DIMACS Technical Report 2004-55
November 2004

Optimal Protein Encoding

by

Endre Boros[1]
RUTCOR
Rutgers University
Piscataway, New Jersey 08854

Logan Everett[2]
DIMACS
Rutgers University
Piscataway, New Jersey 08855

[1]Permanent Member
[2]NSF-REU Program Participant

ABSTRACT

A common task in Bioinformatics is the search of sequence databases for matching sequences. In protein sequence databases, searching is hindered by both the increased amount of data and the complexity of sequence similarity metrics. Protein similarity is not simply a matter of character matching, but rather is determined by a matrix of scores assigned to every match and mismatch [5]. One strategy to increase search speed is to map sequences into a binary space where the Hamming distance between strings is comparable to the similarities of the original sequences [4]. Within binary Hamming spaces, statistically proven sampling methods can be used for fast, reliably sensitive searches [2]. However, mapping the protein alphabet to a binary Hamming space often comes with a certain level of distortion. We have employed Linear Programming techniques to model and study the nature of these mapping schemes. Specifically, we have found the theoretically minimum distortion achievable for several biological scoring matrices, as well as corresponding optimal encoding weights. We have also analyzed the use of these encoding weights to generate pseudo-random binary encodings that approach the theoretically minimum distortions.

1 Introduction

A precise method for encoding symbols in the protein alphabet to binary strings is useful for faster, more accurate searches of protein databases [4]. Similarity of Protein Strings is based on summing the scores of each corresponding pair. Each score comes from a matrix of scores acquired from biological data. The most used scoring matrices are the PAM Series and BLOSUM Series [5]. Due to the size of Protein Databases and the complexity of looking up the score of each corresponding pair, searching Protein Databases is an exhaustive task.

A general concept for more efficient searches is to map each symbol in the protein alphabet to a binary string of constant length ℓ. Through such a mapping, an entire database of protein strings can be encoded into binary strings. The Hamming distances between these binary strings can be reliably approximated by using appropriate sampling methods [3]. However, these mappings are only useful if the Hamming distances between binary strings are representative of the similarity scores from a scoring matrix. An exact representation is not always possible, but those with less distortion should provide more accurate searches [4].

We have created a precise mathematical model that covers all possible encoding schemes and used it to solve for a scheme with the least distortion possible. In this report, we will discuss our model in precise terms. We will then discuss our analysis work done to observe its effectiveness. The rest of the paper describes our exact implementation of the model and some future work.

2 Encoding model

Let $\Sigma = \{a_1, ..., a_n\}$ denote a finite alphabet of size n, and let us assume that a symmetric "similarity" matrix $M \in \mathbb{R}^{n \times n}$ is given. We shall view the entries $M_{ij} = M_{ji}$ of this matrix as a measure of similarity of symbols a_i and a_j. In particular, in the biological application we consider in this paper, we have $n = 20$ and each symbol in the alphabet Σ correspond to an amino acid. Among the similarity scoring matrices, known and used in the literature, we shall consider the PAM and BLOSUM series (see e.g., [5]).

Following the methodology proposed in [4], we shall consider the problem of representing the given alphabet Σ by a set of n binary vectors, i.e., a mapping

$$\phi : \Sigma \longrightarrow \{0,1\}^{\ell}, \tag{1}$$

called an *encoding scheme*. For the sake of simplicity, we shall use the notation $\phi(a_i) = \phi_i = (\phi_{i1}, \phi_{i2}, \ldots, \phi_{i\ell}) \in \{0,1\}^{\ell}$, and let us recall that the *Hamming distance*

$$H(\phi_i, \phi_j) = |\{k \mid \phi_{ik} \neq \phi_{jk}\}|$$

is simply the number of coordinates in which the two binary vectors differ.

Our goal is to find an encoding scheme ϕ for which ℓ is as small as possible, and for which the pairwise Hamming distances $H(\phi_i, \phi_j)$, $1 \leq i < j \leq n$ represent the similarities of the corresponding symbols, as given in the scoring matrix M.

To be able to formulate this problem, and in particular our objective, more precisely, let us convert the similarity scores in a given scoring matrix M to distance values, i.e., we consider

$$D_{ij} = M_{ii} + M_{jj} - 2M_{ij}, \tag{2}$$

for $1 \leq i, j \leq n$, as proposed in [4]. Since M can be assumed to be symmetric (this assumption holds true for both the PAM and BLOSUM series), we obtain a symmetric distance matrix $D \in \mathbb{R}^{n \times n}$ (i.e., in which $D_{ij} = D_{ji}$ holds for all $i \neq j$). Let us note that we have $D_{ii} = 0$ for all $i = 1, ..., n$, that is that the "distance" of a symbol from itself is zero, which is a quite natural assumption, on the one hand. On the other hand however, the similarity scores M_{ii} are not all the same in the PAM or BLOSUM matrices, thus transforming these possibly different values all to zero seem to lead to a slight loss of information (in [2] a double encoding scheme was proposed to compensate for this information loss, leading to substantially larger formulations – in this paper we followed the approach proposed in [4], and leave it for future research to develop computationally efficient methods to handle the computationally more difficult double encoding scheme). Let us also note that in all considered scoring matrices (in the PAM and BLOSUM series) we have nonnegative diagonal entries, and all off-diagonal entries are either nonpositive, or have smaller absolute value than the diagonal entry, resulting in nonnegative "distance" values by (2).

