

# DASH: Localising Dynamic Programming for Order of Magnitude Faster, Accurate Sequence Alignment

Paul Gardner-Stephen, Greg Knowles,  
Embedded Systems Laboratory,  
School of Informatics & Engineering, Flinders University,  
GPO BOX 2100, Adelaide 5001, AUSTRALIA  
gardners@infoeng.flinders.edu.au gknowles@infoeng.flinders.edu.au

## Abstract

*In this paper we present our genomic and proteomic sequence alignment algorithm, DASH, which results in order of magnitude speed improvement when compared to NCBI-BLAST 2.2.6[1], with superior sensitivity.*

*Dynamic programming (DP) is the predominant contributor to search time for algorithms such as BLAST and FastA/P[2]. Improving the efficiency of DP provides an opportunity to increase sensitivity, or significantly reduce search times and help offset the effects of the continuing exponential growth in database sizes.*

*Specifically, for nucleotide searching we have demonstrated an order of magnitude speed improvement with significantly improved sensitivity, or alternatively moderate speed up with further sensitivity gains, depending on the parameters selected. Smith-Waterman[3] complete DP is used as the sensitivity benchmark. Similar speed and sensitivity results are presented for protein searching.*

*Since our algorithm is highly parallel, we have developed dedicated hardware which we will present in a companion paper[4], and a distributed version of our software (DDASH), which we expect to provide linear speedup on a cluster.*

## 1. Introduction

Much has been published regarding genomic and proteomic searching. An over-simplified summary could be that a number of algorithm developments, such as FastA/P, NCBI-BLAST (BLAST) versions 1[5] and 2, FLASH[6] and CAFE[7], have progressively resulted in speed improvements of around three to four orders of magnitude over a complete search algorithm, such as Smith-Waterman. In general, each successive speed improvement has been at the expense of sensitivity. This process has been largely driven

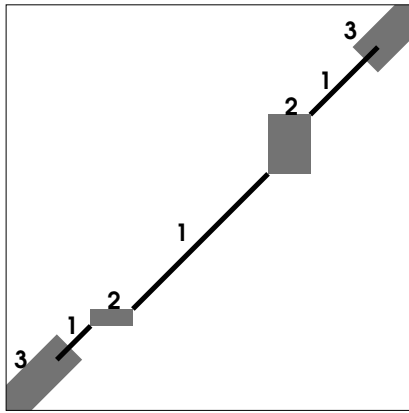
by database size (and hence search time) increasing faster than Moore's Law (which relates to the rate of increase in computational power over time). This successive replacement of search algorithms with faster and less sensitive ones has serious consequences for biologists. It is recognised that relatively few biologists have a good understanding of what is likely to be omitted by a particular search algorithm [8].

### 1.1. The DASH Algorithm

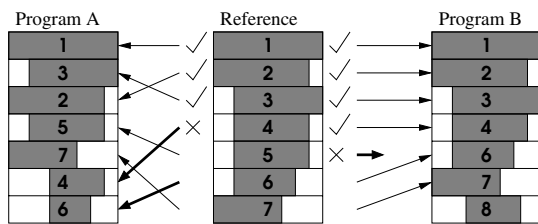
We have developed the Diagonal Aggregating Search Heuristic (DASH) algorithm[9] with this dilemma in mind. This has culminated in a three stage algorithm. First, non-gapped alignments (diagonals) are identified with the aid of an indexed version of the database. Second, these diagonals are aggregated where possible by performing global DP alignment of the regions between them. Finally, DP is performed at each end of the aggregated diagonals. This process, and the regions of DP are illustrated in figure 1, where the products of each stage are appropriately numbered. The reduction in DP load can clearly be seen in that the bands around the non-gapped alignments are not evaluated.

Query sequence filtering in DASH is performed by excluding from the first stage those k-tuples which occur excessively often. This frequency information is stored in, and pertains to each division of the database. This results in a localised filtering algorithm, which takes into account localised sequence frequency aberrations in the database, without penalising the sensitivity of the entire search. A second effect is to allow extension of alignments over excluded regions, since neither the query or subject sequences are altered. Hence DP is able to cover any sequence segments which have been filtered.

To reduce the incidence of outrageously scored low complexity sequences, DASH scores nucleotide alignments involving approximate matches (e.g. N vs G) differentially to exact matches (e.g. G vs G).



