Available online at www.sciencedirect.com

SCIENCE DIRECT®

Journal of Computer and System Sciences 70 (2005) 342–363

JOURNAL OF COMPUTER AND SYSTEM SCIENCES

www.elsevier.com/locate/jcss

Designing seeds for similarity search in genomic DNA

Jeremy Buhler[a,*], Uri Keich[b], Yanni Sun[a]

[a] *Department of Computer Science and Engineering, Washington University, St. Louis, MO 63130, USA*
[b] *Department of Computer Science, Cornell University, 4130 Upson Hall, Ithaca, NY 14853, USA*

Received 25 January 2004; received in revised form 16 September 2004

Abstract

Large-scale comparison of genomic DNA is of fundamental importance in annotating functional elements of genomes. To perform large comparisons efficiently, BLAST (Methods: Companion Methods Enzymol 266 (1996) 460, J. Mol. Biol. 215 (1990) 403, Nucleic Acids Res. 25(17) (1997) 3389) and other widely used tools use seeded alignment, which compares only sequences that can be shown to share a common pattern or "seed" of matching bases. The literature suggests that the choice of seed substantially affects the sensitivity of seeded alignment, but designing and evaluating seeds is computationally challenging.

This work addresses the problem of designing a seed to optimize performance of seeded alignment. We give a fast, simple algorithm based on finite automata for evaluating the sensitivity of a seed in a Markov model of ungapped alignments, along with extensions to mixtures and inhomogeneous Markov models. We give intuition and theoretical results on which seeds are good choices. Finally, we describe *Mandala*, a software tool for seed design, and show that it can be used to improve the sensitivity of alignment in practice.

Keywords: Genomic DNA; Biosequence comparison; String matching; *Seeded alignment*; *Mandala*

1. Introduction

Genomes and genomic sequence databases provide a fundamental reference tool for molecular biologists. These databases are used primarily to search for DNA sequences similar to (i.e. differing by few mutations from) a query sequence, or for pairs of sequences similar to each other. Applications of

[*] Corresponding author.

E-mail addresses: jbuhler@cse.wustl.edu (J. Buhler), keich@cs.cornell.edu (U. Keich), yanni@cse.wustl.edu (Y. Sun).

0022-0000/$ - see front matter © 2005 Published by Elsevier Inc.
doi:10.1016/j.jcss.2004.12.003

similarity search include detecting repetitive elements [35] and noncoding parts of genes, augmenting the power of gene-structure prediction [23], comparing whole genomes [13], and identifying sequences of unknown origin or function. Public genomic DNA sequence databases such as GenBank are growing exponentially [28], driving demand for fast comparison algorithms and heuristics that nonetheless are as sensitive as possible to biologically meaningful sequence conservation.

Seeded alignment is the dominant paradigm for accelerating large-scale genomic sequence comparison. BLAST [3,2,4] and other widely used tools [34,21] apply alignment algorithms like Smith-Waterman [36] only to pairs of sequences that exhibit prior evidence of similarity in the form of a shared *seed*, typically a common short substring or *word* of matching bases. [1] All matching words between two sequences can be found quickly, so seeded alignment efficiently directs computational resources toward pairs of sequence regions most likely to exhibit high similarity. The words in a sequence database can also be statically indexed [8,21] to accelerate subsequent searches for word matches.

While words are the most popular type of seed for seeded alignment, discontiguous patterns of matching bases have seen considerable use in the sequence comparison literature. A discontiguous pattern spanning s bases, unlike a word of length s, requires matching pairs of bases at only a subset of the positions $\{0, 1, \ldots s - 1\}$. Califano and Rigoutsos, in their FLASH comparison tool [8], found that randomly chosen discontiguous patterns in practice yielded the highest sensitivity to pairs of similar sequences when used to index a database. Buhler [6] formally established the sensitivity of random patterns in developing the randomized LSH-ALL-PAIRS comparison algorithm. Discontiguous patterns have also been used to accelerate seeded alignment algorithms including that of Pevzner and Waterman [31] and, more recently, the BLASTZ algorithm [33,34]; Ma and co-workers' [25,26]; and work by Brejova et al. [5].

PatternHunter introduced an important formal innovation to seeded alignment: the *resource-constrained paradigm* of seed design. This paradigm fixes the computational cost of seeded alignment a priori by fixing the number of different seeds to be used and the approximate false-positive rate for each seed. It then asks how to choose seeds that maximize the probability of detecting ungapped alignments described by a probabilistic model.

The resource-constrained paradigm of seed design is well-suited to BLAST-like tools, in which the cost of using more than one or a few seeds to search a database is unacceptably high, as well as to static indexing schemes in which the number of indices, and hence of seeds, can be larger but is constrained by storage and disk access costs. However, actually designing seeds in the resource-constrained paradigm, even for simple probabilistic alignment models, is computationally challenging. Moreover, most existing work on resource-constrained seed design (with the notable exception of [5]) does not consider alignment models more informative than an i.i.d. random sequence of matches and mismatches.

This work describes tools for resource-constrained seed design. We address the following problem:

> *Given a collection of ungapped genomic sequence alignments of fixed length ℓ, whose distribution of matching base pairs is described by a kth-order Markov model $\mathcal{M}$, and resource limits w and n, find n seeds $\pi_1 \ldots \pi_n$, each inspecting w bases, such that the* ***sensitivity****, or probability that at least one seed detects a random alignment from $\mathcal{M}$, is maximized.*

[1] BLASTN, unlike BLASTP, does *not* compute a neighborhood of each word in the query because typical word lengths are much longer for DNA than for protein.