Approximate Similarity Search in Genomic Sequence Databases using Landmark-Guided Embedding

Ahmet Sacan [†,‡]
[†]Dept. of Computer Science and Engineering
The Ohio State University
Columbus, OH, USA
sacan@cse.ohio-state.edu

I. Hakki Toroslu [‡]
[‡]Computer Engineering Dept.
Middle East Technical University
Ankara, Turkey
toroslu@ceng.metu.edu.tr

Abstract

Similarity search in sequence databases is of paramount importance in bioinformatics research. As the size of the genomic databases increases, similarity search of proteins in these databases becomes a bottle-neck in large-scale studies, calling for more efficient methods of content-based retrieval. In this study, we present a metric-preserving, landmark-guided embedding approach to represent sequences in the vector domain in order to allow efficient indexing and similarity search. We analyze various properties of the embedding and show that the approximation achieved by the embedded representation is sufficient to achieve biologically relevant results. The approximate representation is shown to provide several orders of magnitude speed-up in similarity search compared to the exact representation, while maintaining comparable search accuracy.

1 Introduction

With the advent of high-throughput sequencing methods, the genomic sequences have been accumulating at an ever increasing rate. GenBank, a central database of publicly available DNA sequences, has been doubling in size every 15 months [3]. Since the sequencing of Hemophilus influenza in 1995, close to 700 organisms have been completely sequenced and published, and there are currently more than 3000 ongoing genome sequencing projects [14].

Homology search over these genomic sequence databases is a crucial step in the inference of functional and evolutionary relationships among proteins. According to a survey, similarity search makes up 35% of the tasks in bioinformatics research [10]. The increase in database size and the demands of large-scale analysis have been the driving forces of several efforts to speed up the similarity search process. The most successful of these efforts have been based on fast retrieval and *stitching* of common short subsequences. The goal of this study is to develop more effective common subsequence retrieval methods without significant compromise to the sensitivity of the similarity search results.

BLAST [1], which is currently the popular tool for biological homology search, is based on a heuristic that assumes presence of short exact matches between homologous sequences. For a given subsequence length k, a hash table of all possible k-mers is used to map the subsequences in the database. For a new query sequence, all k-mers of the query are searched using the hash table for finding matching k-mers in the database. The k-mer hits of the database are then tested for extension to generate longer matching regions and obtain the final local alignments [21].

There have been several improvements over BLAST that achieve more efficient or sensitive identification of evolutionarily close k-mers. These improvements were obtained mainly by relaxing the "short exact matches" assumption of BLAST's heuristic approach. Pattern Hunter [15] uses non-consecutive residues to construct k-mers, detecting replacements in the sequence better. The Piers method [6] guides inexact matching of short query segments using randomly selected seeds, achieving faster response at the cost of a small degradation in sensitivity.

In contrast to BLAST-like methods that are based on hashing short sequences, there have been *indexed search* approaches to the sequence retrieval problem. Note that the sequences are not objects in a multi-dimensional Euclidean space, which makes the spatial access methods (SAMs) such as R-tree and its variants [13, 20, 2] inapplicable. This has prompted the application of metric indexing methods, which do not need the original objects to be represented in multi-dimensional space, but only require that the distance measure between objects be metric (i.e., satisfy symmetry, non-negativity, and triangle inequality properties). In metric indexing, the relative distances of the sequences are used to organize and partition the data into a hierarchical structure based on the distances to representative sequences of the partitions at each level. The triangle inequality is then used to prune the search space during the traversal of the metric-tree while answering a similarity search query. A survey of the metric indexing based methods can be found in [22].

Due to the requirements on the distance measure, the metric indexing methods have only considered the basic edit distance measure, where an identity matrix is used as the residue substitution matrix [5]. The identity matrix may be appropriate for nucleotide sequences where the substitutability of the nucleotides is almost uniform. However, the identity matrix does not give biologically accurate results for protein sequences, where the similarities and differences among individual residues become biologically more significant. [17] considers modelling other substitution matrices as near-metric based on the maximum and minimum substitution values whereas [26] have uses mPAM [27], a biologically more sensitive metric substitution matrix.

