

Genome on Demand: Interactive Substring Searching *

Tamer Kahveci

Ambuj K. Singh

Department of Computer Science
University of California, Santa Barbara, CA 93106
{tamer,ambuj}@cs.ucsb.edu

1 Motivation and methods

The explosive growth of genome databases makes similarity search a challenging problem. Current search tools are non-interactive in the sense that the user has to wait a long time until the entire database is inspected.

We consider the problem of interactive string searching, and propose two k -NN (k -Nearest Neighbor) search algorithms. For a given query, our techniques start reporting the initial results quickly. Later, these results are periodically refined depending on the user satisfaction. We briefly discuss these techniques in the following sections. Detailed description of these methods and further experimental results are available in [3].

1.1 LIS: Local statistics-based interactivity

Our first technique, called *LIS* (*Local statistics-based Interactive Search*), approaches this problem by transforming the information from string space into vector space. This is done by sliding a window of a predefined length, w , on all the strings in the database. Each positioning of this window corresponds to a substring. The *frequency vector* (count of each letter in a string) [2] is then computed for each such substring. Each frequency vector maps to a point in a multi-dimensional integer space. These points are then clustered into MBRs (Minimum Bounding Rectangles) using an MRS index structure [2]. Since, the length of all substrings considered are equal, all frequency vectors lie on the same multi-dimensional plane, called the *data plane*. This process is repeated for various values of w .

We identify a representative frequency vector for each MBR as the point that is closest to all of the points in that MBR. This point is computed as the projection of the centroid of an MBR to the data plane.

For a given k -NN query, q , LIS starts by constructing an MBR that covers the frequency vectors of all possible query substrings of length w , for all resolutions in the MRS index structure that is less than $|q|$. Later, the representative vectors of these MBRs are determined. The k^{th} order statistics for each MBR in the MRS index structure is then computed

based on the *GED* (*Gapless Edit Distance*) distribution between that MBR and the query MBR. The computation of the order statistics is explained in detail in [3].

Our k -NN algorithm hierarchically finds the next MBR of the index structure in ascending order of the mean of their k^{th} order statistics, and inspects them iteratively using a highly optimized search tool, such as BLAST [1]. Intermediate results are reported to the user along with confidence level. Confidence level exhibits *how confident the user should be with the current partial results*.

Confidence intervals are computed based on the results discovered so far and the *GED* distributions of the uninspected MBRs. It shows the probability that the k^{th} closest distance in the remaining MBRs is not less than the k^{th} closest distance found so far.

We reduce the total search time of this technique by pruning the MBRs on the fly that do not contain results better than the ones found so far. We name this version of LIS as *LIS-prune*.

1.2 GIS: Global statistics-based interactivity

Our second technique, called *GIS* (*Global statistics-based Interactive Search*), uses the available statistical theory for string databases [4] to estimate the score of the maximal alignment. BLAST uses this theory to estimate the quality of a result once an alignment is determined. We use the same model to predict which database blocks contain good results prior to the actual alignment.

Let $\Sigma = \{\alpha_1, \alpha_2, \dots, \alpha_\sigma\}$. Assume that the letters are sampled with probabilities $\{p_1, p_2, \dots, p_\sigma\}$ in the data string, and $\{q_1, q_2, \dots, q_\sigma\}$ in the query string. Let the score of a match of α_i and α_j be $s(\alpha_i, \alpha_j)$. Define $f(\lambda) = \sum_{i,j} p_i p_j e^{\lambda s(\alpha_i, \alpha_j)}$. BLAST uses the unique positive solution, λ^* , to the equation $f(\lambda) = 1$ in the statistical computation. Karlin and Altschul [4] show that the expected value for the score of the maximal alignment of two strings can be approximated as $\ln(mn)/\lambda^*$, where m and n are the lengths of the two compared strings.

