Optimal Sum-of-Pairs Multiple Sequence Alignment using Incremental Carrillo-and-Lipman Bounds.

Arun S. Konagurthu[1,2] Peter J. Stuckey[1,3]

[1] Department of Computer Science and Software Engineering,
The University of Melbourne, Victoria, 3010, Australia.
{arun/pjs}@cs.mu.oz.au

[2] Victorian Bioinformatics Consortium
Department of Biochemistry and Molecular Biology
Monash University, Victoria, 3800, Australia.

[3] NICTA Victoria Laboratory, Australia.

Address for correspondence: Peter J. Stuckey, Department of Computer Science and Software Engineering, The University of Melbourne, Victoria, 3010, Australia.
Phone: +613-8344-1341
FAX: +613-9348-1184

Accepted for publication in Journal of Computational Biology

Abstract

Alignment of sequences is an important routine in various areas of science, notably molecular biology. Multiple sequence alignment is a computationally hard optimization problem which involves the consideration of different possible alignments in order to find an optimal one, given a measure of goodness of alignments. Dynamic programming algorithms are generally well-suited for the search of optimal alignments, but are constrained by unwieldy space requirements for large numbers of sequences. Carrillo and Lipman devised a method that helps to reduce the search space for an optimal alignment under a sum-of-pairs measure using bounds on the scores of its pairwise projections. In this paper we generalize Carrillo and Lipman bounds and demonstrate a novel approach for finding optimal sum-of-pairs multiple alignments that allows incremental pruning of the optimal alignment search space. This approach can result in a drastic pruning of the final search space polytope (where we search for the optimal alignment) when compared to Carrillo and Lipman's approach and hence allows many runs that are not feasible with the original method.

1 Introduction

Simultaneous alignment of multiple sequences is a difficult problem of great importance in computational molecular biology. Multiple alignments are used in various application areas that include molecular modelling, protein structure-function analysis, sequence fragment assembly, evolutionary phylogenetic study, database search, and primer design amongst others (Needleman and Wunsch, 1970; Murata et al., 1985; Thompson et al., 1994). With these motivations, automated multiple alignment tools have long been a topic of elaborate research.

Dynamic programming has been widely used to solve the optimization problem of aligning sequences (Needleman and Wunsch, 1970; Murata et al., 1985). However dynamic programming's asymptotic complexity increases exponentially with the dimension ($O(n^2 2^n l^n)$ for the sequence alignment problem, where l is the mean length of n sequences to be compared under a sum-of-pairs measure). The multiple sequence alignment problem using various criteria of optimality has been shown to be NP-Hard (Wang and Jiang, 1994; Just, 2001). As a result many tools and methods use approximate algorithms that trade-off optimality with the speed (Thompson et al., 1994; Notredame et al., 2000; Hughey and Krogh, 1996).

There are few tools and methods that construct an optimal alignment using sum-of-pairs cost criterion (Lipman et al., 1989; Gupta et al., 1995; Kececioglu, 1993; Stoye et al., 1997; Reinert et al., 1997, 2000; Althaus et al., 2002). Most of these tools implement the method similar to the one designed by Carrillo and Lipman that considerably restricts the size of exploration space in which the optimal solution can be searched (Carrillo and Lipman, 1988). The central idea of Carrillo and Lipman approach is that every multiple alignment imposes a pairwise alignment on any sequence pair. While treating the alignment of n sequences as a path in n-dimensional lattice, this imposed alignment on each pair can be viewed as a projected path in two-dimensional space. It is then possible to find bounds on the cost of projection of the optimal path. In practice, however, it has been observed that these bounds are over-estimated. Hence tools such as MSA (Lipman et al., 1989) implement a heuristic variant of Carrillo and Lipman's algorithm, using tighter bounds than the guaranteed ones, that does not ensure a mathematical optimum (Notredame, 2002).

Major advances in the the search for optimal alignments and the reduction of the exploration space include the following: Altschul and Lipman (1989) propose a different cost model (alignments scored as the cost of an evolutionary tree) instead of the standard sum-of-pairs cost scheme. Gupta

et al. (1995) show an efficient implementation of lattice exploration using a variant of Dijkstra's single-source shortest paths. Gusfield (1993) gives a bounded-error approximation method for sum-of-pairs sequence alignment which can be used as an alternative lower bound on the cost of an optimal alignment. Stoye et al. (1997); Stoye (1998) show a divide and conquer algorithm (`DCA`) which slices the input sequences into subsets of segments small enough to enable a massive speed-up on the regular approaches using heuristic bounds. Lermen and Reinert (2000) implement the $\mathcal{A}^*$ algorithm (goal-directed unidirectional search) that speeds up the shortest path computation by transforming the edge weights without losing the optimality of the shortest path. Reinert et al. (2000) combines the divide and conquer technique (Stoye et al., 1997) with the efficient bounding strategies in Lermen and Reinert (2000). Kececioglu (1993) introduces the maximum weight trace problem (which contains as a special case the minimum sum-of-pairs alignment problem) and proposes a branch and bound algorithm for it. Reinert et al. (1997) show a branch-and-cut algorithm for an integer linear programming (ILP) formulation of the maximum weight trace problem. Althaus et al. (2002) propose a general ILP formulation of multiple sequence alignment problem using arbitrary gap costs and describe a branch-and-cut method to find optimal alignments.

A severe constraint in the implementation of Carrillo and Lipman's approach is the space usage which is the result of exaggerated nature of the bounds (Gupta et al., 1995). The core of this paper is a novel method that successfully reduces the space usage when compared to Carrillo and Lipman's method. We formulate the sequence alignment problem as one which maximizes the score of an alignment under a sum-of-pairs measure. In our approach, the improvement in space usage is derived from the generalization of constraints on the score of projections of optimal multiple alignments of n sequences into some k-space, $k < n$. We show in this paper a novel method for constructing optimal multiple alignments using the incremental use of these generalized Carrillo and Lipman constraints. The results show a drastic pruning of the search space for optimal alignments. This space reduction that we have achieved allows the calculation of optimal alignments for data sets that were previously infeasible with the original Carrillo and Lipman method.

The outline of this paper is as follows. Section 2 describes the basic notations used in this paper and defines the problem of sequence comparison. Section 3 provides a brief review of Carrillo and Lipman's approach to constrain the search space of an optimal multiple alignment based on the scores of its pairwise projections. Section 4 generalizes Carrillo and Lipman's method for any k-space projections. Section 5 establishes the mathematical basis of space improvements of our incremental method and

describes with an example the construction of optimal alignment using our approach. Section 6 explains various materials and methods used to undertake this work. Section 7 gives the experimental results of various comparisons between our Incremental approach and Carrillo and Lipman's method on real data sets of protein sequences from HOMSTRAD and BALIBASE. We conclude this paper with a short discussion.

2 Basic Definitions and The Problem of Sequence Comparison

Suppose that the alphabet $\aleph$ is a finite set of t symbols (20 amino acid single letter codes in case of protein sequences), $\aleph = \{\alpha_1, \cdots, \alpha_t\}$.

A sequence of length k is a set of symbols of the form,

$$S = (\alpha_{n_1}, \cdots, \alpha_{n_k})$$

where for each j=1, $\cdots, k$, n_j is a natural number such that α_{n_j} is a symbol from $\aleph$.