[26] uses a multiple vantage point (MVP) tree to index subsequences. MVP tree, like other vantage point trees, is built by means of a top-down recursive process and does not gracefully support insertions and deletions. M-tree of [7] maintains a height-balanced tree to overcome this problem. Despite having good performance in general metric indexing applications, M-trees still suffer having a large number of sequence comparisons in biological sequence search (see Venkateswaran et al. 25 for a comparison).

Reference-based indexing have also been applied to similarity search in biological databases. In [25], a variable number of reference sequences are assigned to each database sequence, and the distances to the reference points are used to avoid unnecessary distance calculations between query and database sequences. Even though the number of distance calculations is minimized in reference-based indexing, the search is performed sequentially (i.e., every single sequence in the database is tested), which does not scale well for larger database sizes.

One further problem of the metric search methods, as they have been employed so far, is that only the global similarity between sequences is considered (with [26] being an exception). In the biological domain, the global similarity has limited applicability, and requires that the sequences being compared be evolutionarily very close. The end-gaps, which are normally not penalized in the biological domain, are also not handled gracefully by these methods. We note that the global similarity measure may find use in searching very similar proteins in whole-genome comparisons; however, it is far from being applicable to the general homology search problem, which ultimately relies on detection of locally conserved short subsequences.

In this study, we limit our focus to the k-mer search, which is the main step in biologically relevant local search of homologous sequences. We propose an approximate similarity search which is based on landmark-guided embedding of the k-mers. We map the k-mers of a sequence database to a vector space based on their distances to a reference set of k-mers (denoted as *landmarks*). The k-mers in the embedded space are then indexed using spatial access methods for fast similarity search.

The contributions of this study include: (1) hybridizing Fastmap [9] and LMDS [8] methods to achive more robust and accurate sequence embedding, (2) providing an approximate vector representation of sequences, (3) showing that the embedded representation allows efficient and biologically relevant indexing and similarity search.

2 Methods

Throughout this presentation, we use q to denote an input query sequence of length m whose symbols are from an alphabet of size σ. The set of database sequences are denoted as $S = s_1, s_2, ..., s_N$ where N is the number of sequences in the database. We generate all k-mers from both database and query sequences using a sliding window over sequences with a step size of one symbol.

The edit distance between two sequences is defined as the minimum cost of edit operations (insert, delete, replace) that transform one sequence to the other. The cost of replacing an individual symbol to another is looked up from a substitution matrix M. The costs of insertions and deletions are generally provided as an optimized gap penalty parameter. Without loss of generality, we used the weighted Hamming distance instead of the general alignment distance between sequences in order to decrease the analysis time. Because the gap penalty is usually larger than mismatch scores, weighted Hamming distance is sufficient when comparing short k-mers (see [26] for a proof correctness).

Our goal is to represent the set of k-mers in a low-dimensional space while preserving the distances among them as much as possible. Note that the k-mers cannot be directly represented as points in multi-dimensional vector space, therefore the classical dimension reduction techniques that rely on presence of the original high-dimensional vector space are not applicable here. Moreover, in the context of similarity search, the mapping has to be easily extensible to new query objects without requiring re-embedding of the whole database. These requirements lead us to the landmark based methods that generate a metric-preserving embedding using distances to only a small selection of sequences.

The FastMap method by [9] uses an iterative embedding procedure where at each iteration, the data is embedded onto an axis formed by two data points and the projection of

Symbol	Definition
σ	length of the alphabet
k	length of each k-mer subsequence
q	query sequence for which a similarity search is being performed
M	the substitution matrix that gives the costs of replacing symbols
d	the dimensionality of the space in which the k-mers are embedded
N	number of k-mers to be embedded
n	number of landmark points (sequences)
$D_{A,B}$	the distances of sequences in set A to those in set B
Δ	squared distances
D'	Euclidean distance in the embedded space

Table 1. Symbols used in this presentation and their definitions.

the data onto this axis is used as input for the next iteration. The landmark points at each step are chosen heuristically to be as distant as possible in order to account for the highest variance in the data distribution. FastMap relies on the assumption that the original space is a Euclidean space and makes use of the 'cosine law' for projection and embedding. This assumption causes the embedding to be unstable if the original space is not Euclidean.