To eliminate redundancies, we shall represent D by the so called *Real Distance Vector* $\mathbf{d}$, defined by

$$\mathbf{d} = (D_{ij} \mid 1 \leq i < j \leq n).$$

Note that $\mathbf{d} \in \mathbb{R}^{\binom{n}{2}}$.

Analogously, let us introduce the notation $H_{ij} = H_{ji} = H(\phi_i, \phi_j)$ for $i \neq j$ for a given encoding scheme (1), and consider the so called *Hamming Distance Vector* defined by

$$\mathbf{h}(\phi) = (H_{ij} \mid 1 \leq i < j \leq n).$$

An encoding scheme ϕ of (1) is said to represent the similarity matrix M with *distortion* ε (see [4]) for some $0 \leq \varepsilon \leq 1$, if there exists a positive scalar λ such that the inequalities

$$\mathbf{d}(1 - \varepsilon) \leq \lambda \mathbf{h}(\phi) \leq \mathbf{d}(1 + \varepsilon) \tag{3}$$

hold, where the inequalities above are meant componentwise (i.e., where $D_{ij}(1-\varepsilon) \leq \lambda H_{ij} \leq D_{ij}(1+\varepsilon)$ hold for all $1 \leq i < j \leq n$).

Ideally, we would like $\mathbf{h}(\phi)$ to be scalable to $\mathbf{d}$ by a single real value $\lambda \geq 0$, or in other words, we would prefer a solution with $\varepsilon = 0$ distortion. In practice this may not be achievable however, since H represents a distance matrix, corresponding to Hamming distances between binary vectors, while D is derived from a given scoring matrix M, and may not necessarily correspond to any reasonable notion of "distance". Let us add that for the scoring matrices considered in this paper, in fact D satisfies the triangle inequalities $D_{ij} + D_{jk} \geq D_{ik}$ for almost all triples i, j, k of indices, indicating that we should be able to obtain encoding schemes with small distortion values (see also [4]).

It is also clear intuitively that allowing longer binary vectors in our encoding (larger ℓ values), we should be able to achieve better representation (smaller ε values in (3)), implying that the goal of minimizing both ε and ℓ is perhaps contradictory.

In this paper we study the problem of determining the smallest possible value for ε, when no limit on ℓ is imposed. Furthermore, we also study techniques to approximate the best possible value for ε when some limits are imposed on ℓ.

Our approach is based on a linear programming reformulation of (3), presented in the next section.

3 A linear programming reformulation

To arrive to a mathematical programming formulation of our problem, let us consider an encoding scheme ϕ, and associate to it the binary vectors

$$\mathbf{b}^k = (\phi_{ik} \mid i = 1, ..., n), \tag{4}$$

for $k = 1, ..., \ell$. Let us observe that given the ordered collection of binary vectors $\mathcal{B} = \{\mathbf{b}^k \in \{0,1\}^n \mid k = 1, ..., \ell\}$, we can reconstruct in a unique way from $\mathcal{B}$ the mappings $\phi_i \in \{0,1\}^\ell$, $i = 1, .., n$ such that (4) hold for all $k = 1, ..., \ell$. Let us also note that in fact the order of these binary vectors can be changed arbitrarily, without changing their Hamming distances. In other words, if $\mathcal{B}' = \{\mathbf{b}^{\pi(k)} \mid k = 1, ..., \ell\}$ for some permutation π of $\{1, ..., \ell\}$, and ϕ'_i, $i = 1, ..., n$ are the binary mappings corresponding to $\mathcal{B}'$, then we have $H(\phi_i, \phi_j) = H(\phi'_i, \phi'_j)$ for all $1 \leq i < j \leq n$. Thus, we can equivalently specify (an equivalent class of) the mappings ϕ_i, $i = 1, ..., n$, by specifying the number of occurrences of each binary vector $\mathbf{b} \in \{0,1\}^n$ in the ordered collection $\mathcal{B}$.

Let us also notice that if we replace a binary vector $\mathbf{b} \in \mathcal{B}$ by its componentwise complement (i.e., the vector $\mathbf{b} = (b_1, b_2, ..., b_n)$ by $\overline{\mathbf{b}} = (\bar{b}_1, \bar{b}_2, ..., \bar{b}_n)$, where $\bar{x} = 1 - x$), then again the pairwise Hamming distances between the corresponding mappings remain the same. Hence, we can also assume, without any loss of generality that only one of any complementary pair of vectors occur with a positive multiplicity in the collection $\mathcal{B}$. For instance, we can assume that the multiplicity of $\mathbf{b} \in \{0,1\}^n$ in $\mathcal{B}$ can be positive only if $b_1 = 1$. Furthermore, since the vector $\mathbf{b} = (1, 1, ..., 1)$ does not contribute anything to the pairwise Hamming distances in the corresponding encoding scheme, we can also assume without any loss of generality that the multiplicity of this vector in $\mathcal{B}$ is zero.