An alignment of set of sequences $S_1, \cdots, S_n$ is another set of sequences, $S'_1, \cdots, S'_n$, such that each sequence S'_i is obtained from S_i by inserting gap symbols ('-') in positions where some of the other sequences have a non-gap symbols that satisfies the following conditions:

1. If the lengths of $S_1, \cdots, S_n$ is $k_1, \cdots, k_n$ respectively, each sequence in the set $S'_1, \cdots, S'_n$ has the same length, l, such that $\max(k_1, \cdots, k_n) \leq l \leq k_1 + \cdots + k_n$.

2. Ignoring the gap symbols, every S'_i is precisely the string S_i.

An illustration of a possible alignment of three sequences S_1 = CYRWT, S_2 = ECHYR, and S_3 = YRIW is shown in Figure 1, where the symbols of these sequences follow the single letter code convention for representing amino acid residues.

Given a scoring function $f : \aleph \times \aleph \rightarrow \Re$ and gap penalty function $\mathcal{G}$, the problem of sequence comparison is to find an optimal way to align a set of sequences such that the total measure of score of the alignment (sum of scoring function over the aligned pairs of symbols from $\aleph$, minus the sum of the penalties for gaps given by gap penalty function) is maximized. In this work we use the linear gap penalty function $\mathcal{G} = \lambda g$ in a sum-of-pairs measure where λ is the gap length and g is the per-symbol gap cost.

Any given set of n sequences $S_1, \cdots, S_n$ having lengths $k_1, \cdots, k_n$ respectively can be associated with a lattice, $\mathcal{L}(S_1, \cdots, S_n)$ in n-dimensional space (from here on, referred as n-space). This lattice

consists of $k_1 \times \cdots \times k_n$ n-space cells. Each cell corresponds to a group of n symbols, where each symbol belongs to a different sequence. The cell corresponding to the first symbol of each of the sequences is called the source and the cell corresponding to the last symbol of the sequences is called sink. Both source and sink together are referred in this paper as end corners.

The alignment of set of sequences $S_1, \cdots, S_n$ can be associated with a path $\gamma(S_1, \cdots, S_n)$ from source to sink in the lattice, $\mathcal{L}(S_1, \cdots, S_n)$. (see Figure 1)

Figure 1

For the alignment shown in Figure 1 the path encoded using the cell indices with source $\langle 0,0,0 \rangle$ and sink $\langle 5,5,4 \rangle$ is defined by the trace left by the traversal through the following edges: $\langle 0,0,0 \rangle \rightarrow \langle 0,1,0 \rangle \rightarrow \langle 1,2,0 \rangle \rightarrow \langle 1,3,0 \rangle \rightarrow \langle 2,4,1 \rangle \rightarrow \langle 3,5,2 \rangle \rightarrow \langle 3,5,3 \rangle \rightarrow \langle 4,5,4 \rangle \rightarrow \langle 5,5,4 \rangle$.

Let the measure of score of any given path γ be denoted by $\zeta(\gamma)$. There exists at least one optimal path, $\gamma^o(S_1, \cdots, S_n)$, such that the measure ζ attains the maximum value for γ^o.

Dynamic programming is popularly used to find the optimal paths (Needleman and Wunsch, 1970; Murata et al., 1985). Each cell in the dynamic programming lattice $\mathcal{L}(S_1, \cdots, S_n)$ has an associated score which indicates the best path from that cell to the source. The score to each cell is derived from its immediately preceding cells in its neighborhood. The central idea of this method is to recursively find all optimal paths to the source for all these cells in $\mathcal{L}$. The lattice can be filled from source to sink in row-major, column-major or anti-diagonal way. Each cell also holds a pointer to mark the preceding cell that contributed to its optimal path to enable a trace back of the optimal path from sink to source through these pointers. The computations in this standard dynamic programming model for calculation of γ^o takes $O(\prod_{i=1}^{n} k_i)$ steps each of which involves $O(2^n)$ operations.

The projection of a n-space path γ in the lattice $\mathcal{L}(S_1, \cdots, S_n)$ into any sub-space $\mathcal{L}'_{i_1,\cdots,i_k} = \mathcal{L}(S_{i_1}, \cdots, S_{i_k})$, for each $i_1 < \cdots < i_k \mid \{i_1, \cdots, i_k\} \subseteq \{1, \cdots, n\}$ is defined as the path associated with the imposed alignment of γ in the sub-space, $\mathcal{L}'_{i_1,\cdots,i_k}$. The projected path is denoted by $\overleftarrow{\gamma}_{i_1,\cdots,i_k}$. Figure 2 shows the projections (and the imposed alignments) of the path shown in Figure 1 into sub-spaces associated with each of its sequence pair.

Figure 2

Sum-of-pairs Alignment (SP-alignment) is a multiple alignment in which the measure of score of a path is equal to the sum of scores of all its projected pairwise paths:

$$\zeta(\gamma) = \sum_{\forall 1 \leq i < j \leq n} \zeta(\overleftarrow{\gamma}_{ij}).$$

The score of any pairwise path γ_{ij} corresponding to the some alignment of the sequences S_i and S_j is calculated as follows:

$$\zeta(\gamma_{ij}) = \sum_{k=0}^{|\gamma_{ij}|} \begin{cases} f(S'_i(k), S'_j(k)) & S'_i(k) \text{ and } S'_j(k) \in \aleph \\ g & S'_i(k) \text{ or } S'_j(k) \in \{-\} \\ 0 & S'_i(k) \text{ and } S'_j(k) \in \{-\} \end{cases}$$

where $|\gamma_{ij}|$ is the length of the path γ_{ij}, S'_i, S'_j are aligned sequences corresponding to that path with $S'_i(k)$ and $S'_j(k)$ representing the k^{th} column of the alignment.

3 Review of Carrillo and Lipman's Algorithm.

In the context of SP-alignments, Carrillo and Lipman designed a method for determining the optimal path $\gamma^o(S_1, \cdots, S_n)$ for $n > 2$ with significantly fewer computations (Carrillo and Lipman, 1988). Carrillo and Lipman's method is based on a basic observation that the score of projection of an optimal multiple alignment into any of its sequence pairs must be at most as great as the score of pairwise alignment between those two sequences (Altschul and Lipman, 1989).

Let $\gamma^h(S_1, \cdots, S_n)$ be a known heuristic path in n-space and π^o_{ij} be the optimal alignment of any pair of sequences S_i and S_j, $\forall\ 1 \leq i < j \leq n$, Carrillo and Lipman showed that,

$$\underbrace{\sum_{\forall 1 \leq i < j \leq n} \zeta(\overleftarrow{\gamma}^h_{ij}) - \sum_{\substack{\forall 1 \leq i < j \leq n, \\ (i,j) \neq (k,l)}} \zeta(\pi^o_{ij}) \leq \zeta(\overleftarrow{\gamma}^o_{kl})}_{Carrillo-Lipman\ bound.}$$

Rearranging terms we get,

$$L - U + \zeta(\pi^o_{kl}) \leq \zeta(\overleftarrow{\gamma}^o_{kl}) \tag{1}$$

where,

$L = \sum_{1 \leq i < j \leq n} \zeta(\overleftarrow{\gamma}^h_{ij})$ is the sum of the scores of all projected heuristic alignments, and
$U = \sum_{1 \leq i < j \leq n} \zeta(\pi^o_{ij})$ is the sum of all pairwise optimal alignments.

