Evaluation of anchoring schemes for fast DNA sequence alignments

Stéphane Guyetant, Dominique Lavenier
{sguyetan, lavenier}@irisa.fr
IRISA, Université de Rennes 1,
Campus de Beaulieu,
35042 RENNES Cedex - France

Abstract

To be performed within a reasonnable amount of time, a whole genome search needs a heuristic step, like the anchor generation of BLAST algorithm. We present the possibilities of a translation of such an heuristic on a hardware filter and show the methods to evaluate and quantify the quality of anchoring step and the speed-up achieved.

1 Introduction

Searching a nucleotide sequence against a whole genome, or even a whole database is routinely performed to identify the possible functionnality of a sequence. Speeding up the process with heuristics shall not decrease the sensibility while we may loose weak similarities: a too restrictive anchor generation will give less hits, decreasing the overall execution time but will miss significant alignments. On the other hand, refining the anchoring search is tricky as the whole database is processed throught this step. The balance between spending more time finding less but more valuable anchors and having a fast anchoring step can be solved by the use of dedicated hardware. A prototype harware filter currently under development in our team can do both by introducing parallelism in anchor finding.

The principle of anchoring: The idea is to quickly find small words, called seeds, or hits, or anchors that betray a possible alignment. Any alignment may contain this short and ungapped word of strong similarity. An exhaustive scoring method is then applied to extend these shorts alignments or to join several of them. If we represent an ungapped alignment with a "1" for a match and a "0" for a mismatch, then we are searching a short region with a high density of ones.

Widely accepted algorithms: BLAST [1] is by far the most used software to perform alignments. Regardless of its actual implementation, the heuristic seeks exact matches with a default size of 11 nucleotides along every alignment of the query and the database. It suffers from a lack of sensitivity, and its false negatives can be quite high ranked when an alignment score is computed with dynamic programming.

A new seeding scheme has been proposed in the software PATTERNHUNTER [8], for which one hit is obtained when the alignment obeys to a given pattern. The pattern is described by a succession of "1", when a match must occur, and "0" reflects a "don't care" position. For weight 11 (the same as BLAST), their optimal pattern, 11010010100110111, shows significant enhancement on both specificity, because the model has a smaller hit probability in low-similarity regions, and sensitivity, through a greater hit probability in high-similarity regions.

Other programs, including FASTA [9], CHAOS [4] or BLAST2 [2], generate of two or more hits. FASTA looks for the 10 best scores obtained by grouping together several nearby short seeds (4 to 6 nucleotides). CHAOS chains nearby hits: while into an obvious alignment, it is cheap to extend it with the heuristic, thus reducing the final search space. BLAST2 links two close hits if the rigorous alignment between the two hits does not overcome a threshold score.

2 New anchoring ideas

General purpose processors are dedicated to arithmetic operations and do not handle strings efficiently. For example, the most efficient dynamic programming implementation in software for string alignment is PARALIGN [10], which uses the special multimedia instructions of Pentium processors, can achieve 150 million MCPS (matrix cell per second) on a Pentium III at 400MHz (28M transistors) when a dedicated chip with 0.3M transistors running at 50MHz can perform 3200 million MCPS [6]. This illustrates how a hardware implementation allows string processing more easily and efficiently; on the same way, we translate the heuristic step into a hardware filter, for which execution time is not bounded to the size of the anchor. Our limitation is merely the maximum frequency of implementation that implies some regularity properties on the heuristic algorithm.

We decided to limit the length of anchors to the size of a small exon, say 30 to 40bp, on the assumption that exonic parts are better conserved throught evolution. Default values of CHAOS and BLAST2 go that way: they respectively search a 34bp and a 40bp zone. On the other hand, small anchors are of limited interest for specificity and from a statistical point of view, as increasing their size limits the probability of getting

a hit by chance [7].

Non-position-specific patterns: Looking for a given pattern is quite simple, particularly for an exact match subsequence, but may be too restrictive because it introduces tough constraints on the position of matches. Our idea is that just "at least k in n" functions can help us in finding weak seeds that lead finally to a more sensitive algorithm. For too small k/n ratios, the heuristic obviously yields a lot of erroneous alignments. Moreover, 35% identity between two coding sequences is the threshold that allows a match to be accurately identified [3].

