ISSN 1746-7659, England, UK
Journal of Information and Computing Science
Vol. 8, No. 4, 2013, *pp*. 264-274

A Theoretical Study of Extreme Parallelism in Sequence Alignments

Chung-E Wang

Department of Computer Science, California State University, Sacramento

(*Received March27, 2013, accepted August 25, 2013*)

Abstract. In this paper, we describe fully parallelized architectures for one-to-one, one-to-many, and many-to-many sequence alignments using Smith-Waterman algorithm. The architectures utilize the principles of parallelism and pipelining to the greatest extent in order to take advantage of both intra-sequence and inter-sequence parallelization and to achieve high speed and throughput. First, we describe a parallelized Smith-Waterman algorithm for general single instruction, multiple data (SIMD) computers. The algorithm has an execution time of $O(m+n)$, where m and n are the lengths of the two biological sequences to be aligned. Next, we propose a very-large-scale integration (VLSI) implementation of the parallel algorithm. Thirdly, we incorporate a pipelined architecture into the proposed VLSI circuit, producing a pipelined processor that can align a query sequence with a database of sequences at the speed of $O(m+n+L)$, where m is the length of the query sequence and n and L are the maximum length and the number of sequences in the database, respectively. Finally, we make use of our pipeline architecture to perform all possible pairs of pair-wise alignments for a group of L sequences with a maximum sequence length of m in $O(mL)$ time. Checking all pairs of pair-wise alignments is essential to the overlap-layout-consensus (OLC) approach for de novo assembly.

Keywords: Sequence alignment, sequence database search, Smith-Waterman algorithm, parallel algorithm, VLSI circuit, pipelined architecture, de novo assembly, overlap-layout-consensus approach.

1. Introduction

A sequence alignment is a way of matching two biological sequences to identify regions of similarity that may indicate functional, structural or evolutionary relationships between the two sequences. Sequence alignment is a fundamental operation of many bioinformatics applications such as sequence assembly, sequence database search, and short read mapping. There are two different types of sequence alignments: global alignments and local alignments. Global alignment is to find the best alignment across the entire two sequences. Local alignment is to find regions of high similarity in parts of the sequences. Today, local alignments are often preferable. In this paper, we focus on local sequence alignments.

The Smith-Waterman algorithm [1] is the most sensitive algorithm for performing sequence alignment. Here sensitivity refers to the ability to find the optimal alignment. The Smith-Waterman algorithm uses a linear gap function which was later improved by Gotoh with affine gap penalties [2]. The Smith-Waterman algorithm requires $O(mn)$ computational steps, where m and n are the lengths of the two sequences to be aligned. $O(mn)$ is quite efficient for comparing a pair of sequences. However, it's inadequate for performing multiple alignments, a common task for sequence database searches. A sequence database search is to compare a query sequence with a database of sequences and identify database sequences that resemble the query sequence above a certain threshold. To search a database of L sequences, L sequence alignments must be performed. Thus, without any parallelization, a time of $O(mnL)$ is required to search a sequence database, where m is the length of the query sequence and n and L are the maximum sequence length and the number of sequences in the database, respectively.

Due to substantial improvements in multiprocessing systems and the rise of multi-core processors, parallel architectures such as Field Programmable Gate Arrays (FPGAs), Graphics Processing Units (GPUs) and Very-Large-Scale-Integration (VLSI) circuits have been used to accelerate the Smith-Waterman algorithm and sequence database searches [3-28]. However, most of these previously established enhancements can only speed up the algorithm by a constant factor. That is, most of these enhancements still require an execution time of $O(mn)$ to align two biological sequences and thus require $O(mnL)$ time for a sequence database search.

In [6], Hurley claimed without giving details that using *O(n)* processing elements a sequence database search can be done in *O(n+N)* time, where *n* is length of the query sequence and *N* is the size of the database. This is theoretically impossible. In theory, using *p* processing elements, you can only speedup an algorithm's execution time from *T* to *T/p*. Sequentially, the best execution time for a sequence database search is $O(n^2N)$, where *N* is the size of the database and *n* is the length of the query sequence and database sequences. Thus, the best can be achieved with *O(n)* processing elements is $O(n^2N)/O(n)$ which is *O(nN)* not *O(n+N)*.