Observation 3.1. *Suppose,* $\forall\ i < j\ \mid i, j \in (1, \cdots, n)$, X_{ij} *are paths in lattice* $\mathcal{L}(S_1, \cdots, S_n)$ *whose scores of projection into any pair of sequences* S_i *and* S_j *is at least* $L - U + \zeta(\pi^o_{ij})$. *Then,* $\exists$ *a* n*-space polytope,* $\mathbf{X_{cl}} \subseteq \mathcal{L}$ *where,*

$$\mathbf{X_{cl}} = \bigcap_{\forall 1 \leq i < j \leq n} X_{ij}$$

such that only the paths in $\mathbf{X_{cl}}$ *are possible candidates to be an optimal path,* γ^o.

Observation 3.2. *Let,* $\forall\ i < j \ \mid i, j \in (1, \cdots, n)$, Y_{ij} *be set of cells in the square* $\mathcal{L}'_{ij} = \mathcal{L}(S_i, S_j)$ *whose end corners (source and sink) are traversed by some path of score at least* $L - U + \zeta(\pi^o_{ij})$. *Let* $\overrightarrow{Y}_{ij}$ *be set of points* $x \in \mathcal{L}(S_1, \cdots, S_n)$ *such that* $\overleftarrow{x}_{ij} \in Y_{ij}$, *where* $\overleftarrow{x}_{ij}$ *is the projection into* $\mathcal{L}'_{ij}$. *The set* $\overrightarrow{Y}_{ij}$ *contains all paths in* X_{ij} *and the set*

$$\mathbf{Y_{cl}} = \bigcap_{\forall 1 \leq i < j \leq n} \overrightarrow{Y}_{ij}$$

contains all paths in $\mathbf{X_{cl}}$.

To determine the region $Y_{ij} \subset \mathcal{L}'_{ij}$, we need to find the best path through each of the cells in $\mathcal{L}'_{ij}$. To enable this computation, a dynamic programming algorithm is applied in both source-to-sink and sink-to-source directions. Given both these computations, for any cell C, we now have the optimal path from C-to-source (due to source-to-sink computations), and the optimal path from C-to-sink (due to sink-to-source computations). Therefore, this enables us to compute the score of the optimal path through each of the cells in $\mathcal{L}'_{ij}$. Y_{ij} is computed in $O(\sum_{i<j}^{n} k_i k_j)$ steps where $k_1, \cdots, k_n$ are lengths of the sequences $S_1, \cdots, S_n$ (Carrillo and Lipman, 1988). As an illustration of the notion of bounded space Y_{ij}, we present an example. Consider the following three sequences: $S_1 =$ PCVCGGQ, $S_2 =$ MPRVCVCGQ, and $S_3 =$ DVCVC. Figure 3 shows the matrices associated with each sequence pair ($\{S_1, S_2\}$, $\{S_1, S_3\}$, and $\{S_2, S_3\}$). The value in each cell of the matrices represents the score of the optimal path between the end corners passing through that cell. The shaded areas (Y_{12}, Y_{13}, and Y_{12} respectively) guarantee to contain the respective projections of optimal path, $\gamma^o(S_1, S_2, S_3)$. Figure 3

Using the above reasoning Carrillo and Lipman proved that it suffices to consider only a subspace $\mathbf{Y_{cl}}$ (a n-space polytope) to restrict the search for the optimal path and hence it is unnecessary to apply the dynamic programming method on the whole lattice $\mathcal{L}(S_1, \cdots, S_n)$. The computational requirement of Carrillo and Lipman's algorithm is a function of the size of the subspace $\mathbf{Y_{cl}}$ in the lattice $\mathcal{L}(S_1, \cdots, S_n)$ plus the number of computations necessary to generate it (Carrillo and Lipman, 1988). Note that the Carrillo-Lipman bound is tightest when $\gamma^h = \gamma^o$.

The Carrillo and Lipman's algorithm for finding SP-alignments can be broadly summarized as follows:

1. Find bounds on the score of projection of optimal alignment onto each of its sequence pairs.

2. Use these constraints to restrict the size of the dynamic programming lattice.

3. Find the optimal alignment in the restricted space

4 The generalization of Carrillo and Lipman constraints on multiple alignments.

Carrillo and Lipman's approach can be generalized to find the constraints on optimal multiple alignment using bounds on the scores of its projections into any k-space ($2 \leq k < n$).

Theorem 4.1. *The search of a n-space optimal path can be constrained using the scores of its k-space projections for any $2 \leq k < n$.*

Proof. By the definition of optimality, $\zeta(\gamma^h) - \zeta(\gamma^o) \leq 0$.
Let $\overleftarrow{\gamma}_{i_1,\cdots,i_k}$ be the projection of any path, $\gamma(S_1, \cdots, S_n)$ into a k-space corresponding to the sequences $S_{i_1}, \cdots, S_{i_k}$. Then

$$\sum_{\forall 1 \leq i_1 < \cdots < i_k \leq n} \zeta(\overleftarrow{\gamma}_{i_1,\cdots,i_k}) = {}^{n-2}C_{k-2}\zeta(\gamma)$$

Hence

$$\sum_{\forall 1 \leq i_1 < \cdots < i_k \leq n} \left(\zeta(\overleftarrow{\gamma}^h_{i_1,\cdots,i_k}) - \zeta(\overleftarrow{\gamma}^o_{i_1,\cdots,i_k})\right) \leq 0$$

Let $\pi^o_{i_1,\cdots,i_k}$ denote an optimal path in a k-space lattice determined by the sequences $S_{i_1}, \cdots, S_{i_k}$, $\mathcal{L}(S_{i_1}, \cdots, S_{i_k})$. Since $\zeta(\pi^o_{i_1,\cdots,i_k}) \geq \zeta(\overleftarrow{\gamma}^o_{i_1,\cdots,i_k})$, we get

$$\sum_{cond_1} \zeta(\overleftarrow{\gamma}^h_{i_1,\cdots,i_k}) - \sum_{cond_2} \zeta(\pi^o_{i_1,\cdots,i_k}) \leq \zeta(\overleftarrow{\gamma}^o_{j_1,\cdots,j_k}).$$

where, $cond_1 \equiv \forall 1 \leq i_1 < \cdots < i_k \leq n$, and $cond_2 \equiv \forall 1 \leq i_1 < \cdots < i_k \leq n, (i_1, \cdots, i_k) \neq (j_1, \cdots, j_k)$. Rearranging terms in the above inequality we get,

$$L_k - U_k + \zeta(\pi^o_{j_1,\cdots,j_k}) \leq \zeta(\overleftarrow{\gamma}^o_{j_1,\cdots,j_k}). \tag{2}$$

where, $L_k = \sum_{\forall 1 \leq i_1 < \cdots < i_k \leq n} \zeta(\overleftarrow{\gamma}^h_{i_1,\cdots,i_k})$ and $U_k = \sum_{\forall 1 \leq i_1 < \cdots < i_k \leq n} \zeta(\pi^o_{i_1,\cdots,i_k})$. □

