Improving the Practical Space and Time Efficiency of the Shortest-Paths Approach to Sum-of-Pairs Multiple Sequence Alignment

Sandeep K. Gupta *
skgupta@lcs.mit.edu
MIT Laboratory for Computer Science

Introduction

Alignment of multiple DNA and protein sequences is a fundamental problem in molecular biology. A variety of combinatorial definitions of the alignment problem are used in practice. Finding an optimal multiple sequence alignment seems to require time and space exponential in the number of sequences for all definitions, and is provably hard for a few. Nevertheless it is often useful to align a small number of sequences optimally.

One of the existing programs that attempts to construct an optimal alignment, for some definition of multiple sequence alignment, is `MSA` [1, 4]. Most other programs attempt to compute a global multiple sequence alignment. `MSA` is one of the few that attempts to compute an optimal alignment based on some well-defined score. Methods other than `MSA` tend to use much less space because they do not search for an optimal alignment.

The stimulus for this paper is a successful attempt to reduce the space usage of `MSA` by significant multiplicative factors. On some runs, we achieve an order of magnitude reduction in space usage and a significant multiplicative improvement in running time. The space reduction we have achieved makes many runs that were not feasible with the original `MSA` feasible for the first time, and speeds up many runs that before were marginally feasible but would thrash extensively while allocating virtual memory. Many of these thrashing runs can now fit in main memory, which can lead to a huge double-digit multiplicative reduction in wall-clock time.

The resource-intensive part of `MSA` is an implementation of a complex variant of Dijkstra's single-source shortest-paths algorithm. The version of multiple sequence alignment that `MSA` solves can be formulated as a single-source, single-destination shortest-paths problem on a mesh-shaped dynamic programming graph. The dimension of the mesh is the number of sequences, and the number of vertices is the product of the sequence lengths.

To save space the dynamic programming graph is generated "on-the-fly", so that only subgraphs are stored in memory at any given time. The essential idea in our space usage improvements is to prove algorithmic invariants regarding when edges and vertices of the graph first need to be created, and when they can be safely destroyed. To further improve the running time we carefully recoded a few subroutines that consumed the bulk of the running time, as detected by runtime profiling of the code. Though `MSA` will never be able to compete with heuristic alignment algorithms that can align dozens of sequences of triple-digit lengths, the new version is significantly more powerful than the original.

Multiple Sequence Alignment

Let $S_1, \ldots, S_K$, be the input sequences. Let Σ be the input alphabet. A multiple sequence alignment A is a rectangular character array on the alphabet $\Sigma' := \Sigma \cup \{\texttt{-}\}$ that satisfies the following conditions:

1. There are exactly K rows.

2. Ignoring dashes, row I is precisely the string S_I.

Each multiple sequence alignment induces a pairwise alignment $A_{i,j}$ on the pair of sequences S_i, S_j. This alignment is obtained by simply copying rows i and j of A, with the proviso that columns with a '-' in both rows are ignored.

```
--VLSPADKTNVKAAWGKVGAHAGEYGAEALE-
-VHLTPEEKSAVTALWGKVNVD--EVGGEALGR
--GLSDGEWQLVLNVWGKVEADIPGHGQEVLI-
MKFFAVLALCIVGAIASPLTADEASLVQSS---
```

Figure 1: An alignment of prefixes of four globin sequences

*Sandeep Gupta was advised by Alejandro Schäffer at Rice University and was supported in part by grants from the National Science Foundation and the Texas Advanced Technology Program. Address: Room 520, 545 Technology Square, Cambridge MA 02139. Web: http://www.pdos.lcs.mit.edu/~skgupta/

# Data Set	Old Time	New Time	Old Space	New Space
Globins A (7)	429 s	157 s	15.1 Mbytes	4.8 Mbytes
Globins B (10)	385 s	130 s	11.3 Mbytes	6.1 Mbytes
Proteases A (5)	79 s	37 s	5.7 Mbytes	3.0 Mbytes
Proteases B (4)	24 min	9 min	81 Mbytes	5.8 Mbytes
Kinases A (5)	35 min	10 min	67 Mbytes	6.6 Mbytes
Kinases B (6)	stopped	118 m	> 300 Mbytes	25 Mbytes
Kinases C (4)	530 s	210 s	35 Mbytes	4.7 Mbytes
RH A (4)	394 min	68 min	380 Mbytes	31 Mbytes

Table 1: Comparison of resource usage for two versions of `MSA`. Number of sequences in each data set is in parentheses.

Figure 1 shows an example multiple alignment computed by `MSA` using the default parameter settings. It has four rows, one per sequence. In this case each input sequence has 30 characters, and the output alignment has 33 columns. In general, the input sequences can be of varying lengths. [3] contains a very thorough description of the cost models and algorithms used in `MSA`.

Results

In this section, we compare the performance of `MSA` 1.0 with our improved version on real data sets. We ran all the tests on a Sun SparcStation 10 with 128Mbytes of RAM. We used the `cc` C compiler with the -O flag for optimization. To measure time we took the sum of the user and system times reported by the `time` command. To measure space usage we ran the `top` command while `MSA` was running, and noted the peak amount of space usage that it reported. We used data sets extracted from the data sets presented by McClure, Vasi and Fitch [5].

Table 1 shows the relative time and space usage of `MSA` 1.0 (old) and `MSA` 2.0 (new) on these data sets. In the case of Kinases C, we had to stop the old `MSA` run after it had run overnight and used more than 300Mbytes of memory. In the case of RH A, the old `MSA` run thrashed extensively, and therefore, the actual wall-clock time from start to finish was 44 hours (much larger than the cpu time usage reported in the table) compared to approximately 1 hour for the new version.

Conclusions

We have made improvements in the space and time usage of the `MSA` program for multiple sequence alignment. Since memory is finite on any computer, our improvements make feasible many problems that were not feasible with the first version of `MSA`. The changes to the code involved what would generally be considered low-level implementation decisions. Our work serves as yet another example of the fact that in programs that allocate a lot of memory, it pays to be careful in looking for opportunities to free memory. Also, in programs that have a tight inner loop that takes most of the time, it pays to try a variety of methods of coding that inner loop.

Obtaining the Source

The source code is currently available by anonymous ftp from `softlib.cs.rice.edu`. Contact Alex Schaffer (schaffer@cs.rice.edu) by e-mail for details. There is also a web server which will perform MSA 2.0 runs at `http://ibc.wustl.edu/msa.html`.

References

[1] H. Carrillo and D. Lipman. The multiple sequence alignment problem in biology. *SIAM Journal of Applied Mathematics*, 48:1073–1082, 1988.

[2] S. K. Gupta, J. D. Kececioglu, and A. A. Schäffer. Making the shortest-paths approach to sum-of-pairs multiple sequence alignment more space efficient in practice. *Proceedings of the 6th Conference on Combinatorial Pattern Matching*, 1995.

[3] S. K. Gupta, J. D. Kececioglu, and A. A. Schäffer. Improving the practical space and time efficiency of the shortest-paths approach to sum-of-pairs multiple sequence alignment. *Journal of Computational Biology*, to appear.

[4] D. J. Lipman, S. F. Altschul, and J. D. Kececioglu. A tool for multiple sequence alignment. *Proceedings of the National Academy of Science USA.*, 86:4412–4415, 1989.

[5] M. A. McClure, T. K. Vasi, and W. M. Fitch. Comparative analysis of multiple protein-sequence alignment methods. *Mol. Biol. Evol.*, 11:571–592, 1994.