Since the vectors of $\mathbf{b} \in \{0,1\}^n$ can equivalently be described by the associated sets

$$S = S(\mathbf{b}) = \{i \mid b_i = 1\} \subseteq [n],$$

we can reformulate our problem in terms of integer multiplicities associated to subsets $S \subseteq [n]$ for which $1 \in S$ and $S \neq [n]$. Let us denote by

$$\Omega \;=\; \{S \subseteq [n] \mid 1 \in S,\; S \neq [n]\}.$$

To every such set $S \in \Omega$, let us associate a binary vector $a_S = (a_{ij} \mid 1 \le i < j \le n)$ where

$$a_{ij} = \begin{cases} 1 & \text{if } |S \cap \{i,j\}| = 1, \\ 0 & \text{otherwise.} \end{cases}$$

Clearly, the component a_{ij} represent exactly the contribution of vector $\mathbf{b}$ (with $S(\mathbf{b}) = S$) to the Hamming distance of ϕ_i and ϕ_j in the corresponding encoding scheme. Let us denote by A the matrix, the columns of which are the vectors a_S, for $S \in \Omega$. Let us also denote by $\mathbf{y} = (y_S \mid S \in \Omega)$ the vector of multiplicities of the corresponding binary vectors in the collection $\mathcal{B}$, which according to the above, defines the encoding scheme. Then problem (3) can be reformulated as the problem of finding reals ε, $\lambda > 0$ and integers y_S for $S \in \Omega$ satisfying the inequalities

$$\mathbf{d}(1-\varepsilon) \le \lambda A\mathbf{y} \le \mathbf{d}(1+\varepsilon) \tag{5}$$

For any solution of this problem, we have the corresponding value of ℓ defined uniquely by

$$\ell = \sum_{S \in \Omega} y_S.$$

More specifically, the problem of finding the smallest distortion, assuming that $\ell \le L$, where L is a given upper bound, can be formulated as the mathematical programming problem

$$\begin{gathered} \min \quad \varepsilon \\ \text{subject to the constraints} \\ \mathbf{d}(1-\varepsilon) \le \lambda A\mathbf{y} \le \mathbf{d}(1+\varepsilon) \\ \sum_{S \in \Omega} y_S \le L \\ \lambda > 0, \text{ and } y_S \in \mathbb{Z}_+ \text{ for } S \in \Omega. \end{gathered} \tag{6}$$

This formulation has several drawbacks, including the large number of integer variables $(2^{n-1}-1)$, and the nonlinearity in its constraint set (both the y_S variables and λ are unknowns). Even though, in the considered application we have only $n = 20$, the number of variables is "only" about $500,000$. This fact coupled with the integrality of these variables and the nonlinearity in the constraint set, results in a problem which is perhaps still too difficult to solve exactly with currently existing techniques and software.

In this paper we study a relaxation of this problem, in which we disregard the upper bound on the value of ℓ and the integrality of the y_S variables. Noticing that λ appears only in products with the y_S variables, let us introduce

$$x_S = \lambda y_S \quad \text{for} \quad S \in \Omega,$$

and denoting by $\mathbf{x} = (x_S \mid S \in \Omega)$ the vector of the variables defined in this way, we can formulate a linear programming relaxation of problem (6) as follows:

$$\min \quad \varepsilon$$

subject to the constraints

$$\begin{aligned} A\mathbf{x} + \varepsilon\mathbf{d} &\geq \mathbf{d} \\ -A\mathbf{x} + \varepsilon\mathbf{d} &\geq -\mathbf{d} \\ x_S \in \mathbb{R}_+ \text{ for } S \in \Omega. & \end{aligned} \tag{7}$$

Let us note that (7) is a relaxation of (6), i.e., from any solution of the latter we can get a solution for the first one having the same objective function value, by simply defining $x_S = \lambda y_S$ for all subsets $S \in \Omega$. Thus, the optimum value of (7) is a lower bound of the optimum value of (6), regardless of the limit L. Denoting by ε^* the optimum value of (7), and by $\widehat{\varepsilon}(L)$ the optimum of (6) as a functions of L, we can thus claim

$$\varepsilon^* \leq \widehat{\varepsilon}(L) \quad \text{for all} \quad L \in \mathbb{Z}_+. \tag{8}$$

Let us remark that we can also claim that

$$\varepsilon^* \;=\; \lim_{L \longrightarrow \infty} \widehat{\varepsilon}(L). \tag{9}$$

To see this latter claim, let us note first that by the definition of problem (6), we have

$$\widehat{\varepsilon}(k+1) \;\leq\; \widehat{\varepsilon}(k) \qquad \text{for all integers} \qquad k = 1, 2, ...$$

from which, together with (8) the existence of the limit in (9) follows, together with the inequality $\varepsilon^* \leq \lim_{L \longrightarrow \infty} \widehat{\varepsilon}(L)$. To see the equality, let us note that in problem (7) we have $\varepsilon^* \geq 0$, implying the existence of a finite optimum. Since in the problem the matrix A has integral coefficients and the real distance vector $\mathbf{d}$ can also be assumed to have rational entries, the optimum of (7) is attained by a rational assignment x_S^* for $S \in \Omega$. Denoting by Q a common multiple of the denominators in the rational expressions of these variables, it follows then that the equations $y_S^* = Qx_S$ for $S \in \Omega$ define an integral solution of (5) with some appropriate value of λ and with $\varepsilon = \varepsilon^*$. Hence, we can construct an encoding scheme achieving ε^* distortion from these y_S^* values, implying that for $L \geq \sum_{S \in \Omega} y_S^*$ we have $\widehat{\varepsilon}(L) \leq \varepsilon^*$ which together with (8) proves (9).