Observation 4.1. *Suppose,* $\forall\ i_1 < \cdots < i_k \ \mid \{i_1, \cdots, i_k\} \subseteq \{1, \cdots, n\}$, $X_{i_1,\cdots,i_k}$ *are paths in lattice* $\mathcal{L}(S_1, \cdots, S_n)$ *whose scores of projection into* k *sequences* $S_{i_1}, \cdots, S_{i_k}$ *is at least* $L_k - U_k + \zeta(\pi^o_{i_1,\cdots,i_k})$. *Then* $\exists$ *a* n*-space polytope* $\mathbf{X_{gcl}} \subseteq \mathcal{L}$, *where,*

$$\mathbf{X_{gcl}} = \bigcap_{\forall 1 \le i_1 < \cdots < i_k \le n} X_{i_1,\cdots,i_k}$$

such that only the paths in $\mathbf{X_{gcl}}$ *are possible candidates to be an optimal path,* γ^o.

Observation 4.2. *For all* $i_1 < \cdots < i_k \ \mid \{i_1, \cdots, i_k\} \subseteq \{1, \cdots, n\}$, *let* $Y_{i_1,\cdots,i_k}$ *be set of cells in the* k*-space lattice* $\mathcal{L}'_{i_1,\cdots,i_k} = \mathcal{L}(S_{i_1}, \cdots, S_{i_k})$ *whose end corners are traversed by some path of score at least* $L_k - U_k + \zeta(\pi^o_{i_1,\cdots,i_k})$. *Let* $\overrightarrow{Y}_{i_1,\cdots,i_k}$ *be set of points* $x \in \mathcal{L}(S_1, \cdots, S_n)$ *such that* $\overleftarrow{x}_{i_1,\cdots,i_k} \in Y_{i_1,\cdots,i_k}$, *where* $\overleftarrow{x}_{i_1,\cdots,i_k}$ *is the projection into* $\mathcal{L}'_{i_1,\cdots,i_k}$. *The set* $\overrightarrow{Y}_{i_1,\cdots,i_k}$ *contains all paths in* $X_{i_1,\cdots,i_k}$ *and the set*

$$\mathbf{Y_{gcl}} = \bigcap_{\forall 1 \le i_1 < \cdots < i_k \le n} \overrightarrow{Y}_{i_1,\cdots,i_k} \tag{3}$$

contains all paths in $\mathbf{X_{gcl}}$.

For any $k > 2$, the direct computation of $\pi^o_{i_1,\cdots,i_k}$ is highly expensive. In the next section we demonstrate a method in which this optimal path can be calculated in the restricted space given by earlier projections.

5 The Incremental Approach

The key idea of our approach is to incrementally compute optimal sequences $\pi^o_{i_1,\ldots,i_k}$ for each dimension k from 2 to n and apply the generalized Carrillo and Lipman pruning outlined in Observation 4.2 at each dimension. By computing only within the intersections of the previously pruned spaces this approach reduces the time and space requirements significantly.

The algorithm works as follows. We first compute $\pi^o_{i_1 i_2}$ and $Y_{i_1 i_2}$ for $1 \le i_1 < i_2 \le n$ as usual. Let $\Gamma_{i_1 i_2} = Y_{i_1 i_2}$ for $1 \le i_1 < i_2 \le n$.

Now we iterate k from 2 to n. In the k^{th} iteration. For each $1 \le j_1 < \cdots < j_{k+1} \le n$ we calculate

$$\Phi_{j_1,\ldots,j_{k+1}} = \bigcap_{cond_3} \overrightarrow{\Gamma}^{j_1,\ldots,j_{k+1}}_{i_1,\cdots,i_k}$$

where $cond_3 \equiv \forall 1 \le i_1 < \cdots < i_k \le n, \{i_1, \ldots, i_k\} \subseteq \{j_1, \ldots, j_{k+1}\}$ and $\overrightarrow{\Gamma}^{j_1,\ldots,j_{k+1}}_{i_1,\cdots,i_k}$ is the reverse projection of $\Gamma_{i_1,\cdots,i_k}$ onto the space $L(S_{j_1}, \ldots S_{j_{k+1}})$. $\Phi_{j_1,\ldots,j_{k+1}}$ is the intersection of all of the pruned

spaces for k dimensions applicable to the given $k+1$ dimensions. We then calculate $\pi^o_{j_1,\ldots,j_{k+1}}$ in this restricted space $\Phi_{j_1,\ldots,j_{k+1}}$.

Once we have calculated $\pi^o_{j_1,\ldots,j_{k+1}}$ for all $1 \leq j_1 < \cdots < j_{k+1} \leq n$ we can calculate U_{k+1} and then calculate $Y_{j_1,\ldots,j_{k+1}}$ (the application of generalized Carrillo and Lipman pruning from Observation 4.2).

For each $1 \leq j_1 < \cdots < j_{k+1} \leq n$ we calculate

$$\Gamma_{j_1,\ldots,j_{k+1}} = \Phi_{j_1,\ldots,j_{k+1}} \cap Y_{j_1,\ldots,j_{k+1}}$$

The process continues until we calculate $\pi^o_{1,\ldots,n}$. Figure 4

In the conventional Carrillo and Lipman's algorithm the multiple alignment is constrained based on the score of pairwise projections. In our approach, however, the pruning is done gradually on each increasing dimension from pairwise to triplets to quads to quints, and so on. For example, let S_1, S_2, S_3, S_4, S_5 be any five sequences whose optimal alignment is to be determined. We begin our approach by finding the bounds on projections of the optimal alignment into all of its ${}^5C_2$ pairs of sequences, $\{(S_1, S_2)$, (S_1, S_3), (S_1, S_4), (S_1, S_5), (S_2, S_3), (S_2, S_4), (S_2, S_5), (S_3, S_4), (S_3, S_5), $(S_4, S_5)\}$, using the constraint $L - U + \zeta(\pi^o_{i_1 i_2}) \leq \zeta(\overleftarrow{\gamma}^o_{i_1 i_2})$.

The original Carrillo-Lipman method proceeds from here to find a polytope of paths in 5-space constrained by the above pairwise bounds. Instead, the incremental approach explores the regions in the next dimension (3-space) such that all paths in it satisfy the constraints on their corresponding pairwise projections. We find ${}^5C_3$ such regions associated with each of the triplets: $\{(S_1, S_2, S_3)$, (S_1, S_2, S_4), (S_1, S_2, S_5), (S_1, S_3, S_4), (S_1, S_3, S_5), (S_1, S_4, S_5), (S_2, S_3, S_4), (S_2, S_3, S_5), (S_2, S_4, S_5), $(S_3, S_4, S_5)\}$

In the context of a particular three dimensional region $\mathcal{L}'_{123} = \mathcal{L}(S_1, S_2, S_3)$ we obtain the 3-space polytope $\Phi_{123} \subset \mathcal{L}'_{123}$ as the intersection of Y_{12}, Y_{13} and Y_{23} (appropriately reverse projected). This polytope, Φ_{123} will contain the projection of the optimal path $\gamma^o(S_1, S_2, S_3, S_4, S_5)$ into (S_1, S_2, S_3), $\overleftarrow{\gamma}^o_{123}$. Φ_{123} also contains paths that are possible candidates for the optimal path in (S_1, S_2, S_3), π^o_{123} (see Lemma 5.1 below).

