98)*.

Kent,W.J. (2002) BLAT: the BLAST-like alignment tool. *Genome Res.*, **12**, 656–664.

Linial,M., Linial,N., Tishby,N. and Yona,G. (1997) Global self organization of all known protein sequences reveals inherent biological signatures. *J. Mol. Biol.*, **268**, 539–556.

Linial,N., London,E. and Rabinovich,Y. (1995) The geometry of graphs and some of its algorithmic applications. *Combinatorica*, **15(2)**, 215–245.

Ning,Z., Cox,A.J. and Mullikin,J.C. (2001) SSAHA: a fast search method for large DNA databases. *Genome Res.*, **11**, 1725–1729.

Wooton,J.C. and Federhen,S. (1993) Statistics of local complexity in amino acid sequences and sequence databases. *Comput. Chem.*, **17**, 149–163.

APPENDIX: BACTERIAL PROTEIN COMPARISON

The *E.coli* (strain K12) and *V.cholerae* proteomes were obtained from the NCBI Genbank FTP site. The two proteomes totaled 8117 sequences containing 2.51 million amino acids. Both proteomes were masked to remove low-complexity regions using Wooten and Federhen's Seg program (Wooton and Federhen, 1993) with parameters 12/1.8/2.0, as recommended by its documentation for database-database comparisons.

Analyses were run using LSH-ALL-PAIRS-SIM with embeddings of distances based on the BLOSUM-62 score

matrix, with lengths ℓ and score thresholds θ as given in Table 2 and $\rho_{\text{fn}} \leq 5\%$. ℓ-mer pairs found by the algorithm were used to seed ungapped and gapped extension, producing gapped alignments of variable length. We used BLASTP's default gap penalties $(-11, -1)$ and its estimates of the Karlin-Altschul parameters K and λ to assign significance to each similarity. Alignments were filtered for significance at $E < 10^{-10}$.

NCBI's BLASTP 2.2.3 was run on the same sequences with its default parameters.