Protein similarity search with subset seeds on a dedicated reconfigurable hardware

Pierre Peterlongo[1], Laurent Noé[2], Dominique Lavenier[1], Gilles Georges[1],
Julien Jacques[1], Gregory Kucherov[2], and Mathieu Giraud[2]

[1] Symbiose, IRISA, INRIA, CNRS, Université Rennes 1
[2] Sequoia/Bioinfo, LIFL, INRIA, CNRS, Université Lille 1
http://www.irisa.fr/remix/arc.html

Abstract. Genome sequencing of numerous species raises the need of complete genome comparison with precise and fast similarity searches. Today, advanced seed-based techniques (spaced seeds, multiple seeds, subset seeds) provide better sensitivity/specificity ratios. We present an implementation of such a seed-based technique onto parallel specialized hardware embedding reconfigurable architecture (FPGA), where the FPGA is tightly connected to large capacity Flash memories. This parallel system allows large databases to be fully indexed and rapidly accessed. Compared to traditional approaches like the Blastp software, we obtain both significant speed-up and better results. As our knowledge, this is the first attempt to exploit modern seed features for parallelizing similarity search.

Key words: similarity search, spaced seeds, subset seeds, indexing, FPGA, reconfigurable architecture, dedicated hardware

1 Introduction

Sequence similarity search is one of the fundamental tasks in genomic research. It mainly aims at locating similar regions in DNA or protein sequences which correspond to biologically relevant "conserved" regions. A typical task, for example, is to query a genomic databank with a newly discovered DNA sequence. Observed similarities with other known genes witness their putative common biological function and direct further investigations.

With rapidly growing genomic databases, bioinformatics projects processing hundreds of gigabytes of data represent computationally challenging tasks. They usually do not consider a single gene entity but thousands of them, at the genomic level. As searching for similarities between raw sequences is often the first step to more complex bioinformatics analysis, and as this process requires a huge computing power, there is a great interest in optimizing these computations.

Different approaches have been studied to reduce the computation time while keeping the same sensitivity than Blast, a commonly used software based on a seed-based heuristic [1] (see section 2.1). They all exploit parallelism, but at different levels. One immediate way consists in splitting a genomic databank across a cluster of PC, like in the mpiBLAST implementation [2]. In that scheme, each processor performs independently a search on a part of the databank. A final step merges the results. This parallelization is very efficient since the communication overhead between the PCs is minimal. Another way is to parallelize the algorithm itself on a dedicated hardware (see section 2.2).

This paper presents an implementation of a recently proposed seed-based heuristic, called *subset seeds*, on a parallel hardware designed for indexing large volumes of data such as genomic banks. Two levels of parallelism are considered: a coarse-grained level and a fine-grained level. Here, only the first one will be discussed: it makes a subtle use of subset seeds to simultaneously run several partial searches on large indexes stored in Flash memory. The fine-grained level is the same than what was proposed for fixed seeds of nucleic Blast in [3].

The rest of the paper is organized as follows. The next section introduces a background of similarity search. Section 3 describes our parallel strategy based on subset seeds. Section 4 presents some performance benchmarks obtained on a biological application.

2 Similarity searches

2.1 Seed-based similarity search

An alignment between two sequences is defined in terms of a scoring function minimizing possible substitutions, deletions and insertions needed to transform one sequence into the other. Given a set of scores assigned to those edit operations, dynamic programming (DP) equations compute the best local alignment between two sequences in quadratic time [4]. Some optimizations achieve a subquadratic complexity [5], but the computation time remains prohibitive for whole genome comparisons.

Most of the time, true alignments contain small patterns, called *seeds*, that are shared by the two sequences in an exact way. These seeds are used to reduce DP computations to a small neighborhood of seed occurrences. For example, the Blast [1] single-hit strategy proceeds in 3 stages (Figure 1):

- Stage 1: searching for words of size k (the *seeds*) that occur in both strings,
- Stage 2: extending each seed by allowing a limited number of substitutions, and keeping only those with a score greater than a given threshold,
- Stage 3: applying the full DP algorithm to successfully extended seeds.