4 Encoding schemes of given dimension

The above arguments show that from the optimal solution of the linear programming formulation (7) we can arrive to an encoding scheme achieving the theoretically best possible distortion ε^*. However, in practice Q might be way too large, resulting in a very high dimensional binary encoding, negating the practical usefulness of the approach. We shall show below some possible approaches to derive from the optimal solution of (7) encoding schemes

of some prescribed dimensionality ℓ. Let us denote, as above, by x^*_S, $S \in \Omega$ the optimal solution of problem (7) and let ε^* be its optimum value.

In the first approach, we set $X = \sum_{S\in\Omega} x^*_S$, and for a fixed value of ℓ we set $y^1_S = \lfloor \frac{\ell}{X} x^*_S \rfloor$ for all $S \in \Omega$. These integer multiplicities yield an encoding scheme ϕ^1_i, $i = 1, ..., n$ of dimension $\ell' = \sum_{S\in\Omega} y^1_S$. Let us denote by $\varepsilon^1(\ell)$ the distortion achieved by this encoding scheme. Since the number of non-zero values in the optimal solution of (7) is limited by the rank of the system, i.e., by $2\binom{n}{2} = n(n-1)$, we have

$$\ell - n(n-1) \;\leq\; \ell' \;\leq\; \ell.$$

We claim that

$$\varepsilon^* \;=\; \lim_{k\to\infty} \varepsilon^1(k),$$

which follows readily from the proof of (9), since whenever ℓ/X is an integer multiple of Q, the obtained integral assignment $y^1_S = \frac{\ell}{X} x^*_S$ for $S \in \Omega$ yields an encoding scheme achieving the best possible ε^* distortion.

The drawback of this approach is that ℓ' may be different from ℓ (by at most $n(n-1)$), implying that $\varepsilon^1(k)$ is not necessarily a monotone function of k.

As a second approach, we may try to fix the above problem, by introducing independent random binary parameters $z_S \in \{0, 1\}$ for $S \in \Omega$, such that $Prob(z_S = 1) = \frac{\ell}{X} x^*_S - \lfloor \frac{\ell}{X} x^*_S \rfloor$, and setting $y^2_S = y^1_S + z_S$ for all $S \in \Omega$. This approach results in an encoding length $\ell'' = \sum_{S\in\Omega} y^2_S$. Clearly, we have $\ell - n(n-1) \leq \ell'' \leq \ell + n(n-1)$ as worst case bounds. However, we also have that ℓ'' itself is a random variable, and its expected value is ℓ, suggesting that the corresponding distortion value, which we denote by $\varepsilon^2(\ell)$ behaving more smoothly, as a function of ℓ, than $\varepsilon^1(\ell)$. Similarly to the previous case, it can be seen that

$$\varepsilon^* \;=\; \lim_{k\to\infty} \varepsilon^2(k).$$

Finally, in a third approach we choose everything randomly, and only the distribution of the random variables depend on the optimal solution of (7). More precisely, let us generate a random sequence $(S_1, S_2, ...)$ of subsets from Ω, such that for each k the set S_k is chosen independently from the previous ones, and such that $Prob(S_k = S) = \frac{x^*_S}{X}$ for all $S \in \Omega$. Let us then define for every given value ℓ the integer variables

$$y^3_S = |\{k \mid S = S_k, 1 \leq k \leq \ell\}| \quad \text{for} \quad S \in \Omega,$$

i.e., the frequency of the set S in the subsequence $(S_1, S_2, ..., S_\ell)$. Let us further denote by $\varepsilon^3(\ell)$ the distortion achieved by the corresponding encoding scheme. The advantage of this approach is that the obtained encoding scheme has exactly ℓ as its dimension. The drawback of the approach is that $\varepsilon^3(\ell)$ may be even less monotone, as a function of ℓ, than those obtained by the previous approaches. Still, it can be shown easily that we have

$$\varepsilon^* \;=\; \lim_{k\to\infty} \varepsilon^3(k).$$

5 Computational results

We have tried the approach described above with various scoring matrices available online or from other publications, including the BLOSUM series (40,45,62 and 80) and the PAM series (40,120 and 250) (see e.g., the web-site [1]).

5.1 Solving the linear programming problem (7)

For all seven data sets we have computed first the real distance matrix by (2), and then solved the linear programming problem (7). Since in all these data sets we have $n = 20$, the number of variables in each of these LP-s is $N = 1 + |\Omega| = 2^{n-1} = 524,288$, while the number of constraints is $M = 2\binom{n}{2} = n(n-1) = 380$. For our computations we have used a Linux system with two Xeon 3.06GHz processors (though our computations utilized only one of those processors) and with 3GB RAM. All codes were developed in C++, and for solving the LP-s we used the CPLEX 8.1.1 library.