The optimal paths of all possible triplets are explored using this method. For each $i_1 < i_2 < i_3 \mid i_1, i_2, i_3 \in \{1, 2, 3, 4, 5\}$, $\Phi_{i_1 i_2 i_3}$ is then further pruned (see Theorem 5.1 below) by eliminating any path in this space which is less than $L_3 - U_3 + \zeta(\pi^o_{i_1 i_2 i_3})$, where L_3 and U_3 are defined as in Theorem 4.1 above. This gives $\Gamma_{i_1 i_2 i_3}$. This results in the bounds on projection of optimal alignment $\gamma^o(S_1, S_2, S_3, S_4, S_5)$ on all its triplets.

Using these triplet bounds we obtain the 4-space regions associated with the quadruplets: $\{(S_1, S_2, S_3, S_4), (S_1, S_2, S_3, S_5), (S_1, S_2, S_4, S_5), (S_1, S_3, S_4, S_5), (S_2, S_3, S_4 S_5)\}$. For each $i_1 < i_2 < i_3 < i_4 \mid i_1, i_2, i_3, i_4 \in \{1, 2, 3, 4, 5\}$, we find the 4-space polytopes $\Phi_{i_1 i_2 i_3 i_4}$ in the same manner as in the case of triplets. We then explore this space and calculate the optimal quadruple paths, $\pi^o_{i_1 i_2 i_3 i_4}$. Using the scores of optimal quadruple paths we prune every $\Phi_{i_1 i_2 i_3 i_4}$ by eliminating those paths that are less than $L_4 - U_4 + \zeta(\pi^o_{i_1 i_2 i_3 i_4})$ to give $\Gamma_{i_1 i_2 i_3 i_4}$

These quadruplet spaces $\Gamma_{i_1 i_2 i_3 i_4}$ are then finally used to find the 5-space polytope, $\Phi_{12345} \subset \mathcal{L}(S_1, S_2, S_3, S_4, S_5)$ which contains all paths that are possible candidates for the optimal path, $\gamma^o(S_1, S_2, S_3, S_4, S_5)$. The exploration will lead to the calculation of optimal multiple alignment of S_1, S_2, S_3, S_4, S_5.

The correctness of the approach follows from the results below. We show that the pruning that is done at lower dimensions never removes an optimal sequence for any higher dimensions. We then show by induction that an optimal path (for any set of dimensions) is never pruned.

Lemma 5.1. *For each $2 \leq k \leq l$ and $1 \leq h_1 < \cdots < h_l \leq n$ and $1 \leq i_1 < \cdots < i_k \leq n$ where $\{i_1, \ldots, i_k\} \subseteq \{h_1, \ldots, h_l\}$ we have that the optimal path $\pi^o_{h_1,\ldots,h_l}$ is in the reverse projection of $Y_{i_1,\ldots,i_k}$ to the space $\mathcal{L}(S_{h_1}, \ldots, S_{h_l})$.*

Proof. We make use of the generalized Carrillo and Lipman bounds pruning, assuming that $h_1, \ldots, h_l$ are the only dimensions in the problem, and show that the pruning with more dimensions is weaker. Define $L_k^{h_1,\ldots,h_l}$ and $U_k^{h_1,\ldots,h_l}$ as follows.

$$L_k^{h_1,\ldots,h_l} = \sum_{\forall 1 \leq i_1 < \cdots < i_k \leq n, \{i_1,\ldots,i_k\} \subseteq \{h_1,\ldots,h_l\}} \zeta(\overleftarrow{\gamma}^h_{i_1,\ldots,i_k})$$

$$U_k^{h_1,\ldots,h_l} = \sum_{\forall 1 \leq i_1 < \cdots < i_k \leq n, \{i_1,\ldots,i_k\} \subseteq \{h_1,\ldots,h_l\}} \zeta(\pi^o_{i_1,\ldots,i_k})$$

Now $Y_{i_1,\ldots,i_k}$ is the space that contains only paths at least of score $L_k - U_k + \zeta(\pi^o_{i_1,\ldots,i_k})$.

If the only dimensions in the original problem were $h_1, \ldots, h_l$ then we could apply generalized Carrillo and Lipman pruning (Observation 4.2) to create a space $Y'_{i_1,\ldots,i_k}$ with paths at least of score $L_k^{h_1,\ldots,h_l} - U_k^{h_1,\ldots,h_l} + \zeta(\pi^o_{i_1,\ldots,i_k})$ and be guaranteed that $\pi^o_{h_1,\ldots,h_l}$ appeared in the reverse projection of this space. We show that $Y_{i_1,\ldots,i_k} \supseteq Y'_{i_1,\ldots,i_k}$ since $L_k - U_k \leq L_k^{h_1,\ldots,h_l} - U_k^{h_1,\ldots,h_l}$.

By definition

$$L_k = L_k^{h_1,\ldots,h_l} + \sum_{\forall 1 \leq i_1 < \cdots < i_k \leq n, \{i_1,\ldots,i_k\} \not\subseteq \{h_1,\ldots,h_l\}} \zeta(\overleftarrow{\gamma}^h_{i_1,\ldots,i_k})$$

and

$$U_k = U_k^{h_1,\ldots,h_l} + \sum_{\forall 1 \le i_1 < \cdots < i_k \le n, \{i_1,\ldots,i_k\} \not\subseteq \{h_1,\ldots,h_l\}} \zeta(\pi^o_{i_1,\ldots,i_k})$$

Hence we need to prove that

$$\sum_{\forall 1 \le i_1 < \cdots < i_k \le n, \{i_1,\ldots,i_k\} \not\subseteq \{h_1,\ldots,h_l\}} \zeta(\overleftarrow{\gamma}^h_{i_1,\ldots,i_k}) - \zeta(\pi^o_{i_1,\ldots,i_k}) < 0$$

But by the definition of optimality of $\pi^o_{i_1,\ldots,i_k}$ $\zeta(\pi^o_{i_1,\ldots,i_k}) > \zeta(\overleftarrow{\gamma}^h_{i_1,\ldots,i_k})$ and hence the result holds. □

Theorem 5.1. *For each $2 \le k \le l$ and $1 \le h_1 < \cdots < h_l \le n$ and $1 \le i_1 < \cdots < i_k \le n$ where $\{i_1,\ldots,i_k\} \subseteq \{h_1,\ldots,h_l\}$ we have that the optimal path $\pi^o_{h_1,\ldots,h_l}$ is in the reverse projection of $\Gamma_{i_1,\ldots,i_k}$ to the space $\mathcal{L}(S_{h_1},\ldots,S_{h_l})$.*

Proof. The proof is by induction on k. The base case when $k = 2$ follows directly from Lemma 5.1 since $\Gamma_{i_1 i_2} = Y_{i_1 i_2}$.