About five years ago, it was understood that instead of contiguous k-words, it is more advantageous for Stage 1 to use so-called *spaced seeds* that correspond to gapped diagonals in the DP matrix. The idea of using spaced seeds for biological sequence comparison was first proposed in PatternHunter software [6] and then, used in a more elaborate form in YASS software [7]. Theoretical design and usage of better seeds is an active field of research [8–12]. For protein search, Stage 1 of Blastp looks for words in databank sequences that are sufficiently close (in terms

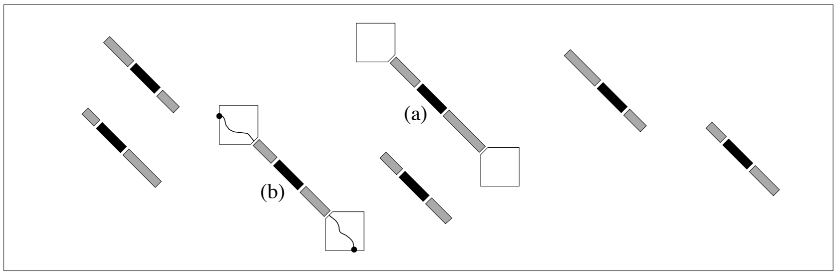

Fig. 1. Schematic view of the Blast 3-stages algorithm. Stage 1: identify exact seeds (black diagonal). Stage 2: compute seed extension allowing a small number of substitution errors (gray diagonal). Stage 3: perform a full DP computation (white square) on remaining seeds. Here only the seed (b) leads to an output alignment.

of the scoring function) to the query word. This strategy is captured by the general concept of *vector seeds* proposed in [10]. Recent works on seed-based protein search [13, 14] study more general seeds at this stage. An important advantage of extended seed models is the possibility of *design* of appropriate seeds according to sensitivity/selectivity criteria and the class of target alignments. Moreover, instead of using a single seed, one can use *several* seeds simultaneously (so-called *multiple seeds*), to further improve the sensitivity/selectivity trade-off.

2.2 Dedicated hardware for similarity search

As in the software case, similarity searches can be exactly performed using specialized hardware. The dynamic programming equations can be projected on 2D or 1D systolic arrays [15–17]. A backtracking phase follows the score computation phase to build the alignment [18]. Special edition scores significantly reduce the hardware resources [16]. Between 1990 and 2006, more than twenty different architectures were proposed on VLSI or FPGA circuits [19]. See [20] or [21] for some recent works.

There are fewer hardware implementations of seed-based heuristics. The first ASIC implementation of a seed-based heuristic was done in 1993 with the BioSCAN architecture [22]. Several FPGA implementations have been independently developed since 2003 (see [19] for a review). Some authors developed both a new algorithm (DASH) with better sensitivity than Blast as well as an FPGA implementation [23].

3 Parallel implementation of subset seeds

To the best of our knowledge, no dedicated hardware has been proposed so far to efficiently implement modern seed-based sequence comparison methods. Moreover, some features of those methods are costly to implement at the software level, but can be easily implemented in hardware.

3.1 Subset seeds for protein searches

The detection of an occurrence of a seed (a *hit*) in the Stage 1 is done by first constructing an index for all patterns corresponding to the seed. Very general seed models, such as vector seeds [10], lead to more expressive hit definitions but also to more complex and less cache-efficient implementations of this process. More specifically, while traditional seeds imply accessing one entry of the index for each query pattern (direct indexing), vector seeds store, for each pattern, the set of all possible patterns that reach a given score threshold. They can thus access the main index several times at non-contiguous positions, inducing a larger latency.

In this work, we use the *subset seeds* model, first proposed in [24] for DNA similarity search. Subset seeds are more expressive than spaced seeds but less expressive than vector seeds. The main idea of subset seeds is that they use elements (seed letters) that distinguish between different types of mismatches. The main advantage of this model is that it provides a powerful seed definition and at the same time preserves the possibility of direct indexing.

Consider the alphabet of amino acids $\Sigma = \{C, F, Y, W, M, L, I, V, G, P, A, T, S, H, Q, E, R, K, D, N\}$. A *subset seed* is defined as a word $s_1 s_2 \dots s_m$ such that:

- each seed letter s_i denotes a partition of the alphabet Σ, grouping amino acids that can be exchanged at this position,
- the subset seed $s_1 s_2 \dots s_m$ matches an alignment fragment $(x_1, y_1)(x_2, y_2)\dots(x_m, y_m) \in (\Sigma^2)^m$ if, for each position i, amino acids x_i and y_i belong to the same set according to partition s_i.

