An Integrated Approach to Fast, Sensitive, and Cost-Effective Smith-Waterman Multiple Sequence Alignment

Frank Lau

December 15, 2000
MB&B 452: Bioinformatics Module

Introduction

Multiple Sequence Alignment (MSA) is a critical tool for phylogenetic analysis, homologous gene finding, and gene clustering. The most sensitive method for MSA was first described by Smith-Waterman algorithm (SW) (1981)[1] and improved by Gotoh (1982)[2]. The SW algorithm represents the degree of similarity between each pair of the sequences with a similarity score. A straightforward implementation calculates this for each query sequence-database sequence pair and is accordingly computationally intensive: For a query of length *m* and a database sequence of length *n*, the matrix requires a running time proportional to *mn*.

As nucleotide databases double in size every 15 months[3], the computational resources required for sequence database searches will rise exponentially. Although computing resources are also increasingly exponentially, the demands of multiple sequence alignment will soon outstrip those gains. Thus, MSA of databases will take a longer time to perform.

We can improve alignment times by either reducing the alignment sensitivity or by developing specialized hardware. Currently, the first solution is implemented in heuristic methods such as FASTA[4] and BLAST[5]. These algorithms, by hashing and subdividing the sequences, are up to 40 times faster than straightforward implementations of SW. However, they are less sensitive to distantly related sequences and hence biologically relevant information can be lost. The second solution comes in the form of parallel processing hardware such as Paracel's GeneMatcher and Compugen's Bioccelerator. However, this method has the setback of being prohibitively expensive for most users.

Two recent papers significantly reduce SW calculation times in novel fashion. The first, "Computational space reduction and parallelization of a new clustering approach to large groups of sequences" (Trelles, 1998)[6] relies on a clustering approach to remove unnecessary pairwise calculations. The second, "Six-fold speed-up of Smith-Waterman sequence database searches using parallel processing on common microprocessors" (Rognes and Seeberg, 2000)[7] takes advantage of commonly available Single-Instruction, Multiple-Data (SIMD) processors to perform parallel calculations without specialized hardware. The first approach yields a maximum of 98% speed reduction with eight parallel processors. The benchmark for this comparison is the SSEARCH algorithm (Pearson, 1991)[8], a standard variant of SW. The second approach allows us to reduce the number of processors to one. The integration of these two approaches optimizes SW MSAs, with optimization being the best combination of speed, sensitivity and cost.

Computational Space Reduction (CSR)

The innovative method described by Trelles *et al* relies on finding the subset of database sequences that fall under a 'Special Interest Group' (SIG) and comparing the similarity scores to two similarity thresholds: T_I and T_S. The SIG is found by randomly selecting a 'leader' sequence X_i from the entire set of sequences $\{X_1, X_2, \ldots X_n\}$. A set of similarity scores $S_{i,j}$ is then computed between the leader and the remaining *n-1* sequences, which are represented as X_j.

T_I and T_S now come into play. T_I is the sequence similarity threshold as described by Sander and Schneider (1991)[9]; it is dependent on the length of the alignment. Effectively, it represents the minimum value indicates meaningful similarity between two sequences. T_S is a higher threshold meant to provide a "safety" level of similarity; any sequence with a score above T_S has a very high certainty of being similar to the query sequence.

Each $S_{i,j}$ is compared against T_I and T_S and three possible scenarios result:

1) $S_{i,j} \geq T_S$. X_j belongs to the SIG led by X_i.
2) $S_{i,j} < T_I$. X_j does not belong to the SIG led by X_i.
3) $T_I \leq S_{i,j} < T_S$. X_j potentially belongs to SIG. These sequences are grouped into a "second comparison round" and must re-compared.

Scenario 2 is the key to reducing the computational space. Because X_j is not minimally related to X_i, it is assumed that X_j is unrelated to all of the sequences in SIG. None of the pairwise similarity calculations between X_j and the SIG sequences will then be performed. In the optimal case, Trelles *et al* found that this removed 90.31% of the calculations. However, it is critical to select a correct T_I – if the low threshold is too high, then related sequences will be missed and this approach is ineffectual.