In [12], Rajko et al. showed that they can use *p* processors to cut the execution time of aligning two sequences from *O(mn)* down to *O(mn/p)* as long *as p=O(n/log n)*, where *m* and *n* are lengths of the two sequences to be aligned. Since *p* is limited by *O(n/log n)*, in the extreme, their algorithm has an execution time of *O(mlog n)*.

Besides parallel processing, heuristic methods are other commonly used approaches for speeding up sequence database searches. Heuristic methods are intended to gain computational performance, potentially at the cost of accuracy or precision. Popular alignment search tools in this category such as FASTA [29], BLAST [30] and BLAT [31] indeed gain some speed. However, the sensitivity is compromised. For sequence database searches, sensitivity refers to the ability to find all database sequences that resemble the query sequence above a threshold.

Here, we describe a parallel Smith-Waterman algorithm for general Single Instruction Multiple Data (SIMD) computers which achieves a significant improvement in speed over previous algorithms without sacrificing any sensitivity. This parallel algorithm requires an execution time of *O(m+n)* to align two sequences with lengths of *m* and *n*, respectively. Second, we propose a Very Large Scale Integration (VLSI) implementation of the parallel algorithm. Thirdly, we use the pipeline technique to overlap the execution times of alignment checking of database sequences. The resulting pipeline has a throughput rate of *O(1)* execution time per database sequence. Consequently, the time complexity of the proposed pipeline processor is *O(m+n+L)*, where *m* is the length of the query sequence and *n* and *L* are the maximum sequence length and the number of sequences in the database, respectively. Finally, we expand our pipelined architecture to perform all possible pair-wise alignments for a group of sequences. Checking all pairs of pair-wise alignments is essential to the overlap-layout-consensus (OLC) approach for de novo assembly [32-34]. Without any parallelization, $O(m^2L^2)$ time is required to align all possible pairs of sequences for a group of *L* sequences with a maximum sequence length of *m*. For the same task, our pipeline processor boasts a running time of *O(mL)*.

2. The Smith-Waterman Algorithm

The Smith-Waterman algorithm is used to compute the optimal local-alignment score. Let $A = a_1 a_2 \ldots a_m$ and $B = b_1 b_2 \ldots b_n$ be the two sequences to be aligned. A weight $w(a_i, b_j)$ is defined for every pair of residues a_i and b_j. Usually $w(a_i, b_j) <= 0$ if $a_i \neq b_j$, and $w(a_i, b_j) > 0$ if $a_i = b_j$. The penalties for starting a gap and continuing a gap are defined as g_{init} and g_{ext} respectively. The optimal local alignment score *S* can be computed by the following recursive relations:

$$E_{i,j} = \max \{ E_{i,j-1} - g_{ext}, H_{i,j-1} - g_{init} \} \quad (1)$$

$$F_{i,j} = \max \{ F_{i-1,j} - g_{ext}, H_{i-1,j} - g_{init} \} \quad (2)$$

$$H_{i,j} = \max \{0, E_{i,j}, F_{i,j}, H_{i-1,j-1} + w(a_i, b_j) \} \quad (3)$$

$$S = \max \{ H_{i,j} \}; \quad (4)$$

The values of $E_{i,j}$, $F_{i,j}$ and $H_{i,j}$ are 0 when $i<1$ and $j<1$. The Smith-Waterman algorithm is a dynamic programming algorithm requiring an $m \times n$ matrix, or so-called *alignment* matrix, to store and compute H, E, and F values column by column.

3. An Intra-sequence Parallelized Smith-Waterman Algorithm For SIMD Computers

Intra-sequence parallelization refers to parallelization within a single pair of sequences, in contrast to inter-sequence parallelization, where parallelization is carried out across multiple pairs of sequences.

Figure 1 shows the computational dependencies of the Smith-Waterman alignment matrix. This dependency structure illustrates the feature commonly exploited by existing parallelized Smith-Waterman algorithms in the literature. Our parallelization utilizes the following scheme: Since cells on a bottom-left to top-right diagonal have the