**Figure 1. Illustration of DASH algorithm and Dynamic Programming Zones**



**Figure 2. Example of User Satisfaction Sensitivity Measure**

## 1.2. Measuring Sensitivity From a User's Perspective

We have developed a sensitivity metric which is designed to mimic the satisfaction of biologists actually using an on-line search facility. We consider the optimal alignments returned by Smith-Waterman as our benchmark. Only the most significant alignment against any given sequence of the database is considered, since that determines the relative rank of the alignment in the results. From the top, it is tested whether they occur perfectly in the results of the program being scrutinised. As soon as an omission, or partial omission is identified, testing stops. The rank reached is recorded as the sensitivity score. Thus we measure the rank above which we trust that the algorithm has not omitted an alignment. This is illustrated in figure 2. Program A omits part of the fourth alignment of the reference results, and thus scores three. Program B omits the entirety of the fifth alignment, and so scores four. The order of alignments is not considered.

We also use an alternative metric, to demonstrate that the

performance of DASH is not dependent on a single measure. Here the benchmark is the twenty five most statistically significant alignments obtained by Smith-Waterman. We measure the percentage coverage by each program against this benchmark. In this way, we assess both complete and partial matches.

## 2. Results

To verify the DASH algorithm, we performed a comparison of both speed and sensitivity, using Smith-Waterman as a benchmark. Two hundred random complete sequences were selected from a draft of the Human Genome[10] for nucleotide, and the same number from genpept[11] for protein. The Smith-Waterman results were obtained using the seqaln[12] program, our DDASH distribution framework, and approximately 300 CPU days of the APAC and SAPAC super computer facilities. BLAST 2.2.6 results were also obtained for comparison. In all cases, the databases were loaded completely into RAM.

It is clear that for both protein and nucleotide searching DASH is capable of searching an order of magnitude faster than BLAST, while actually improving sensitivity (table 1 and figures 3, 4, 5 and 6, modes 1 and 2). For the DASH modes which approach the run time of BLAST (i.e. modes 3 and 4), even greater improvements in sensitivity are observed. Figure 7 shows the sensitivity of BLAST and DASH using the alternative metric, demonstrating that the performance of DASH is not dependent on a single measure.

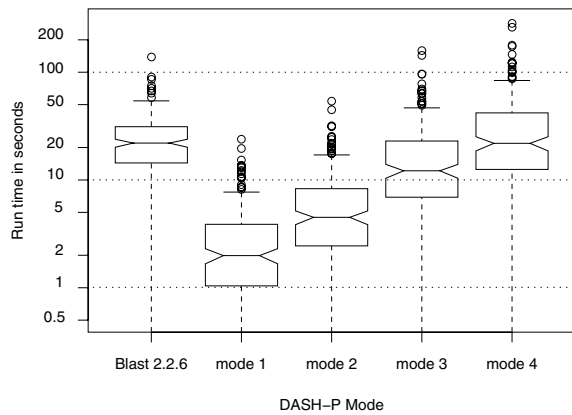
BLAST frequently misses alignments as soon as the first or second rank, particularly for certain classes of nucleotide searches. This is in part due to the query sequence filtering employed by BLAST in its default mode of operation. When the filtering excludes significant portions of the query sequence BLAST tends to fragment alignments which would cover those regions. In light of this, we also considered the 130 cases where BLAST did not filter. BLAST then achieves considerably higher scores (median of 10 versus 2, and mean 19.17 versus 12.11).

DASH in contrast, pushes the median scores beyond the twentieth to twenty-fifth ranks. This represents a marked improvement over existing sequence filtering algorithms, such as [13] - [17].

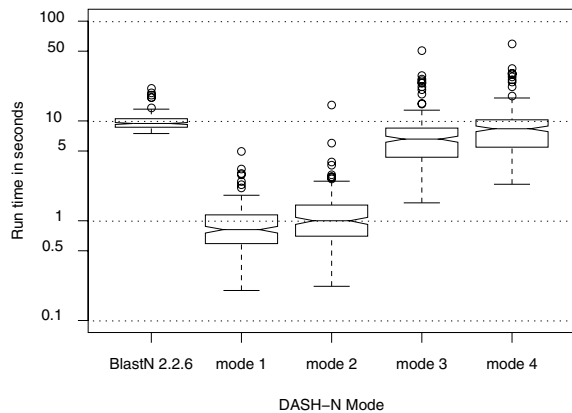
Finally, figure 8 reveals that the speedup of DASH over BLAST depends on the query sequence length. Investigation reveals that this is related to a large static time component (around 7 seconds for nucleotide). It is our understanding that a large fraction of searches submitted fall into the sub 500 residue size range, suggesting that typical speed up in a production environment will be much more than ten times. A similar relationship exists for protein, however with greater variability due to the increased DP load involved.