Sequences falling under Scenario 3 are re-compared with the subset of sequences found in Scenario 1. For each of these sequences, if the comparison indicates that $S_{j,j} \geq T_S$ then the sequence belongs to the SIG. Otherwise, discard the sequence by the rules of Scenario 2.

In many cases, more than one SIG will exist for the database. Once the membership of each SIG is found, a complete cross-similarity matrix for each SIG must be computed in the traditional SW manner. Trelles *et al* characterize the MSA as two tasks, with Task 1 (T1) being the determination of the SIGs and Task 2 (T2) being the determination of cross-similarity matrices. In their experience, T2 is the most time-consuming step. However, these tasks are essentially the same as both rely on determining a similarity score by SW methods. Thus, the SIMD method described by Rognes and Seeberg can be applied to reduce hardware cost.

Single-Instruction Multiple Data (SIMD) Computation of $S_{i,j}$

SIMD processors were first introduced under Intel's Pentium MMX brand in 1997. Other companies followed suit and now AMD, HP, Sun and a host of other companies produce SIMD processors, making the technology vastly available and more importantly, inexpensive (Rognes and Seeberg, 2000).

The breakthrough of SIMD architecture is the ability to divide the 64-bit registers into eight 8-bit registers that perform calculations simultaneously. For operations that can exploit this ability, this represents a potential 8-fold increase in speed. However, each calculation within the $S_{i,j}$ matrix, as implemented in a straightforward fashion, depends on the calculations from one step earlier. This prevents parallelization.

Earlier approaches overcame this obstacle by performing calculations along the minor diagonal as these are independent (Hughey, 1996; Wozniak, 1997)[10,11]. Rognes and Seeberg, on the other hand, perform calculations parallel to the query sequence. This does not remove cell dependency, but they overcome the problem by relying on a SWAT-like optimization. In SWAT (Green 1993)[12], if a given cell has a value of 0 and the local cell-set maximum *h* is less than the sum (*q* + *r*) of the gap-open penalty *q* and the gap extension penalty *r*, then the entire column or row will stay at zero. This is only effective when *r* is large. If the entire column or row is zero, then cell dependencies are removed and calculations can be avoided. Else wise, a straightforward implementation is called for, but even then represents only a small portion of the entire matrix. This method of calculations is advantageous because it simplifies and speeds up the loading of substitution scores from memory.

A Solution to Computational Bottlenecks and Inefficiencies

A difficulty arising from the division of the processor register into eight sub-registers is that it limits the values that we can process to 0-255. For most matrices, this is not a problem as the overall similarity score will be less than 255. However, in the calculation of the T2 SIG cross-similarity matrices, in which sequences that have been clustered together are all compared in pairwise fashion, it is possible that the score will exceed 255. Rognes and Seeberg solution is to re-compute the score without the SIMD optimization. However, this potentially removing the benefits of integrating the SIMD and CSR approaches because whereas in the CSR approach there were eight processors that could recomputed T2 tasks simultaneously; in SIMD there is only one processor.

Fortunately, there are inefficiencies in the CSR approach and the integration of both approaches may allow us to capitalize on them, thereby improving on both methods. Trelles *et al* found that the marginal returns on increasing the number of processors rapidly declined. Tracing the method, they found the cause: T1 tasks often fail to fully occupy the processors. If the number of processors exceeds the number of T1 tasks, then adding more does no good.

The solution to this inefficiency is to embed, in the software, algorithms for setting up dynamic queues of both tasks waiting to be performed and idle processors. Moreover, T1 and T2 tasks should not be performed in separate batches; the fastest computations came when T2 tasks generated by early T1 calculations were loaded onto processors that were idle during the T1 phase.