Recently, [8] have proposed a scalable landmark-guided metric preserving embedding algorithm, LMDS, that shows better stability properties than FastMap. LMDS first designates a set of n objects as landmark points and applies classical MDS on $n \times n$ matrix $D_{n,n}$ of distances between pairs of landmarks to obtain an embedding in d-dimensional space. The classical MDS [23] computes the d largest positive eigenvalues λ, of the mean-centered inner-product matrix with the corresponding orthonormal set of eigenvectors ν. The d-dimensional embedding vectors for the landmark points are given by the following matrix:

$$L_k = \begin{bmatrix} \sqrt{\lambda_1}\nu_1^T \\ \sqrt{\lambda_2}\nu_2^T \\ \dots \\ \sqrt{\lambda_d}\nu_d^T \end{bmatrix}$$

For the rest of the data points, *distance-based triangulation* is applied to each point x using $\Delta_{x,n}$ vector of squared distances to the n landmark points. The embedding vector $\overrightarrow{x}$ is obtained using the pseudoinverse transpose $L_k^\sharp$ of the landmark embeddings by the formula:

$$\overrightarrow{x} = -\frac{1}{2}L_k^\sharp(\Delta_{x,n} - \Delta_{x,n}^\mu)$$

where $\Delta_{x,n}^\mu$ is the mean value of squared distances to the landmark points.

The quality of the embedding depends partially on the selection of initial landmark points. We use three different methods for designating the landmark points. $LMDS_{rand}$ randomly selects the landmarks from the original data points. $LMDS_{minmax}$ obtains a set of landmarks that are distant from each other by starting from a random landmark and heuristically adding new landmarks such as to minimize the maximum distance to the already selected landmarks (This heuristic is similar to the one proposed by [11]). In order to combine the landmark selection performance of Fatmap and stability of LMDS, we also propose $LMDS_{fastmap}$ method, which uses the same landmark points as found by the Fastmap method on the same dataset.

Once all the database k-mers are embedded into the vector space, the indexing and retrieval tasks can be delegated to spatial access methods. In the experiments section, we present the search speed results of using X-tree [4], however any of the spatial access methods can be used for this purpose. A query k-mer would be embedded into the same vector domain using its distances to the landmarks used in generating the embedding. Using the spatial method of choice, the mapped query can then be searched against the vector representations of the database k-mers.

3 Experiments

The performance of the embeddings is evaluated on synthetic and real datasets. In synthetic datasets, for a given alphabet size σ and subsequence length k, all k-mers were generated. The size of the synthetic datasets were limited to 10,000 sequences, and a random sampling from all possible sequences were performed if the number of k-mers $N = k^\sigma$ exceeded 10,000. An identity substitution matrix is used to calculate the distances between k-mers.

The real data was obtained from the yeast proteins dataset which is used to benchmark BLAST (`ftp.nsbi.nlm.nih.gov/pub/impala/blasttest`). The yeast dataset contains 6,341 protein sequences with a total of about 2.9 million residues. The dataset also contains a separate query set of 103 proteins ranging from 38 to 884 residues in length, whose true positive hits are determined by human experts. Note that the alphabet of the protein sequences has cardinality of 20 and is composed of amino-acid residue symbols.

To accurately model biologically relevant distances among sequences, we used CB-EUC substitution matrix by [16], which is a metric matrix with good sequence alignment performance. Note that according to [19], if a substitution matrix is metric, then the alignment distances of the sequences using this matrix also forms a metric.

For each embedding method and variations, we evaluated the quality of the embedded sequences using Sammon's *metric stress* measure E [18] which quantifies the error in the preservation of the original distances, with a value 0 indicating a lossless embedding:

$$E = \frac{1}{\sum_{i<j}^{n} D_{ij}} \sum_{i<j}^{n} \frac{(D_{ij} - D'_{ij})^2}{D_{ij}}$$

where D_{ij} is the original distance between kmers i and j, and D'_{ij} is the distance in the embedded space.

3.1 The dimensionality of the embedded space

There is an accuracy-performance trade-off on the number of dimensions to be used in the embedded space. As the number of embedding dimensions d is increased, the original data can be represented better, at a higher cost incurred on the similarity search in the embedded space. Since a lossless embedding is not possible, one needs to empirically determine the dimensionality for a desired level of mapping accuracy.