Figure 2 provides an example. Details on how seed letters are chosen are not yet published. Once the seed letters are fixed, we use the approach proposed in [24] to design and select seeds. Theoretical estimates for sensitivity and selectivity are computed from Bernoulli background and foreground models taken from the Blosum-62 matrix models (Blocks database version 5) using the original program of [25]. Seeds achieving the best sensitivity/selectivity ratios are selected for practical evaluation (see section 4).

$$\begin{cases} b_0 = \{CFYWMLIVGPATSNHQEDRK\} \\ b_1 = \{C, FYW, MLIV, G, P, ATS, HQERK, DN\} \\ b_2 = \{C, FYW, ML, IV, G, P, A, TS, H, QE, RK, DN\} \\ b_3 = \{C, F, Y, W, M, L, I, V, G, P, A, T, S, H, Q, E, R, K, D, N\} \end{cases}$$

Fig. 2. Example of seed letters ranging from a *don't care symbol* (b_0, the whole set of amino acids) to a *match symbol* (b_3, the partition into singletons). With this alphabet, the subset seed $s = b_1 b_3 b_2$ matches the alignment fragment $(H, K)(L, L)(F, W)$.

3.2 Hardware search filter

As shown on Figure 3, the hardware prototype architecture, called ReMIX, is composed of 64 GB Flash memory boards, each linked to a FPGA component. An implementation of a seed-based heuristic with fixed seeds was presented in [3]. The key point is that, in the index, each position of each seed pattern is stored *with its neighborhood* (Figure 4), allowing both Stage 1 and Stage 2 to be computed without other memory request.

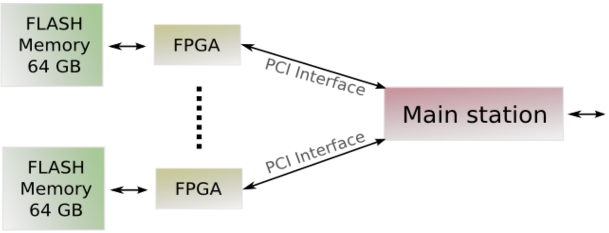

Fig. 3. Principle of the ReMIX architecture. Flash memory boards, linked to a FPGA filter, are linked to a host computer. In this experience, we used four boards.

index key	database pos.	left neighbor	seed	right neighbor
HRT	80743	WGN..IGPG	HRT	QERN..IT
	1403483	ERS..LDLQ	HRT	HWLD..IA
HRV	62716	IKS..GASS	HRV	AKIE..KL
HRW	201235	IGG..PPGI	HRW	SIED..IT

index key	database pos.	left neighbor	seed	right neighbor
H[RK]T	80743	WGN..IGPG	HRT	QERN..IT
	1010697	DSG..HYRW	HKT	KHEL..IT
	1403483	ERS..LDLQ	HRT	HWLD..IA
	1923911	VPR..LRDA	HKT	DVEG..SV
H[RK][ILV]	62716	IKS..GASS	HRV	AKIE..KL
	940017	VVK..FTGQ	HKI	AWLE..KI
	1056293	HYV..IGGD	HKV	RNPH..GM
	1403486	VTF..KDEV	HKI	KARE..QP

Fig. 4. Each index line, on the ReMIX architecture, contains the offset of the seed pattern (database pos.) and its neighborhood (20 amino acids on each side). On the left, the index used on traditional ### and $\overline{###}^{\geq 11}$ seeds gives all the databases positions of occurrences of a given pattern (index key). On the right, the index used with subset seeds gives relatively more data stored at each index key, but the full database size remains the same.

Concretely, Algorithm 1 (below) shows how to find the local alignments between a query and a database, closely following the heuristic exposed in section 2.1. During a preprocessing phase, the databank is indexed off-line with respect to the specified seed (Figure 4). This index is stored into the Flash memory. At runtime, for each position on the query, the index points to the positions matched by the seed s in the database (Stage 1) *and their neighborhoods.* Thanks to the Flash technology, all those data are quickly accessed (the latency of 20 μs for a random access can be hidden by a large number of successive calls), then processed on the FPGA with the neighborhoods in the query (Stage 2). The host

Algorithm 1 Querying a database indexed with one seed

```
Input: database, seed $s$, query
index the database (with respect to $s$)
store the index into Flash memory
for each pattern of the query do
    using the index, focus on similar pattern occurrences in the database (Stage 1)
    for each pattern occurrence in the database do
        using the FPGA, filter out the neighborhoods of the occurrence (Stage 2)
    end for
end for
on host computer, perform DP computations on filtered sets of positions (Stage 3)
Output: local alignments of the query against the database
```

computes the final set of alignments from the remaining set of filtered positions given by the FPGA (Stage 3).