Integrating these two solutions, it may then be possible to dynamically reprogram the algorithm such that if any 8-bit registers were idle during any phase, they could be combined to re-compute cross-similarity matrices whose scores exceeded 255. The limit to the score would now be 65,535 and breaching this limit is very unlikely. The earlier analysis indicates that in many cases, scores will be less than 255. The minimal requirement for reaching the 16-bit register limit are two identical sequences of 65,535 units in length.

Discussion

In their paper, Rognes and Seeberg report that their single Pentium III 500 MHz processor generated a speed up factor of 6.2 over the SSEARCH. This increase in speed applied equally to both short and long sequences. The average number of cell updates was 156 million per second. The fastest search algorithm, the NCBI BLAST 2.0.10, was 7.5 times faster, or 87% more efficient. However, it must be noted that the BLAST search is less sensitive.

With the CSR approach, Trelles saw a 93% reduction in CPU-time. If the integrated approach can fully capture that reduction, for the first time it will be possible for the SW algorithm to be faster than the newest, most sensitive BLAST algorithm.

Moreover, Rognes and Seeberg indicate that a symmetric multiprocessing computer with eight Pentium III 600 MHz Xeon processors can potentially achieve 1500 million cell updates per second. This represents a 1.2 increase in speed per processor and surpasses the speed of a MasPar MP-2 computer with 16 384 CPUs. Perhaps more significantly, the eight processors represent 64 8-bit registers, thus completely removing the bottleneck that was encountered in re-computing T2 matrices. In the extreme case of scores exceeding 65,535, we can simply recomputed with 3, or 4 registers which yield score limits of (2^{24}-1) and (2^{32}-1) respectively. Limits of this size allow us to simultaneously align entire genomes. The main concern now would be the cost of the multiprocessing computer as this brings us back to the original barrier of expensive hardware.

This limitation can be overcome, however. Even today, processors with 128-bit registers are now available in the form of PowerPC G4 processors. This represents 16 8-bit registers and with a robust algorithm, protein MSAs will not even need a multiprocessing computer to fully capitalize on the power of this integrated approach.

Finally, a large, clustered network of inexpensive computers can also replace the multiprocessing computer. This approach might see significant losses in sequence loading time reductions, but as databases grow, but as databases grow, this may represent the most cost-effective approach.

---

[1] Smith, T.F. and Waterman, M.S. (1981) Identification of common molecular subsequences. *J. Mol. Biol.*, **147**, 195-197.

[2] Gotoh, O. (1982) An improved algorithm for matching biological sequences. *J. Mol. Biol.*, **162**, 705-708.

[3] Benson, D.A., Karsch-Mizrachi,I, *et al*. (2000) Genbank. *Nucleic Acids Res.*, **28**, 15-18.

[4] Pearson, W.R., Lipman, D.J. (1988) Improved tools for biological sequence comparison. *PNAS*, **86**, 2444-2448.

[5] Altschul, S.F., Gish, W. *et al*. (1990) Basic local alignment search tool. *J. Mol. Biol.*, **215**, 403-410.

[6] Trelles, O. *et al*. (1998) Computational space reduction and parallelization of a new clustering

approach to large groups of sequences. *Bioinformatics*, **14**, 439-451.
[7] Rognes, T. and Seeberg, E. (2000) Six-fold speed-up of Smith-Waterman sequence database searches using parallel processing on common microprocessors. *Bioinformatics*, **16**, 699-706.
[8] Pearson, W.R. (1991) Searching protein sequence libraries: comparison of the sensitivity and selectivity of the Smith-Waterman and FASTA algorithms. *Genomics*, **11**, 635-650.
[9] Sander, C. and Schneider, R. (1991) Database of homology derived protein structures and the structural meaning of sequence alignment. *Proteins*, **9**, 56-68.
[10] Hughey, R. (1996) Parallel hardware for sequence comparison and alignment. *Comput. Applic. Biosci.*, **12**, 473-479.
[11] Wozniak, A. (1997) Using video-oriented instructions to speed up sequence comparison. *Comput. Appl. Biosci.*, **13**, 145-150.
[12] Green, P. (1993) SWAT. http://www.genome.washington.edu/uwgc/analysistools/swat.htm