Figure 1 shows the metric-stress (left) and the correlation coefficient (right) of the mapped distances with respect to the original distances. The original data is a synthetic set of sequences of length 5, where CB-EUC substitution matrix (alphabet size $\sigma = 20$) is used to calculate the original distances. (Qualitatively similar results were obtained for other k and σ values.) In order to obtain a fair comparison, the same number of landmark points required in Fastmap ($n = 2 \times\ d$) is used in the LMDS methods. The $LMDS_{maxmin}$ and $LMDS_{fastmap}$ methods show similar mapping accuracies, whereas $LMDS_{random}$ requires more dimensions to achieve the same level of accuracy. The Fastmap method performs similar to LMDS methods up to a certain number of dimensions, after which the numerical instability in the mapping accumulates and degrades the mapping accuracy.

Due to its numerical instability, Fastmap does not necessarily give a better embedding as the number of dimensions is increased. Despite this instability, we have observed an important merit of the Fastmap method. Namely, the number of dimensions beyond which Fastmap's accuracy degrades corresponds to the intrinsic Euclidean dimensionality of the original dataset. Notice that in Figure 1, the metric stress achieved by Fastmap at $d = 7$ is comparable to the metric stress achievable by the LMDS methods with higher dimensions. This observation has led us to determine the dimensionality of the target embedding space as this *breaking point* in Fastmap's mapping accuracy. This gives us a more well-defined definition than assessing the convergence of LMDS.

Defining the *breaking point dimensionality* of a given dataset has allowed us to analyze its dependence on the other parameters. As shown in Figure 2 (left), the *breaking point dimensionality* of the dataset increases linearly with both the sequence length k and the alphabet size σ. An identity substitution matrix is used in the distance calculation in order to compare the alphabet size σ across datasets. Note that even though the alphabet size is 20 for proteins, amino-acid substitution matrices impose a clustering on the amino-acid types, which in effect reduces the intrinsic dimensionality of the dataset. The dimensionality of the dataset when the CB-EUC substitution matrix is used ranges between that of the datasets with $\sigma = 4$ and $\sigma = 5$. In fact, the principal component analysis of the CB-EUC matrix shows that the first 5 principal components account for the 98.2% of the variation in the matrix values.

3.2 Number of landmarks

In Fastmap, 2 landmarks are chosen for each dimension to provide an axis of projection for the data, which yields the total number of landmarks to be $n = 2 * d$. Whereas in the LMDS methods, the number of landmarks can be chosen arbitrarily, provided that $n \geq d + 1$. The effect of the number of landmarks on the mapping accuracy is shown in Figure 2 (right). The best landmark selection strategy is that of Fastmap, in terms of monotonically decreasing the metric stress. For $LMDS_{maxmin}$, each new landmark is not guaranteed to improve the mapping accuracy, because the landmark selection heuristic is only an approximation to the optimal selection. However, the LMDS methods do converge to an optimal mapping accuracy, if sufficiently large number of landmark points are used.

The number of landmarks affects the computational complexity of mapping the database to the embedded space, and also of mapping new query sequences for similarity search in the embedded space. Even though LMDS methods can provide further improvement in the mapping accuracy, the number of landmarks would be limited by the amount of time one is willing to spend in mapping new sequences.

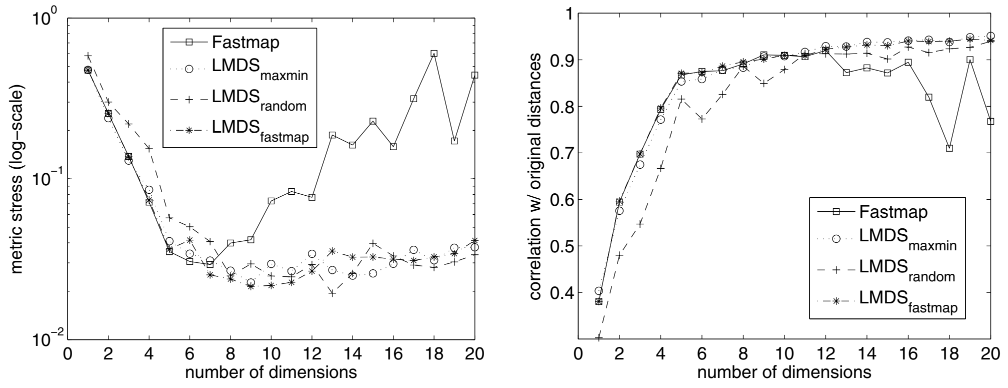


Figure 1. Mapping accuracy of the embedding vs. target dimensionality (k=5).