Program	Min	Med.	Mean	Max.
BLAST-N	7.48	9.40	9.79	21.19
DASH-N 1	0.20	0.82	0.93	4.95
BLAST-P	0.01	21.76	25.23	138.70
DASH-P 1	0.01	1.99	3.22	23.93

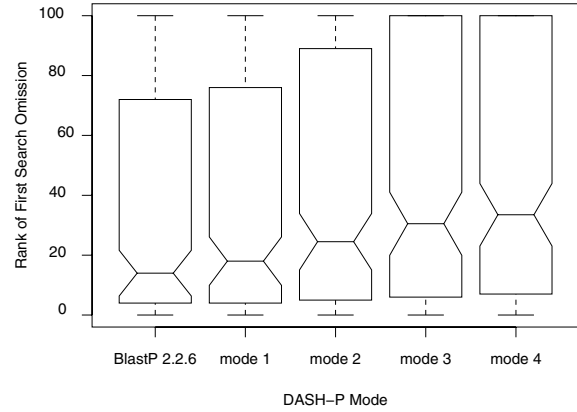
**Table 1. Summary Statistics for DASH mode 1 and BLAST Run Time (seconds)**



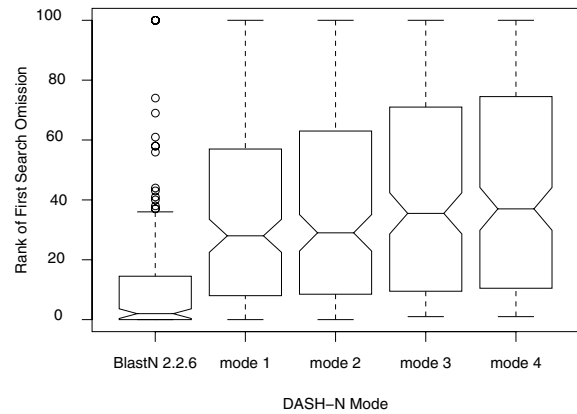
**Figure 3. Speed Comparison for Protein**



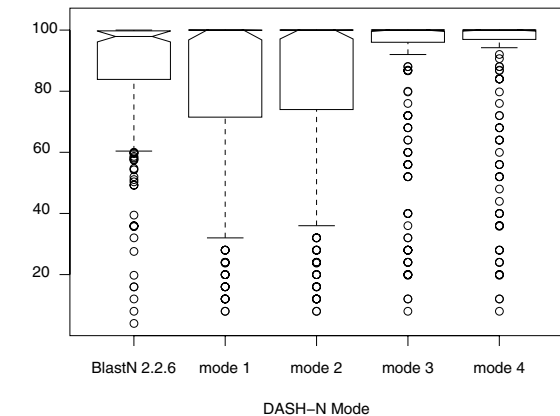
**Figure 4. Speed Comparison for Nucleotide**



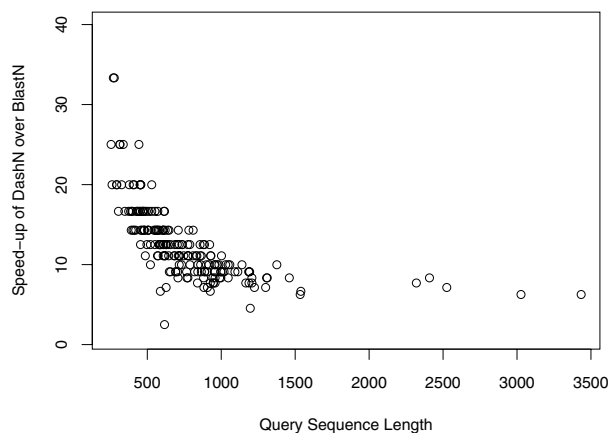
**Figure 5. Sensitivity Comparison for Protein**



**Figure 6. Sensitivity Comparison for Nucleotide**



**Figure 7. % Coverage of Top 25 Smith-Waterman Hits for Nucleotide**



**Figure 8. Query Sequence Length Versus Speedup of DASH-N over BLAST-N**

### 3. Conclusions & Future Directions

In the light of these results, we conclude that DASH is faster than BLAST by an order of magnitude. DASH simultaneously offers greatly improved sensitivity for nucleotide and, significantly improved sensitivity for protein searching. The localised query filtering regime does not suffer from the problems introduced by existing popular filtering algorithms.