Let us remark that due to the large number of the columns in these problems, originally we considered applying a column generation technique. It can be seen that despite the very structured nature of the coefficient matrix, the column generation subtask is equivalent with an unconstrained binary optimization problem, which is an NP-hard optimization problem in general. Even though in this particular case we would have needed to solve such problem in $n = 20$ binary variables (a certainly tractable task computationally), it still could imply a lengthy computation due to the possibly large number of columns to be generated. Since we had access to a computer with appropriate memory, we decided rather to generate the whole coefficient matrix of (7) at once, and solve the linear programming problem in one round.

As expected, just generating this $380 \times 524,288$ matrix and saving it in MPS format required typically about 20min of computing time. The first surprise for us was that after this, reading this matrix from an auxiliary file, and solving the LP (7) by CPLEX took only a few (always less than 5) minutes. A second and even less expected surprise was that for

Table 1: Optimal distortion values.

Data Set	Distortion values from the LP optima		
	Maximum=ε^*	Median	Average
BLOSUM 40	0.07692	0.07692	0.06829
BLOSUM 45	0.00000	0.00000	0.00000
BLOSUM 62	0.00000	0.00000	0.00000
BLOSUM 80	0.00000	0.00000	0.00000
PAM 40	0.05797	0.05797	0.05505
PAM 120	0.16667	0.16667	0.13030
PAM 250	0.25000	0.21719	0.17958

several of these seven data sets we got $\varepsilon^* = 0$ as an optimum value, i.e., many of these distance matrices can be embedded into a Hamming space (of appropriately large dimension) with

zero distortion, see Table 1. We also included in this table the median and average distortions over the $190 = \binom{20}{2}$ pairs of distances, computed from the LP optima. These results confirm the findings of [4], in which a semidefinite programming based approximation was used to construct binary embeddings, and in which the obtained distortion values were unpractically high for some of the data sets, in particular for PAM 120 and PAM 250. Our results show that for these data sets even the theoretically best distortion values are high.

On a companion web-site [6] we provide the used distance matrices (derived from the similarity scores by (2)), and the non-zero components of the optimal basic solution to (7).

To test the sensitivity of this approach, since the used similarity scores are themselves debated in the literature, we decided to apply random perturbation to the distance matrix entries (generating randomly those entries within $\alpha\%$ of the original values, for $\alpha = 1, 2, 5, 10, 25, ...$) and applied the same model for this perturbed matrices. The results, which also can be found on the companion web-site [6], show a reasonably large robustness, and even 5-10% perturbation did not change much of the optimum value.

5.2 Maximum distortion rates of obtained binary encodings

We have computed from the obtained optimal solutions the corresponding binary encodings by all three methods described in the previous section. All results in excel files and graphs of the $\varepsilon^i(\ell)$ functions, for $i = 1, 2, 3$ can be found at the web-site [6].

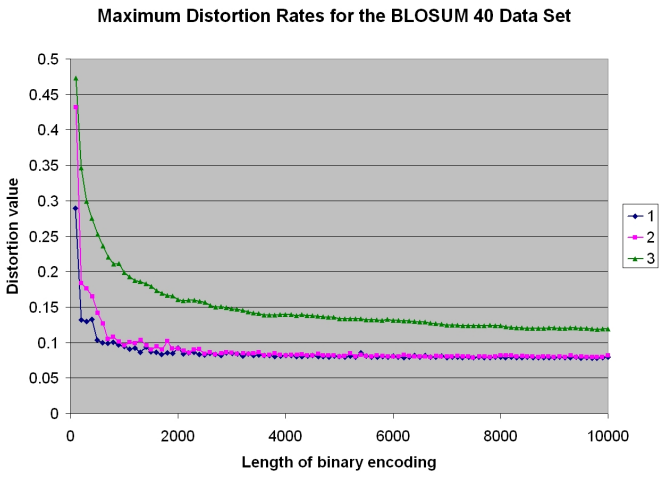

Figure 1: Maximum distortion values ε^1, ε^2 and ε^3, as functions of encoding length ℓ, obtained from the LP optimum for the BLOSUM 40 data set.

To compare the three methods for deriving an encoding scheme from the LP optimum, we included, as illustrations in Figures 1, 2 and 3, the graphs $\varepsilon^i(\ell)$, $i = 1, 2, 3$ for data sets BLOSUM 40, 80 and PAM 120. It can be seen from these (and from all other results, included at the companion web-site) that the fully randomized third method produces the

slowest converging series, and even for large values of ℓ the actual value of $\varepsilon^3(\ell)$ is still significantly larger than ε^*. On the other hand, both of the first two methods produce about the same, fast converging series, and we have $\varepsilon^1(\ell) \approx \varepsilon^2(\ell) \approx \varepsilon^*$ already for $\ell \geq 4000$. In fact, the fully deterministic first method seem to to produce the best results (though the difference between the first and second approach is not significant), and certainly it is the simplest and fastest to compute. Based on these results, the fully deterministic first approach can perhaps be recommended for practical use.

Table 2: Maximal distortion values for various encoding lengths, obtained by the fully deterministic approach.