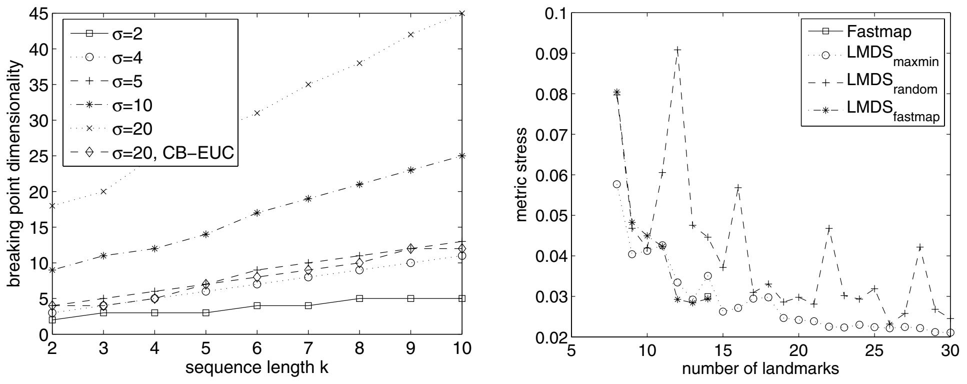


Figure 2. Left: Dependency of dimensionality on sequence length and alphabet size. Right: The effect of the number of landmarks on mapping accuracy (k=5, d=7).

3.3 Similarity search performance

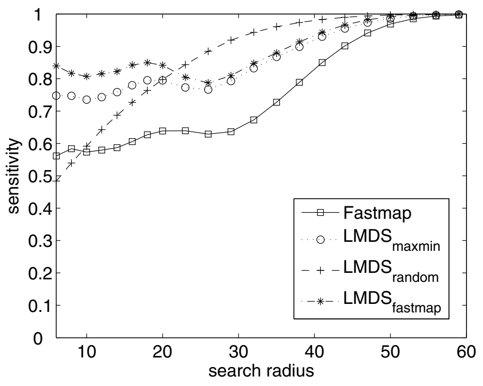
While metric-stress is a good indication of how well the distances among the original sequences are preserved in the embedded space, the similarity search accuracy within the embedded space still remains to be evaluated. In order to test the similarity search performance, we performed range queries on the yeast dataset using the separate set of query proteins and various distance thresholds. For a given query kmer q and distance threshold r, a *range query* in the original sequence space would return all the kmers in the database that are within r edit distance of the query kmer. Similarly, in the embedded vector space, all mapped objects that are within r Euclidean distance away from the image of the query q' are returned.

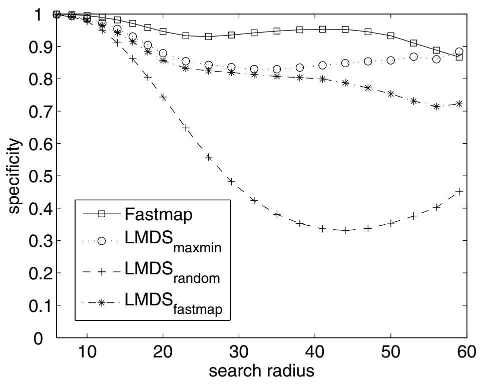
Figure 3 shows the sensitivity (true positive rate) and specificity (1 - false positive rate) of the range queries for different mapping methods under various search radii r. The results are the averages of the queries performed for all kmers in the test query set. The approximation by Fastmap tends to overestimate the original edit distances, which yields less number of hits in the answer set, and thus higher specifity but lower sensitivity compared to other methods. $LMDS_{fastmap}$ combines the landmark selection algorithm of Fastmap with the stability of LMDS to yield the best sensitivity results while having comparable specificity with those of $LMDS_{maxmin}$ and Fastmap.