Let us consider the case when $k = k' + 1 > 2$. By induction, for each $1 \le j_1 < \cdots < j'_k \le n, \{j_1,\ldots j_{k'}\} \subseteq \{i_1,\ldots,i_k\}$ we have that $\pi^o_{h_1,\ldots,h_l}$ appears in each space $\overrightarrow{\Gamma}^{h_1,\ldots,h_l}_{j_1,\cdots,j_{k'}}$, and hence clearly it also appears in $\overrightarrow{\Phi}^{h_1,\ldots,h_l}_{i_1,\cdots,i_k}$ by definition.

Now by Lemma 5.1 $\pi^o_{h_1,\ldots,h_l}$ also appears in $\overrightarrow{Y}^{h_1,\ldots,h_l}_{i_1,\cdots,i_k}$ and hence it appears in $\overrightarrow{\Gamma}^{h_1,\ldots,h_l}_{i_1,\cdots,i_k}$ by definition.

□

By simply applying the above theorem when $l = n$ we have that.

Corollary 5.1. *The incremental Carrillo and Lipman approach correctly calculates $\pi^o_{1,\ldots,n} \equiv \gamma^o_{1,\ldots,n}$.*

6 Materials and Methods

All the programs developed for this work were implemented using standard C. Separate programs were developed to implement both Carrillo and Lipman's approach and the Incremental approach. To construct fast heuristic alignments a program was developed that implements the progressive pairwise approach for constructing multiple sequence alignments using Unweighted Pair Group using Arithmetic Mean(UPGMA) clustering (Sneath and Sokal, 1973) to build a guide tree along which the final heuristic alignment is forced. While developing the programs we ensured the use of similar data structures and programming logic so that comparisons between Carrillo and Lipman's method

and the Incremental method have merit. The source code of our implemention can be obtained from `http://www.cs.mu.oz.au/~arun/msa-incr.html`.

Real sequence data sets were used to compare the performance of both the methods. The short amino acid sequences were extracted from HOMSTRAD (Mizuguchi et al., 1998), the database of protein structure alignments for homologous families. Also, the entire reference 1 set from BAL-IBASE (Thompson et al., 1999) is used in this work.

The Blosum62 (Henikoff and Henikoff, 1992) substitution scoring matrix and a gap penalty of -5 are used as alignment parameters for results in Tables 1-4. To show the variability of space and time usages of both the approaches as a function of alignment parameters, an in-house substitution matrix (created from 400 families of HOMSTRAD structural alignments) with a gap penalty of -9 was used to align the same data sets used in Tables 1-3 of this paper. Details of the substitution matrix and its synthesis can be found at `www.cs.mu.oz.au/~arun/INCR_APPROACH/submat.pdf`

The programs were executed on an INTEL Pentium 4 PC with a 1.3GHz processor and 256 Mbytes of primary memory running on Redhat's Fedora Core 1 Linux operating system.

7 Experimental Results

In this section we undertake various comparisons of the performance of Carrillo and Lipman's Approach with our Incremental Approach. The comparisons between these approaches were made using the follows metrics:

- Peak Space Usage (PSU): In both these approaches the maximum space requirement is a linear function of the number of cells that are needed to be held in the programs data structures to enable an exploration. Also, the data structures used in both these programs are linearly related to the dimension of an exploration. Therefore, for the Carrillo and Lipman's approach Peak Space Usage is calculated as:

$$\text{Peak Space Usage} = |\mathbf{Y_{cl}}| \times n$$

  In the Incremental approach the explorations are done repeatedly within every level of the increment. Hence the Peak Space Usage is measured as:

$$\text{Peak Space Usage} = max_{J \subseteq \{1,\ldots,n\}} |\Phi_J| \times |J|$$

that is the maximum size of Φ for some index set J multiplied by the dimension $|J|$ of that index set.

- Total Operations (TOps): The execution time is dominated by the calculation of optimal paths C-to-sink and C-to-source to enable pruning. We use this to define the "total operations" measure of computational complexity.

  For Carrillo and Lipman's approach the estimate of total operations is measured as:

  $$\text{Total Operations} = |\mathbf{Y_{cl}}| \times (2^n - 1)$$

  For the Incremental approach this is measured as:

  $$\text{Total Operations} = \sum_{J \subseteq \{1,\ldots,n\}} |\Phi_J| \times (2^{|J|} - 1)$$

  Note that neither measures the operations for the original Carrillo and Lipman pruning.

- Time: This metric measures the wall clock time of executions of both the methods in seconds.

- Accuracy(Acc): This measure gives the relatedness of the alignments with respect to the database alignment. The overall accuracy of a multiple alignment is calculated as the mean of accuracy of all possible non-redundant sequence pairs in a multiple alignment. The pairwise accuracy is the percentage of correctly aligned residues with respect to the a database alignment. Note that the two methods always calculate the same accuracy. They are actually not guaranteed to give the same alignment since there may be more than one optimal, but this never occurs in our experiments.

Table 1 shows the comparisons of Peak Space Usage, Total Operations, Time, and Accuracy metrics between Carrillo and Lipman's approach and Incremental approach for data sets from HOMSTRAD. The data sets follow the HOMSTRAD nomenclature. We also show the number of sequences in each data set in parentheses. Note that same alignment parameters (Blosum62 substitution matrix with gap penalty of -5) and heuristic alignment were used for all experiments. Results in Table 1 clearly

show a approximately 4× reduction in the Peak Space Usage using the Incremental approach, and an approximately 2× reduction in the total operations and total execution time.

Both the original Carrillo and Lipman method and our incremental approach are improved if a better heuristic is used. In the next experiment we use the optimal answer as the heuristic input to the algorithms. The results are shown in Table 2. In these experiments sometimes Carrillo and Lipman's approach ran out of memory in the construction of $\mathbf{Y_{cl}}$ and hence we cannot get an accurate gauge of Peak Space Usage and Total Operations, so all the entries are marked as —. From the results it is clear that the incremental approach gains even more benefit from a better heuristic, with a order of magnitude reduction in space requirements, and approximately 3× reduction in operations and approximately 2× reduction in execution time. Here the improvement in execution time is less than that for total operations as input/output becomes a significant proportion of the execution time. We shall see in the next experiment how we can generate a very good heuristic answer rapidly for use by our incremental approach.

In practice, to be able to achieve simultaneous alignment of many sequences Carrillo and Lipman's approach is used with bounds tighter than the guaranteed ones (Lipman et al., 1989; Gupta et al., 1995). We can apply the same idea to the incremental approach, using bounds tighter than the guaranteed ones at each step. In the following experiment we used both methods with a bound $L_k - U_k$ defined as $-50 - 10 \times (k-2)$ rather than using the calculated values. This bound is generous in the sense that the value calculated is optimal for all data sets where we know the optimal value (those in Table 2). However it should be noted that the choice of such a definition for the heuristic bound for very divergent sequences can result in the method missing the optimal alignment. Pre-calculated heuristic bounds based on sequence divergence, composition, dimension of alignment could be used to serve as a good definition. Table 3 shows the comparisons between the heuristic implementations of both the approaches. In this experiment sometimes Carrillo and Lipman's approach could construct $\mathbf{Y_{cl}}$ but ran out of memory for the calculation of the optimal alignment, in which case the Time and Acc extries are shown as —. The heuristic incremental approach clearly allowed simultaneous alignment of datasets with many sequences while the heuristic-Carrillo and Lipman's approach failed on most larger datasets. The incremental method requires an order of magnitude less memory and approximately 6× less operations and one quarter of the time.