ℓ	$\delta^1(\ell)$						
	BLOSUM				PAM		
	40	45	62	80	40	120	250
500	0.1037	0.0585	0.0461	0.0561	0.0955	0.2101	0.2923
1000	0.0946	0.0274	0.0202	0.0158	0.0712	0.1779	0.2672
2000	0.0833	0.0136	0.0107	0.0085	0.0638	0.1727	0.2577
3000	0.0819	0.0093	0.0063	0.0055	0.0633	0.1715	0.2552
4000	0.0802	0.0058	0.0056	0.0053	0.0616	0.1703	0.2535
5000	0.0795	0.0044	0.0041	0.0036	0.0609	0.1692	0.2535
6000	0.0792	0.0039	0.0032	0.0032	0.0598	0.1684	0.2527
7000	0.0788	0.0034	0.0022	0.0025	0.0598	0.1683	0.2527
8000	0.0787	0.0029	0.0022	0.0019	0.0596	0.1683	0.2516
9000	0.0785	0.0026	0.0019	0.0019	0.0596	0.1680	0.2516
10000	0.0781	0.0025	0.0019	0.0019	0.0593	0.1678	0.2516
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
∞	0.0769	0.0000	0.0000	0.0000	0.0580	0.1667	0.2500

In Table 2 we included the maximum distortion values, $\varepsilon^1(\ell)$, obtained by the first encoding method, for various encoding lengths. Since we computed these values for all values of ℓ in the range $100 \leq \ell \leq 10,000$ (and the excel files at the web-site include those values for all multiples of 100), we included in this table

$$\delta^1(\ell) = \min_{100 \leq k \leq \ell} \varepsilon^1(k)$$

as the best achieved distortion value for an encoding not longer than ℓ, for $\ell = 500$, 1000, ..., $10,000$. We can see that for all data sets an encoding length of $\ell = 1000...4000$ achieves a very close approximation of the theoretically best possible value, e.g., we have

$$\delta^1(4000) - \varepsilon^* \leq 0.006$$

for all seven data sets.

Finally, comparing the obtained results to the results of [4] in Table 3, we can conclude that the LP based model produces better embeddings into Hamming space than the semidefinite programming based model. We obtain uniformly better maximum rates using substantially shorter binary encodings.

Table 3: Comparison to earlier results: Maximum distortion rates

	Results from this paper			Results from [4]	
Data set	$\delta^1(500)$	$\delta^1(1000)$	$\delta^1(4000)$	Maximum distortion	Encoding length
BLOSUM 40	0.1037	0.0946	0.0802	0.1000	100,000
BLOSUM 45	0.0585	0.0274	0.0058	(not studied)	
BLOSUM 62	0.0461	0.0202	0.0056	0.0660	25,000
BLOSUM 80	0.0561	0.0158	0.0053	0.0390	100,000
PAM 40	0.0955	0.0712	0.0616	0.1610	100,000
PAM 120	0.2101	0.1779	0.1703	0.3010	100,000
PAM 250	0.2923	0.2672	0.2535	0.4260	100,000

5.3 Average distortion rates of obtained binary encodings

It is interesting to note that we get a quite different picture when looking at the average distortion rates. For each binary encoding we can define the average distortion rate as the average of the actual distortions of the $\binom{n}{2} = 190$ different distances.

As we can see e.g., from Figures 4 and 5 (or more from the web-site [6]), all three methods produce a faster converging average rate, than the corresponding maximum distortion rates. Similarly to maximum distortion rates, the first method seems to produce for all data sets the best average rates. However, as we can see from the last column of Table 1, the average distortion rates obtainable from the LP optima are not much better than the best maximum rates! Of course, these values are not a theoretical lower bound on the achievable average rates, but as we can see from Table 4, our methods do not produce much better results.

Denoting by $\widetilde{\delta}^1(\ell)$ the average distortion rate realized by the first method, we compare the achieved average rates to those from [4] in Table 4. As this comparison shows, the semidefinite programming based approximation technique in [4] seem to focus much better on minimizing the average distortion rate, a practically important parameter, than our LP based approach.

On a theoretical level, this is quite understandable. Our objective focuses exclusively on the maximum rate, and it is not influenced at all by the differences in average rate. Even if there are alternative optima with a much better average rate corresponding, our method may not discover that. It is also quite conceivable that by accepting solutions with somewhat worse maximum rate, we can decrease substantially the corresponding average rate (as the results of [4] seem to suggest).

Table 4: Comparison to earlier results: Average distortion rates

	Results from this paper	Results from [4]	
Data set	$\widetilde{\delta}^1(4000)$	Average distortion	Encoding length
BLOSUM 40	0.0640	0.0320	100,000
BLOSUM 45	0.0016	(not studied)	
BLOSUM 62	0.0013	0.0240	25,000
BLOSUM 80	0.0013	0.0190	100,000
PAM 40	0.0537	0.0870	100,000
PAM 120	0.1249	0.1230	100,000
PAM 250	0.1720	0.1310	100,000