Association of patterns: The kind of work that is done by BLAST2 or CHAOS is to produce a large number of seeds, but they generate a hit only when two or more seeds respect some association rule, the simplest beeing proximity. Our idea is to impose a given level of identity on the neighborhood of the anchors to dismiss isolated ones. The basic function used is: generate a hit if (a) and ((b) or (c)), with (a) an anchor generated by a selective method; (b) and (c) are sensible filters on the left and right neighborhood of (a).

3 Methods for heuristic validation

For testing our ideas, we used a panel of Genbank sequences downloaded from the NCBI ftp site. For the database, we chose the genome of Escherichia_Coli_K12 (4.1M DNA base pairs (bp)), and the human chromosoms 20 (55.9 Mbp) and 21 (34.3 Mbp). For the query sequences, we compiled a various set of ESTs (Expressed Sequence Tag) whose size range from 300 to 500 bp.

Sensitivity and specificity: Sensitivity and specificity are the two metrics usually used to measure the accuracy of a prediction [5]. In the field of the sequence alignment, a heuristic can be seen as the prediction of a possibly relevant alignment. Thus, the first step of our method determines exactly the local alignments between the database and the query sequence. This is done with the SSEARCH implementation of Smith-Waterman algorithm (version 3.4) [11]. Then, when testing a heuristic we run the database against the query and we check if a sequence that gave at least one hit belongs to the output of SSEARCH; if so, the sequence is considered as a *true positive* (TP); if not, it is considered as a *false positive* (FP). The remaining sequences of the output of SSEARCH are the *false negatives* (FN). These three values are added respectively along the experiments, that is to say for each couple database/query. We finally calculate the sensitivity Sn= TP/(TP+FN) and the specificity Sp=TP/(TP+FP). These tests allowed us to select efficient PATTERNHUNTER-like anchoring schemes with increased size, up to 31 bp with a weight 19.

Sensitivity versus similarity: Sensitivity can be seen as the probability of obtaining a hit when aligning the query and the database at a given similarity. Thus, on the corresponding curve, we can compare manifold heuristics by looking at the slope at the inflexion point. We generated the databases at each level of similarity by introducing the corresponding number of mutations into the query sequence. These tests have shown the superiority of the "k in n" filters when more selectivity is desired.

Comparison to BLAST: BLAST beeing the most widely used method to perform fast alignments, it seems interesting to compare specifically to it. We ran systematic tests of the BLAST heuristic with an associative pattern as defined in part 2: the same-size BLAST hit is extended with "k in n" filters. We first verified on 100.000 HSPs generated by BLAST that the extensions selected occurred in more than 99% of the alignments. Then we performed the extension on the neighborhood of raw hits, looking for high selectivity, i.e. the lowest number of results. For instance, extending the blast anchor with a "8 in 16" filter eliminates 96% of generated hits for a quality loss of only 0.8%.

As the filtering is handled by a hardware preprocessing that introduces latency but no delay, we can expect our first raw implementation to run more than 25 times faster than BLAST.

References

1. Altschul, S.F. et al. (1990) Basic local alignment search tool. J Mol. Biol., 215, 403-410.
2. Altschul, S.F. et al. (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. Nucleic Acids Res., 25, 3389-3402.
3. Anderson, I. and Brass, A. (1998) Searching DNA databases for similarities to DNA sequences: when is a match significant? Bioinformatics, 14, 349-356.
4. Brudno, M. and Morgenstern, B. (2001) Fast and sensitive alignment of large genomic sequences.Proc. IEEE CS Conf. Bioinformatics, 138-148.
5. Burset, M. and Guigo, R. (1996) Evaluation of gene structure prediction programs. Genomics, 34, 353-367.
6. Han, T. and Parameswaran, S. (2002) SWASAD: An ASIC design for high speed DNA sequence matching. Proc. 15th Int. Conf. on VLSI Design, IEEE CSP.
7. Kent, J.W. (2002) BLAT - The BLAST-like alignment tool. Genome Res., 12, 656-664.
8. Ma, B., Tromp, J. and Li, M. (2002) Patternhunter: Faster and more sensitive homology search. Bioinformatics, 18, 440-445.
9. Pearson, W.R. and Lipman, D.J. (1988) Improved tools for biological sequence comparison. Proc. Natl. Acad. Sci. USA, 85, 2444-2448.
10. Rognes, T. (2001) ParAlign: a parallel sequence alignment algorithm for rapid and sensitive databases searches. Nucleic Acids Res., 29, 1647-1652.
11. Smith, T. and Waterman, M. (1981) Identification of common molecular subsequences. J. Mol. Biol., 221, 403-420.