Similar to LIS, GIS partitions the database strings into overlapping blocks. Unlike LIS, we store the frequency of all the letters for each partition separately. Given a k -NN

* Work supported partially by NSF under grants BDI-0213903 and EIA-0080134

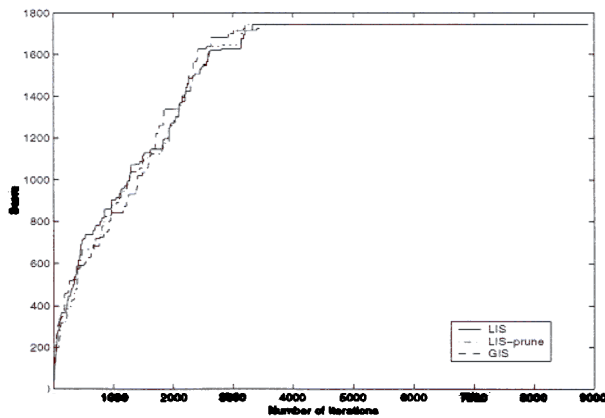


Figure 1. The score found by LIS, LIS-prune, and GIS at different iterations for query set generated from *Homo sapiens* chromosome 18 with 5% mutation on *Homo sapiens* chromosome 18 and *E.coli* (K12 MG1655) dataset.

query, q , GIS starts by computing the frequency vector for q . The score of the maximal alignment for each partition s is approximated by $\ln(|s| \cdot |q|) / \lambda^*$. Since the length of substrings in all the partitions are fixed, $\ln(|s| \cdot |q|) / \lambda^*$ becomes larger when λ^* is smaller. We compute λ^* for each partition using the Newton-Raphson method. Later, we search the partitions in ascending λ^* order using BLAST and report intermediate results to the user for each partition.

2 Experimental results

We downloaded the source code of BLAST, and implemented the MRS index structure. We used BLAST for alignments of DNA strings and the standard Smith Waterman algorithm for alignments of protein strings. We implemented LIS, LIS-prune and GIS. We performed k -NN queries for various values of k on a 1.4 GHz AMD Athlon MP+ computer with 1 GB memory.

Figure 1 shows the average score found by LIS, LIS-prune, and GIS for 1-NN queries at different iterations for queries modified with 5% mutation rate of $|q| = 4000$ query set on *Homo sapiens* chromosome 18 and *E.coli* (K12 MG1655) datasets. Since these two datasets have approximately same number of base pairs, they have similar number of MBRs. As evident from these figures, all of our techniques find the optimal results before half of the database is inspected. This means that, our techniques can distinguish the distant regions in *E.Coli* from closer regions in *Homo sapiens* chromosome 18.

Figure 2 plots the average score found at various iterations for the query set $|q| = 256$ with 10% and 20% modification on SWISSPROT database. The experiments show

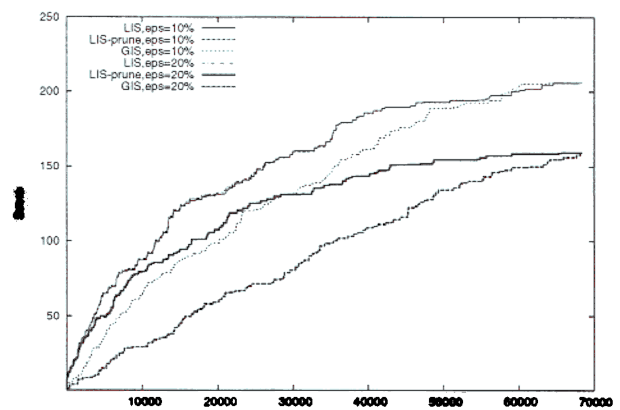


Figure 2. The score found by LIS, LIS-prune, and GIS at different iterations on SWISSPROT dataset for $|q| = 256$, and mutation rates 10% and 20%.

that both LIS and LIS-prune are better than GIS. LIS-prune is slightly better than LIS since it prunes some of the data blocks in advance. For 20% mutation rate, LIS and LIS-prune achieved 75% accuracy after only 31% of the data is processed. On the other hand GIS processes 66% of the dataset to achieve the same quality. This means that BLAST's statistical model fails to predict the high scoring regions for larger alphabets while our model still works well.

Our experimental results show that the proposed techniques achieve high accuracies quickly. In our experiments on DNA strings, GIS found high scoring results slightly faster than LIS. Both techniques achieved 75% accuracy within the first 2.5-35% of iterations. For protein strings, LIS and LIS-prune were much better than GIS.

References

- [1] S. Altschul and W. Gish. Basic local alignment search tool. *J. Molecular Biology*, 1990.
- [2] T. Kahveci and A. Singh. An efficient index structure for string databases. In *VLDB*, pages 351–360, Roma, Italy, September 2001.
- [3] T. Kahveci and A. K. Singh. An interactive search technique for string databases. Technical Report 10, UCSB, 2003.
- [4] S. Karlin and S.F. Altschul. Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes. *Proc. Natl. Acad. Sci.*, 87:2264–2268, March 1990.