3.4 Homology search performance

It must be noted that in the context of homology search, a small distance threshold is sufficient to obtain biologically relevant range queries, because the homologous proteins are expected to share very similar subsequences. Moreover, the homology search procedure is particularly permissive to small errors in the approximation of kmer distances, because kmers from the homologous proteins missed by some of the query subsequences are compensated by other subsequences that correctly return the kmers of the homologous proteins.

Figure 3. Sensitivity (left) and specificity (right) of kmer range search results. (k=6, d=8)





For each of the 103 yeast query proteins, we generated all kmers and searched the yeast dataset for kmer hits. For each distance threshold, a search result is considered to be a true hit if at least one kmer of the query protein returns a kmer of the homologous proteins. Figure 4 shows the homology search results using varying distance thresholds. For comparison, the results of an exact kmer search in the sequence space are also included. Notice that the results of the homology search are in accordance with the results of the kmer search in Figure 3. Namely, the methods that provide more sensitive kmer answer sets also provide more sensitive homology search results.

Even though for a given distance threshold, more kmers are returned per kmer search in the embedding methods compared to an exact kmer search (Figure 4, right), the embedding methods require a smaller search radius to achieve the same level of sensitivity (Figure 4, left). For instance, to achieve 90% sensitivity (i.e., to obtain 90% of the homologous proteins), the $LMDS_{maxmin}$ method returns $3.5\%_{000}$ of the kmers per kmer search, whereas an exact kmer search in the sequence space returns $4.1\%_{000}$ of the kmers. This is due to the fact that in the embedding methods, the approximation errors at lower distance thresholds cause the distances to some of the kmers of homologous proteins to be underestimated. These kmers are then returned in the answer set, whereas they would not be present in a range query in the original sequence space.

Note that smaller kmer lengths k while require a smaller search radius and provide more sensitive homology search, they incur higher false positive rates; whereas higher k values provide more specific results at the cost of sensitivity. $k = 6$ was found to be a good trade-off between sensitivity-specificity of homology search results on the yeast dataset. The relative performance of the methods were similar for other kmer lengths.

3.5 Search time performance

Short subsequences are embedded under the premise that indexing and similarity search in a vector domain is more efficient than those in the sequence domain. In order to illustrate this, we performed indexing in both domains and compared the CPU times for range queries. We employed Slim-tree [24] *metric access method* (MAM) for indexing the sequences, and X-tree [4] *spatial access method* (SAM) for indexing the vector representations resulting from the $LMDS_{fastmap}$ method.

Figure 5 shows the average query times for varying database sizes and search radii. Similarity search in the vector domain achieves approximately 500-fold speed-up over that in the original sequence domain. A search radius of 7 is used while varying the database sizes (left) and a database size of 100,000 is used while varying the search radii (right). A similar trend in search times were observed for other k, database size, and search radius values. While an exhaustive analysis and comparison of MAM and SAM methods are beyond the scope of this study, we note that the search speeds achieved by Slim-tree and X-tree are representative of those achievable by the currently available MAM and SAM methods.

4 Discussion

In this study, we have proposed an approximation to similarity search in sequence databases by embedding the sequences in a vector space based on their distances to selected landmark sequences. We have demonstrated that similarity search in the embedded space can be performed several orders of magnitude faster than that in the original sequence space, without significant loss in the accuracy of the search results. Fastmap and LMDS methods with various landmark selection heuristics are investigated for their embedding and similarity search accuracy.

Figure 4. Homology search performance on the yeast dataset (k=6, d=8)