Although the heuristic incremental approach is not guaranteed to find the optimal solution, we are unaware of any cases where it failed to find the optimal. We can combine the heuristic incremental

approach with the complete incremental approach by using the answer from the heuristic incremental approach as the heuristic input to the complete incremental approach. For example we can prove the optimal answer to bowman using 16 seconds to generate the optimal using the heuristic incremental approach, plus 96 seconds to prove its optimality using the complete incremental approach.

We also ran the heuristic versions of both Carrillo and Lipman as well as the Incremental approach on BALIBASE reference 1 benchmarks consisting of 81 datasets that are known to contain alignments of divergent sequences. Table 4 shows these results. The gain using the Incremental approach is clearly apparent from these results, more than an order of magnitude improvement in space requirements, and significant gains in execution time.

The space and time usages of both the approaches is largely dependent on the scoring function. To demonstrate this variability of space and time usages we undertook the comparisons between the approaches using an in-house scoring matrix. The results of the comparisons are available at:
`http://www.cs.mu.oz.au/~arun/INCR_APPROACH/tables_5_6_7.pdf`
Changing the scoring function resulted in the runs of many data sets to fail using Carrillo and Lipman's method while Incremental approach performed better across all the comparisons.

8 Discussion

In this paper we demonstrate an approach for calculating optimal SP-alignments using linear gap penalties, where for any pair of sequences a fixed penalty is applied whenever a gap symbol in one sequence is aligned with a non-gap symbol in another. In general, affine gap penalties have been shown to be more accurate than linear gap penalties (Altschul and Erickson, 1986). In the affine gap scheme, for any pair of sequences, gaps are variably penalized proportional to the length of continuous runs of gap symbols in one sequence aligned to non-gap symbols in another. For any such continuous run of gap symbols, the penalty is of the form $\mathcal{G}: \ g_o + \lambda g$, where λ is the gap length, g_o is the penalty for initiating a gap and g is the regular per-symbol penalty. However, implementing a dynamic programming algorithm using affine gap penalties is more memory intensive than that of linear gaps. Due to the non-additive nature of affine gap alignments and the impracticality of implementing them using a single dynamic programming lattice (where at every cell in the lattice all possible gap-lengths should be exhaustively tried), a common practice is to use multiple "help-lattices," each help-lattice corresponds to a particular type of alignment column and has its corresponding update rules (Gusfield,

1997). The help-lattices can then be used additively to determine the optimal alignment using affine gaps. See Gusfield (1997) for details.

In theory, both Carrillo and Lipman's method as well as the incremental algorithm discussed in this paper can be extended to affine gap penalties in a sum-of-pairs measure. However, in practice there are few problems in this extension. First, the number of help-lattices grow exponentially with the number of sequences as $O(2^n)$ and hence the demand of space quickly becomes unacceptable. Even if this is overlooked, under a sum-of-pairs scoring function, it is not possible to determine whether or not a column of alignment in its pairwise (and hence subsequent) projections initiates a gap without probing the information of arbitrary number of previous columns. The later problem is common also to the "natural" gap penalties discussed in Altschul (1989) where, for a pairwise alignment, the natural gap penalty is calculated by charging a constant penalty for every continuous run of gap symbols in one sequence aligned to non-gap symbols in another. Altschul (1989) overcomes the problem of natural gap penalties by compromising with a slightly altered definition which he calls "quasi-natural" gap penalties where only the preceding column completely determines the number of gaps the current column of alignment introduces. The implementation of `MSA` (Gupta et al., 1995) with space-time improvements and accommodating affine gap penalties also compromises in the same way by relying on the previous column to completely determine the gap initiation structure of any given column of alignment. Let us call this "quasi-affine"gap penalties. Following the proof sketch of Theorem 4.1 we can show that

$$L_k - U_k + \zeta_q(\pi^o_{j_1,\cdots,j_k}) \leq \zeta_q(\overleftarrow{\gamma}^o_{j_1,\cdots,j_k}). \tag{4}$$

where, ζ_q is the scoring function using quasi-affine gap penalties in a sum-of-pairs measure, $L_k = \sum_{\forall 1 \leq i_1 < \cdots < i_k \leq n} \zeta_q(\overleftarrow{\gamma}^h_{i_1,\cdots,i_k})$ and $U_k = \sum_{\forall 1 \leq i_1 < \cdots < i_k \leq n} \zeta_q(\pi^o_{i_1,\cdots,i_k})$.

Hence it is possible to prove optimality using quasi-affine gap penalties for both Carrillo and Lipman's method as well the incremental approach.

There are a number of subtleties and problems in actually extending the implementation to use quasi-affine penalties. First, the projection used in such an approach cannot remove all gap columns since this changes the quasi-affine penalty function and the theoretical results will fail to hold. For example the three pairwise projections of the alignment shown in Figure 1 would be

```
- C - Y R - W T     - C - Y R - W T     E C H Y R - - -
E C H Y R - - -     - - - Y R I W -     - - - Y R I W -
```

as opposed to those shown in Figure 2. With linear (and affine and natural) gap penalties the gap-gap alignments have no score, this is no longer the case with quasi-affine (or quasi-natural) gap penalties. Second, the calculation of the shadows Y_{ij} (and their multidimensional counterparts $\Gamma_{i_1 \ldots i_k}$) is more complicated since the optimal path through each cell C is not simply determined by summing the optimal C-to-source path with an optimal C-to-sink path. Third, the calculation of optimal solutions using quasi-affine penalties is expensive so in practice an approximation is used. If we do not require provably optimal alignments, then we can ignore the above problems. In practice we would expect that the overly generous nature of the generalized Carrillo and Lipman bound would still lead to optimal alignments being found.

9 Conclusions

We present an approach for calculating optimal sum-of-pairs multiple alignments using incremental Carrillo-Lipman bounds. Our experimental results demonstrate a drastic reduction in the exploration space for optimal alignments compared to the conventional approach. This improvement allows many runs that were unsuccessful using the original method. The incremental method can also be used heuristically by tightening the bounds artificially on every increment. This heuristic method is faster and more space efficient than using an heuristic version of Carrillo and Lipman bounds, an approach used for example in tools such as `MSA` (Lipman et al., 1989).

The architecture of the incremental approach lends itself to straightforward parallelization over symmetric multiprocessors using shared memory. At every different level, the nodes shown in Figure 4 are independent and hence can be calculated in parallel. This could further substantially reduce the total time of execution. We have also discussed the extension of gap scoring scheme from linear to quasi-affine. Future directions for investigation include the use of a tree model for scoring multiple alignments, use of sequence weighting and enabling quasi-affine gap penalties to increase the accuracy of this approach while combining it with divide-and-conquer technique to enable the simultaneous alignments of large protein sequences in reasonably fast time.

10 Acknowledgements

A.S.K. thanks Jun Liu for his suggestions that aided debugging.