In Stage 1, different seed strategies give different matching patterns in the database. The 3-letter fixed seed ### consists in returning only one pattern for each position on the query (Figure 4, on the left). The single-hit Blastp strategy (seed $\overline{\#\#\#}^{\geq 11}$) uses the same index, but explores all the possible indexes of patterns that score at least 11 when compared with the query pattern, giving a theoretically expected number of 26 index calls on the Blosum-62 background distribution of amino acids. On the contrary, the subset seeds use a different index (Figure 4, on the right). For each position on the query, only one index call is needed, reducing the total latency, and thus the total filtering time. As we use a *set* of n subset seeds, the database is indexed following n different ways, and one different index is stored on each board.

4 Performances and conclusion

Our test databank was the hard-masked human chromosome 1 (UCSC Release hg18) translated with respect to the six reading frames (85×10^6 amino acids). The query was a set of seven archea and bacteria proteoms (5.5×10^6 amino acids) deriving from a study on mitochondrial diseases. The goal was to detect potential insertions of mitochondrial genes in the human genome. We selected distinct sets consisting of 1, 2, and 4 subset seeds from the seeds with the best sensitivity/specificity compromises (see section 3.1), with a sensibility comparable to the Blastp seed $\overline{\#\#\#}^{\geq 11}$.

Using one or several boards, we performed tests parallelizing all the algorithm 1, except Stage 3 that is computed by the host computer on the merged results from all the boards. The computation time of Stage 3, lower than Stage 1 and 2, is hidden by successive calls on queries.

Time and data results are shown Table 1. On average, FPGA filters takes approximatively 67 nanoseconds for processing one index line, representing around 15 millions of index entries filtered each second.

Seed model	n	sensitivity	index calls ($\times 10^6$)	positions returned ($\times 10^9$)	filtering time (min:sec)
Fixed seed ###	1	92.87%	5.4	24	24:13
Blastp seed $\overline{\texttt{\#\#\#}}^{\geq 11}$	1	99.09%	92.9	246	257:50
Subset seed n°1	1	99.11%	5.4	231	216:25
Subset seeds n°2	2	99.14%	5.4	max: 109	max: 103:51
Subset seeds n°3	4	99.13%	5.4	max: 69	max: 65:35

Table 1. Comparison between different seeds. The fixed seed ### is given for reference. The value n is the number of seeds considered in the set : the computation is distributed on n boards. Experimental values for sensitivity are obtained through comparison with Smith-Waterman alignments on human chromosomes 1 – 11. All subset seeds presented here were choosen to have a better sensitivity than $\overline{\texttt{\#\#\#}}^{\geq 11}$. Here the $\overline{\texttt{\#\#\#}}^{\geq 11}$ seed calls the index 17 more times than the subset seeds. The number of returned database positions estimates the selectivity. When several seeds and boards are used ($n > 1$), we show the results of the slowest one. As a reference, the usual single-hit Blastp implementation would take more than 3400 minutes on this dataset with a 3 GHz PC.

Even with traditional Blastp seeds, one board with the architecture provides a 13× speed-up over usual software implementation. As the cost of the FPGA circuit and the Flash memory declines, a convenient speed-up is obtained by joining several PCI boards inside a host PC [3].

Moreover, the use of the subset seeds gives an additional speed-up due to reduced access in the memory. Here, the best results (reported to the number of boards) are achieved with a set of 2 subset seeds, giving a 24% speed-up over the implementation of Blastp seeds. Regarding previous results, a simple host computer equipped with 4 boards with those subset seeds is equivalent to a 64-node cluster.

In conclusion, a better algorithmic design of seeds provides an additional speed-up on the same architecture. One possible extension is to design seeds optimized to index only a subpart of the databases positions. This would reduce the index size, and thus speed up the search, but the sensitivity of such seeds remains to be studied. Another open question is: can similar seeds be considered to speed up nucleic similarity searches?