To address this shortcoming, we propose as future research the following model, still based on linear programming:

$$\begin{gathered}
\min \quad \gamma\varepsilon_0 + (1-\gamma)\tfrac{2}{n(n-1)}\mathbf{e}^t(\varepsilon^+ + \varepsilon^-) \\
\text{subject to the constraints} \\
A\mathbf{x} + Diag(\mathbf{d})(\varepsilon^+ - \varepsilon^-) = \mathbf{d} \\
\varepsilon_0\mathbf{e} - (\varepsilon^+ + \varepsilon^-) \geq 0 \\
x \in \mathbb{R}_+^{\Omega}, \text{ and } \varepsilon^+, \varepsilon^- \in \mathbb{R}_+,
\end{gathered} \tag{10}$$

where $\varepsilon^+ = (\varepsilon_{ij}^+ \mid 1 \leq i < j \leq n)$ and $\varepsilon^- = (\varepsilon_{ij}^- \mid 1 \leq i < j \leq n)$ correspond to the distortions of the individual distances, ε_0 is the maximum distortion, $Diag(\mathbf{d})$ is the diagonal matrix, in which the main diagonal elements are the coefficients of the vector $\mathbf{d}$, and the off-diagonal elements are all equal to zero, and where $\mathbf{e} = (1, 1, ..., 1)$ is the full one vector of dimension $n(n-1)/2$.

In this model γ is a parameter. If we choose $\gamma = 1$, then we get a model fully equivalent with (7), i.e., which computes the theoretically best possible maximum distortion rate.

If we choose $\gamma = 0$, then we get a model in which the maximum distortion rate does not play any role, and the optimum value will be the theoretically best achievable average distortion rate.

Finally, if we choose $\gamma = 0.9999$, then we get a lexicographic model, which tries to minimize the maximum distortion rate, and among the optimal solutions for that objective, gives preference to those with a smaller average rate.

Acknowledgements

We would like to thank Professor Jonathan Eckstein for providing access to his well-equipped and fast computer, where we had access to both the CPLEX package and appropriate amount of RAM. We also would like to thank the REU program of DIMACS, Rutgers University for the support they provided.

References

[1] Matrices for Bioccelerator Searches: http://eta.embl-heidelberg.de:8000/misc/mat/

[2] J. Buhler, Provably sensitive Indexing strategies for biosequence similarity search, *Proceedings of the sixth annual international conference on Computational biology* (2002) 90-99.

[3] J. Buhler, Efficient large-scale sequence comparison by locality-sensitive hashing, *Bioinformatics* 17-5 (2001) 419-428.

[4] E. Halperin, J. Buhler, R. Karp, R. Krauthgamer and B. Westover, Detecting Protein Sequences Via Metric Embeddings, *Proceedings of the Eleventh International Conference on Intelligent Systems for Molecular Biology* (2003) 122-199.

[5] S. Henikoff and J. Henikoff, *Amino acid substitution matrices from protein blocks*, Howard Hughes Medical Institute, Fred Hutchinson Cancer Research Center, 1992.

[6] Project web-site: http://dimax.rutgers.edu/ leverett/index.html

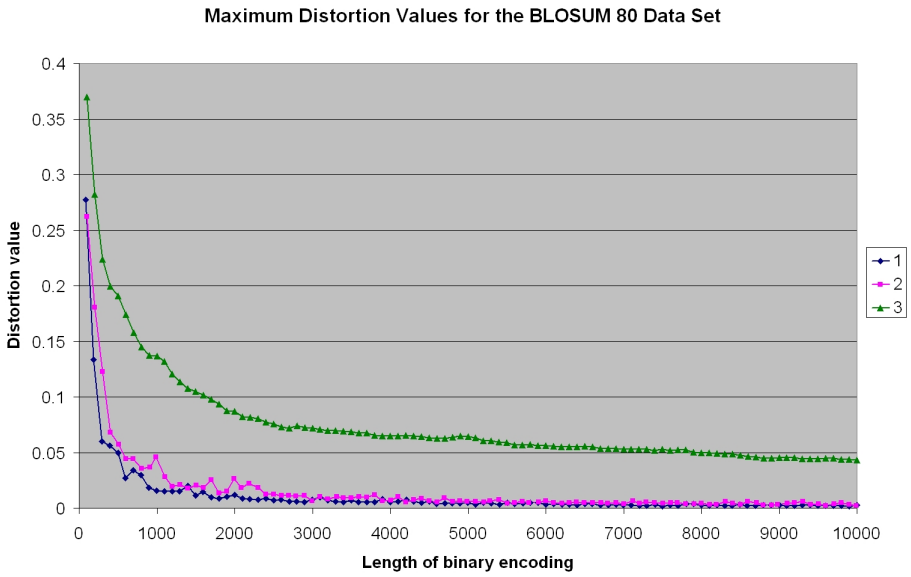

Figure 2: Maximum distortion values ε^1, ε^2 and ε^3, as functions of encoding length ℓ, obtained from the LP optimum for the BLOSUM 80 data set.