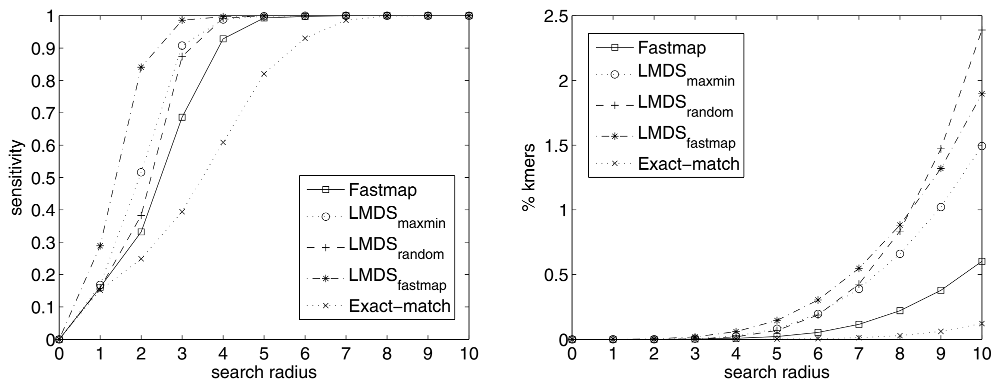
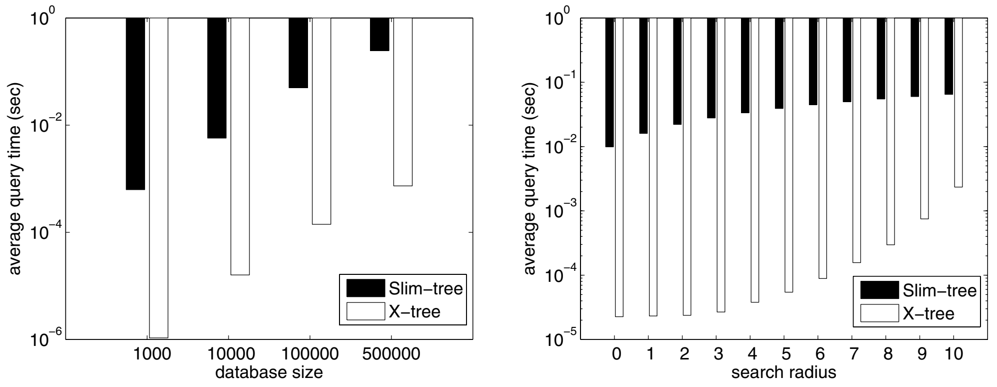


Figure 5. Average query time comparison (k=6, d=8)



While the Fastmap provides a good heuristic for landmark selection, its numerical instability causes degradation in the mapping accuracy. Moreover, Fastmap tends to overestimate the original distances compared to LMDS methods, causing a lower sensitivity in similarity search results. We have proposed $LMDS_{fastmap}$ method which uses the landmarks generated by the Fastmap, yet provides stability in the mapping, yielding better performance in mapping and similarity search. The mapping accuracy achieved by the embedding methods can be further improved using higher dimensionality in the embedding space, or using a larger number of landmark sequences. We have presented a systematic comparison of the performance on synthetic and real sequence datasets.

In this study, we have mainly focused on kmer search, which constitutes a significant initial step to the general homology search problem. These short subsequences can then be extended and stitched to obtain the final sequence alignments. We note that the efficiency of the embedding and the indexing will further depend on the subsequence extension algorithm used. We refer the reader to [12] for details of these algorithms.

We expect the vector representation of sequences to have applications beyond similarity search. For instance, a vector domain simplifies the representation of a group of sequences by their mean vector, which otherwise is not readily available in the sequence domain. We are currently investigating the use of such abstract representations in sequence clustering and multiple sequence alignment applications.

The landmark embedding methods presented here can also be applied to content-based retrieval in other domains such as image and multimedia databases. In such applications, the original space is a high-dimensional space formed by various features extracted from the database objects. The landmark-guided embedding would provide a dimensionality reduction and allow efficient similarity search. Furthermore, the similarity search in these applications are especially tolerant of the approximation errors incurred by the embedding, because the original features and the classification of objects are in general subjectively determined.

5 Acknowledgements

This research was supported by Turkish Scientific and Research Council (TUBITAK) Grant 107E173. We would also like thank the anonymous reviewers for their critical analysis of and corrections to the manuscript.

References

[1] Altschul, S., Gish, W., Miller, W., Myers, E., and Lipman, D. (1990). Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410.

[2] Beckmann, N., Kriegel, H.-P., Schneider, R., and Seeger, B. (1990). The R*-tree: an efficient and robust accessmethod for points and rectangles. *ACM SIGMOD*, pages 322–331.

[3] Benson, D. A., Karsch-Mizrachi, I., Lipman, D. J., Ostell, J., and Wheeler, D. L. (2004). Genbank: update. *Nucleic Acids Res.*, 32, Jan 1:D23–6.