We also plan to continue work on our distributed version (DDASH). This environment has been designed to utilise heterogeneous computing nodes connected by commodity networks. This applies to distributed networks of UNIX, Linux, Windows, MacOS-X as well as dedicated clusters. We plan continue to refining our already light weight communication protocol, to further reduce latency induced overheads. We expect to obtain near linear speed up with number of processors. This is an area for future research.

### 4. Acknowledgements

We wish to acknowledge the support of the Australian Co-Operative Research Centre for Sensor, Signal and Information Processing (CSSIP) Firmware for Genomics project, and the Australian and South Australian Partnerships for Advanced Computing (APAC & SAPAC).

### References

[1] S. F. Altschul *et al*, "Gapped BLAST and PSI-BLAST: a new generation of protein database search programs," *Nucleic Acids Res*, vol. 25, pp. 3389-3402, 1997.  
 [2] W. R. Pearson, "Rapid and sensitive sequence comparison with FASTP and FASTA," *Methods in Enzymology*, vol. 183, pp. 63-98, 1990.

[3] T.F. Smith and M. S. Waterman, "Identification of common molecular subsequences," *The Journal of Molecular Biology*, No. 147(1), pp. 195-197, March 1981.  
 [4] G. Knowles, and P. M. Gardner-Stephen, "A new hardware architecture for genomic and proteomic sequence alignments," presented at 3rd IEEE Computational Systems Bioinformatics Conference, Stanford, USA, 2004.  
 [5] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, D. J. and Lipman, "Basic local alignment search tool" *The Journal of Molecular Biology*, vol. 215, pp. 403-410, 1990.  
 [6] A. Califano and I. Rigoutsos, "FLASH: A fast look-up algorithm for string homology," in *International Conference on Intelligent Systems for Molecular Biology*, 1993, pp. 56-64.  
 [7] H. E. Williams J. Zobel, "Indexing and retrieval for genomic databases," *IEEE Transactions on Knowledge and Data Engineering archive*, vol. 14, Issue 1, pp. 63 - 78, January 2002.  
 [8] F. Galisson, "The FASTA and BLAST programs," July 2000, [http://bioweb.pasteur.fr/seqanal/blast/blast\\_fasta-uk.ps](http://bioweb.pasteur.fr/seqanal/blast/blast_fasta-uk.ps)  
 [9] P. M. Gardner-Stephen and G. Knowles, "DASH: A New High Speed Genomic Search and Alignment Tool," in *4th International Conference on Mathematics and Computers in Biology and Chemistry*, 2003, vol. 1, pp. 121-127.  
 [10] National Centre for Biotechnology Information, "Human Genome, Working Draft," June 2002, <ftp://ftp.ncbi.nih.gov/repository/UniGene/Hs.seq.all.gz>.  
 [11] National Centre for Biotechnology Information, "Protein translations of GenBank CDEs," <ftp://www.ncbi.nih.gov/.../genpept.fsa>.  
 [12] P. Hardy and M. Waterman, "The Sequence Alignment Software Library at USC," 1997, <http://www-hto.usc.edu/software/seqaln/>.  
 [13] H. E. Williams, "Effective Query Filtering For Fast Homology Searching," in *4th Pacific Symposium on Biocomputing*, 1999, pp. 214-225.  
 [14] J. C. Wootton and S. Federhen, "Statistics of local complexity in amino acid sequences and sequence databases," *Computers in Chemistry*, vol. 17, pp. 149-163, 1993.  
 [15] J. C. Wootton and S. Federhen, "Analysis of compositionally biased regions in sequence databases," *Methods in Enzymology*, vol. 266, pp. 554-571, 1996.  
 [16] J. M. Hancock and J. S. Armstrong, "SIMPLE34: an improved and enhanced implementation for VAX and Sun computers of the SIMPLE algorithm for analysis of clustered repetitive motifs in nucleotide sequences," *Comput Appl Biosci*, vol. 10, pp. 67-70, 1994.  
 [17] J-M. Claverie and D. J. States, "Information enhancement methods in large scale sequence analysis," *Computers in Chemistry*, vol. 17, pp. 191-201, 1993.