Acknowledgments. Support for this work was provided by INRIA through the grant ARC Flash “Seed Optimisation and Indexing of Genomic Databases”.

References

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

2. Thorsen, O., Smith, B., Sosa, C.P., Jiang, K., Lin, H., Peters, A., Fen, W.: Parallel genomic sequence-search on a massively parallel system. In: Int. Conference on Computing Frontiers (CF 07). (2007)
3. Lavenier, D., Xinchun, L., Georges, G.: Seed-based genomic sequence comparison using a FPGA/FLASH accelerator. In: FPT 06. (2006) 41–48
4. Smith, T., Waterman, M.: Identification of common molecular subsequences. Journal of Molecular Biology **147**(195–197) (March 1981)
5. Crochemore, M., Landau, G., Ziv-Ukelson, M.: A sub-quadratic sequence alignment algorithm for unrestricted cost matrices. In: Symposium On Discrete Algorithms (SODA 02). (2002) 679–688
6. Ma, B., Tromp, J., Li, M.: PatternHunter: Faster and more sensitive homology search. Bioinformatics **18**(3) (2002) 440–445
7. Noé, L., Kucherov, G.: YASS: enhancing the sensitivity of DNA similarity search. Nucleic Acids Research **33** (2005) W540–W543
8. Csürös, M., Ma, B.: Rapid homology search with two-stage extension and daughter seeds. In: Int. Computing and Combinatorics Conference (COCOON 05). 104–114
9. Buhler, J., Keich, U., Sun, Y.: Designing seeds for similarity search in genomic DNA. Journal of Computer and System Sciences **70**(3) (2005) 342–363
10. Brejová, B., Brown, D., Vinar, T.: Vector seeds: An extension to spaced seeds. Journal of Computer and System Sciences **70**(3) (2005) 364–380
11. Li, M., Ma, M., Zhang, L.: Superiority and complexity of the spaced seeds. In: Symp.on Discrete Algorithms (SODA 06). (2006) 444–453
12. Mak, D., Gelfand, Y., Benson, G.: Indel seeds for homology search. Bioinformatics **22**(14) (2006) e341–e349
13. Kisman, D., Li, M., Ma, B., Li, W.: tPatternhunter: gapped, fast and sensitive translated homology search. Bioinformatics **21**(4) (2005) 542–544
14. Brown, D.: Optimizing multiple seeds for protein homology search. IEEE Transactions on Computational Biology and Bioinformatics **2**(1) (2005) 29–38
15. Kung, H.T., Leiserson, C.: Algorithms for VLSI processors arrays. Addison-Wesley (1980)
16. Lipton, R., Lopresti, D. In: A systolic array for rapid string comparison. H. Fuchs, Ed. Rockville, MD: Computer Science Press (2004) 363–376
17. Chow, E., Hunkapiller, T., Peterson, J.: Biological information signal processor. In: Int. Conf. on Application Specific Array Processors (ASAP 91). (1991)
18. Hoang, D.: Searching genetic databases on splash 2. In: FCCM'93, IEEE Workshop on FPGAs for Custom Computing Machines, Napa, California (1993) 185–191
19. Lavenier, D., Giraud, M.: Bioinformatics Applications. In: Reconfigurable Computing. Springer (2005)
20. Dydel, S., Bala, P.: Large scale protein sequence alignment using FPGA reprogrammable logic devices. In Springer, ed.: FPL'04, LNCS 3203. (2004) 23–32
21. Court, T.V., Herbordt, M.C.: Families of FPGA-based algorithms for approximate string matching. In: ASAP 04. (2004)
22. Singh, R., al.: A Scalable Systolic Multiprocessor System dor Analysis of Biological Sequences. In: Research on Integrated System. G. Borrielo and C. Ebeling (1993)
23. Knowles, G., Gardner-Stephen, P.: A new hardware architecture for genomic and proteomic sequence alignment. In: Comp. Systems Bioinformatics Conf. (2004)
24. Kucherov, G., Noé, L., Roytberg, M.: A unifying framework for seed sensitivity and its application to subset seeds. J. Bioinf. Comp. Biology **4**(2) (2006) 553–569
25. Henikoff, S., Henikoff, J.: Amino acid substitution matrices from protein blocks. Proc. Natl. Acad. Sci. USA **89** (1992) 10915–10919