[4] Berchtold, S., Keim, D. A., and Kriegel, H.-P. (1996). The X-tree: An index structure for high-dimensional data. In *VLDB*, pages 28–39.

[5] Bozkaya, T. and Ozsoyoglu, M. (1999). Indexing large metric spaces for similarity search queries. *ACM Transactions on Database System*, 24(3):361–404.

[6] Cao, X., Li, S. C., Ooi, B. C., and Tung, A. K. H. (June 2004). Piers: An efficient model for similarity search in dna sequence databases. *SIGMOD Record*, 33(2).

[7] Ciaccia, P., Patella, M., and Zezula, P. (1997). M-tree: An efficient access method for similarity search in metric spaces. *Proc. Intl. Conf. on Very Large Databases (VLDB)*, pages 426–435.

[8] de Silva, V. and Tenenbaum, J. B. (2003). Global versus local methods in nonlinear dimensionality reduction. *Proc. NIPS*, 15:721–728.

[9] Faloutsos, C. and Lin, K.-I. (1995). FastMap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. In *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, pages 163–174.

[10] Goble, R. S. C., Baker, P., and Brass, A. (2001). A classification of tasks in bioinformatics. *Bioinformatics*, 17:180188.

[11] Gonzalez, T. (1985). Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306.

[12] Gusfield, D. (1997). *Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology*. Press Syndicate of the University of Cambridge, USA.

[13] Guttman, A. (1984). R-trees: A dynamic index structure for spatial searching. *ACM SIGMOD*, pages 419–429.

[14] Liolios, K., Mavrommatis, K., Tavernarakis, N., and Kyrpides, N. (2008). The Genomes On Line Database (GOLD) in 2007: status of genomic and metagenomic projects and their associated metadata. *NAR Database issue*. in press.

[15] Ma, B., Tromp, J., and Li, M. (2002). Patternhunter: faster and more sensitive homology search. *Bioinformatics*, 18:440–445.

[16] Sacan, A. and Toroslu, I. H. (2007). Amino acid substitution matrices based on 4-body delaunay contact profiles. *IEEE 7th Intl Symp on Bioinformatics and Bioengineering (IEEE-BIBE2007)*, pages 796–802.

[17] Sahinalp, S., Tasan, M., Macker, J., and Ozsoyoglu, Z. (2003). Distance based indexing for string proximity search. *IEEE Data Engineering Conference*, pages 125–137.

[18] Sammon, J. (1969). A nonlinear mapping for data structure analysis. *IEEE Transactions on Computers*, C-18:401–409.

[19] Sellers, P. (1974). On the theory and computation of evolutionary distances. *J. Appl. Math. (SIAM)*, 26:787–793.

[20] Sellis, T., Roussopoulos, N., and Faloutsos, C. (1987). The R$^+$-tree: A dynamic index for multimensional objects. *13th VLDB*, pages 507–518.

[21] Smith, T. and Waterman, M. (1981). Identification of common molecular subsequences. *Journal of Molecular Biology*, 147:195–197.

[22] Taskin, M. and Ozsoyoglu, Z. M. (2004). Improvements in distance-based indexing. *Proceedings of the 16th International Conference on Scientific and Statistical Database Management, SSDBM04*, pages 161–170.

[23] Torgerson, W. S. (1958). *Theory and Methods of Scaling*. New York: Wiley.

[24] Traina, J. C., Traina, A. J. M., Seeger, B., and Faloutsos, C. (2000). Slim-trees: High performance metric trees minimizing overlap between nodes. In *EDBT 00: Proceedings of the 7th International Conference on Extending Database Technology*, pages 51–65.

[25] Venkateswaran, J., Lachwani, D., Kahveci, T., and Jermaine, C. (2006). Reference-based indexing of sequence databases. *VLDB 2006*.

[26] Xu, W., Mao, R., Wang, S., and Miranker, D. P. (2006). On integrating peptide sequence analysis and relational distance-based indexing. In *BIBE '06: Proceedings of the Sixth IEEE Symposium on BionInformatics and BioEngineering (BIBE'06)*, pages 27–34.

[27] Xu, W. and Miranker, D. P. (2004). A metric model of amino acid substitution. *Bioinformatics*, pages 1214–1221.