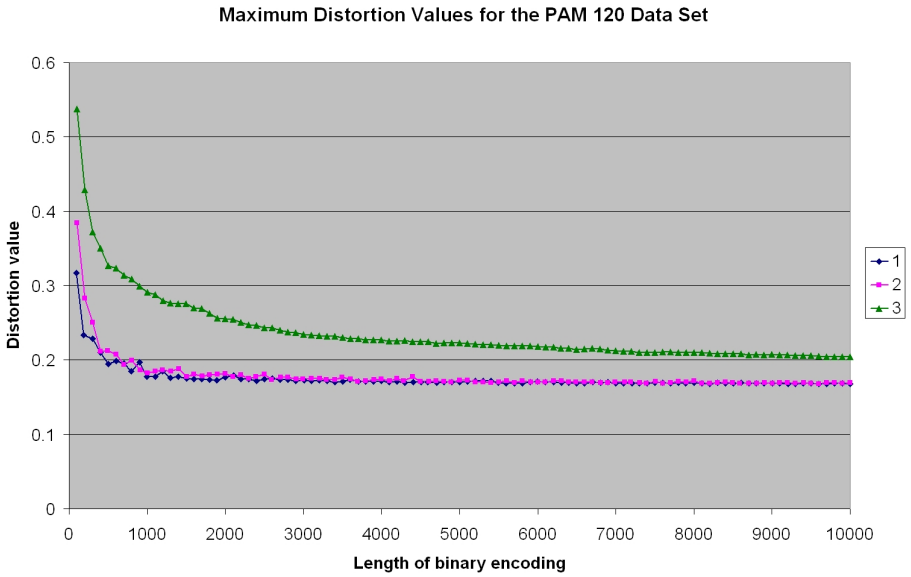

Figure 3: Maximum distortion values ε^1, ε^2 and ε^3, as functions of encoding length ℓ, obtained from the LP optimum for the PAM 120 data set.

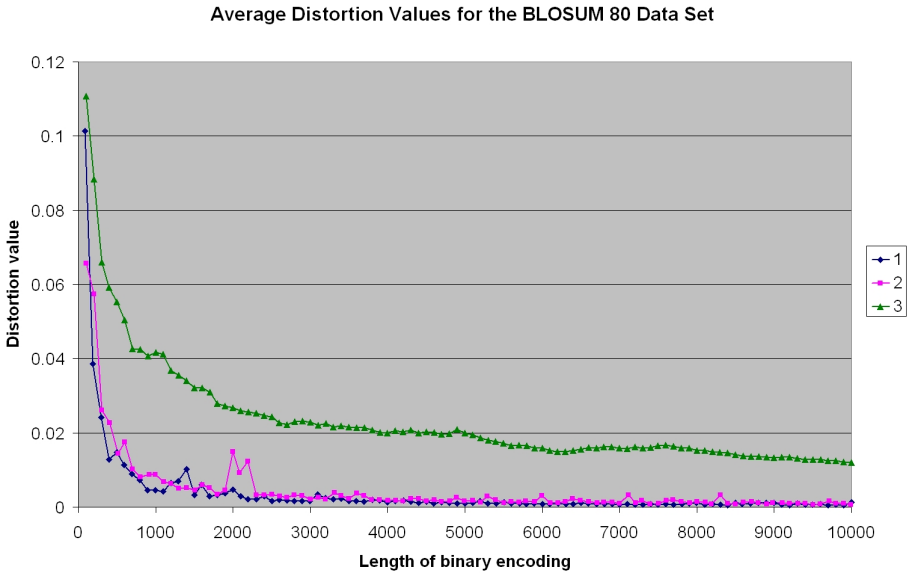

Figure 4: Average distortion values $\widetilde{\varepsilon}^1$, $\widetilde{\varepsilon}^2$ and $\widetilde{\varepsilon}^3$, as functions of encoding length ℓ, obtained from the LP optimum for the BLOSUM 80 data set.

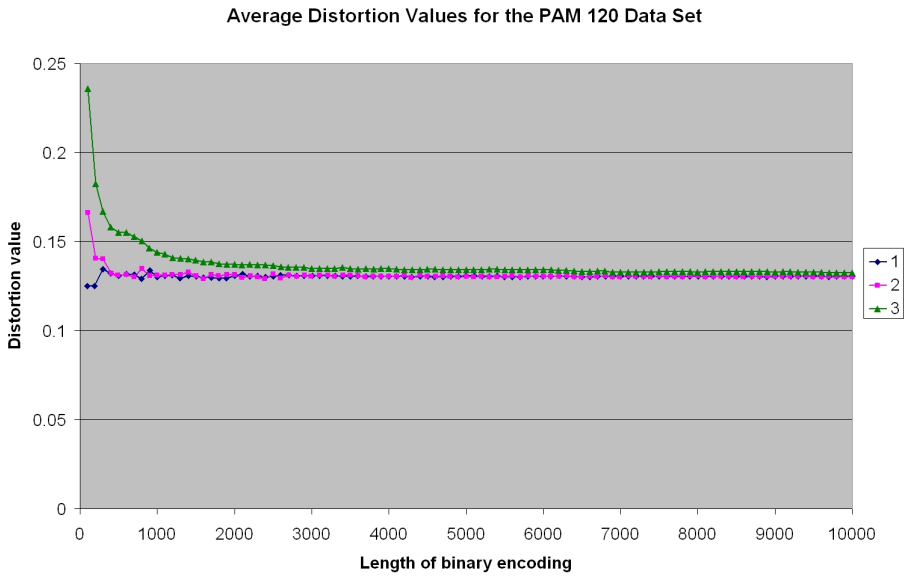

Figure 5: Average distortion values $\widetilde{\varepsilon}^1$, $\widetilde{\varepsilon}^2$ and $\widetilde{\varepsilon}^3$, as functions of encoding length ℓ, obtained from the LP optimum for the PAM 120 data set.