References

Althaus, E., Caprara, A., Lenhof, H.-P., and Reinert, K., 2002. Multiple sequence alignment with arbitrary gap costs: Computing an optimal solution using polyhedral combinatorics. Bioinformatics 18, S4–S16. Suppl. S.

Altschul, S. F., 1989. Gap costs for multiple sequence alignment. J. Theor. Biol. 138, 297–309.

Altschul, S. F. and Erickson, B. W., 1986. Optimal sequence alignment using affine gap costs. Bull. Math. Biol. 48, 603–616.

Altschul, S. F. and Lipman, D. J., 1989. Trees, stars, and multiple biological sequence alignment. S.I.A.M. J. Appl. Math. 49, 197–209.

Carrillo, H. and Lipman, D., 1988. The multiple sequence alignment problem in biology. S.I.A.M. J. Appl. Math. 48(5), 1073–1082.

Gupta, S. K., Kececioglu, J. D., and Schaffer, A., 1995. Improving the practical space and time efficiency of the shortest-paths approach to sum-of-pairs multiple sequence alignments. J. Comput. Biol. 2(3), 459–472.

Gusfield, D., 1993. Efficient methods for multiple sequence alignment with guaranteed error bounds. Bull. Math. Biol. 55(1), 141–154.

Gusfield, D., 1997. Algorithms on strings, trees, and sequences: computer science and computational biology, chapter 11. Cambridge University Press.

Henikoff, S. and Henikoff, J. G., 1992. Amino acid substitution matrices from protein blocks. Proc. Natl. Acad. Sci. U.S.A. 89(22), 10915–10919.

Hughey, R. and Krogh, A., 1996. Hidden markov madels for sequence analysis: extension and analysis of basic method. Comput. Appl. Biosci. 12, 95–107.

Just, W., 2001. Computational complexity of multiple sequence alignment with SP-score. J. Comput. Biol. 8(6), 615–623.

Kececioglu, J. D., 1993. The maximum weight trace problem in multiple sequence alignment. In CPM '93: Proceedings of the 4th Annual Symposium on Combinatorial Pattern Matching, 106–119. Springer-Verlag, London, UK.

Lermen, M. and Reinert, K., 2000. The practical use of the $\mathcal{A}^*$ algorithm for exact multiple sequence alignment. J. Comput. Biol. 7, 655–671.

Lipman, D. J., Altschul, S. F., and Kececioglu, J. D., 1989. A tool for multiple sequence alignment. Proc. Natl. Acad. Sci. U.S.A. 86, 4412–4415.

Mizuguchi, K., Deane, C. M., Blundell, T. L., and Overington, J. P., 1998. HOMSTRAD: A database of protein structure alignments for homologous families. Prot. Sci. 7, 2469–2471.

Murata, M., Richardson, J., and Sussman, J., 1985. Simultaneous comparison of three protein sequences. Proc. Natl. Acad. Sci. U.S.A. 82, 3073–3077.

Needleman, S. B. and Wunsch, C. D., 1970. A general method applicable to the search for similarities in amino acid sequence of two proteins. J. Mol. Biol. 48, 443–453.

Notredame, C., 2002. Recent progress in multiple sequence alignments: A survey. Pharmacogenomics 3, 1–14.

Notredame, C., Higgins, D., and Heringa, J., 2000. T-Coffee: A novel method for multiple sequence alignments. J. Mol. Biol. 302, 205–217.

Reinert, K., Lenhof, H.-P., Mutzel, P., Mehlhorn, K., and Kececioglu, J., 1997. A branch-and-cut algorithm for multiple sequence alignment. In Proceedings of the First Annual International Conference on Computational Molecular Biology (RECOMB-97), 241–249.

Reinert, K., Stoye, J., and Will, T., 2000. An iterative method for faster sum-of-pairs multiple sequence alignment. Bioinformatics 16, 808–814.

Sneath, P. H. A. and Sokal, R. R., 1973. Numerical Taxonomy. W. H. Freeman and Company, San Fransisco.

Stoye, J., 1998. Multiple sequence alignment with the divide-and-conquer method. Gene 211, GC45–GC56.

Stoye, J., Moulton, V., and Dress, A. W., 1997. DCA: an efficient implementation of divide and conquer approach to simultaneous multiple sequence alignment. Comput. Appl. Biosci. 13(6), 625–626.

Thompson, J. D., Higgins, D. G., and Gibson, T. J., 1994. CLUSTAL W: improving the sensitivity of progressive multiple alignment through sequence weighting, position-specific gap penalties and weight matrix choice. Nucleic Acids Res. 22, 4673–4680.

Thompson, J. D., Plewniak, F., and Poch, O., 1999. BALIBASE: A benchmark alignment database for the evaluation of multiple sequence alignment programs. Bioinformatics 15, 87–88.

Wang, L. and Jiang, T., 1994. On the complexity of multiple sequence alignment. J. Comput. Biol. 1, 337–348.

List of Figures

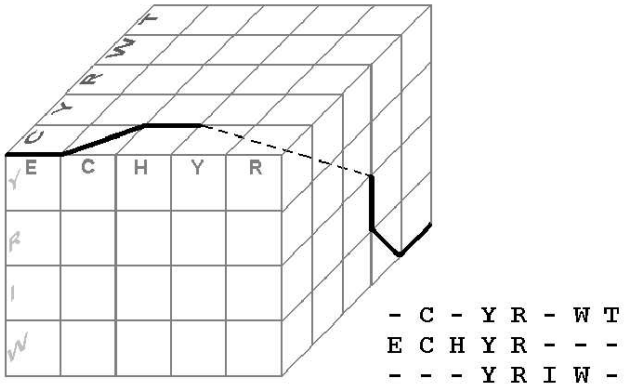

Figure 1: A path in three dimensional space corresponding to an alignment of three sequences.

⇑

A. S. Konagurthu

Figure 1 (of 4)

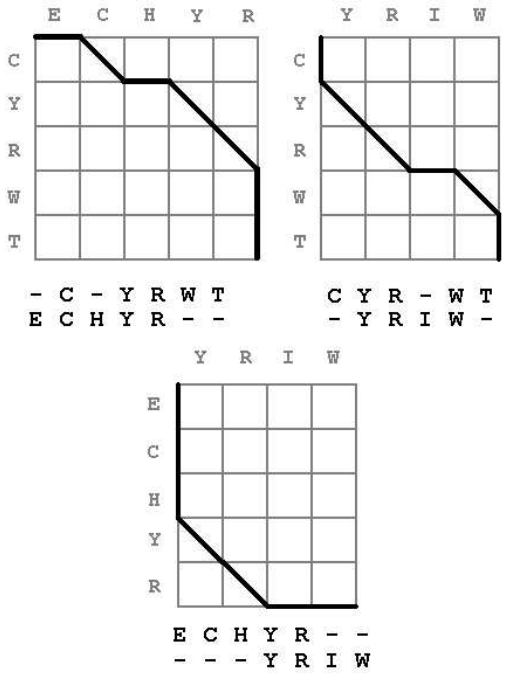

Figure 2: The projection of the path shown in Figure 1 into planes associated with its sequence pairs.

⇑

A. S. Konagurthu

Figure 2 (of 4)

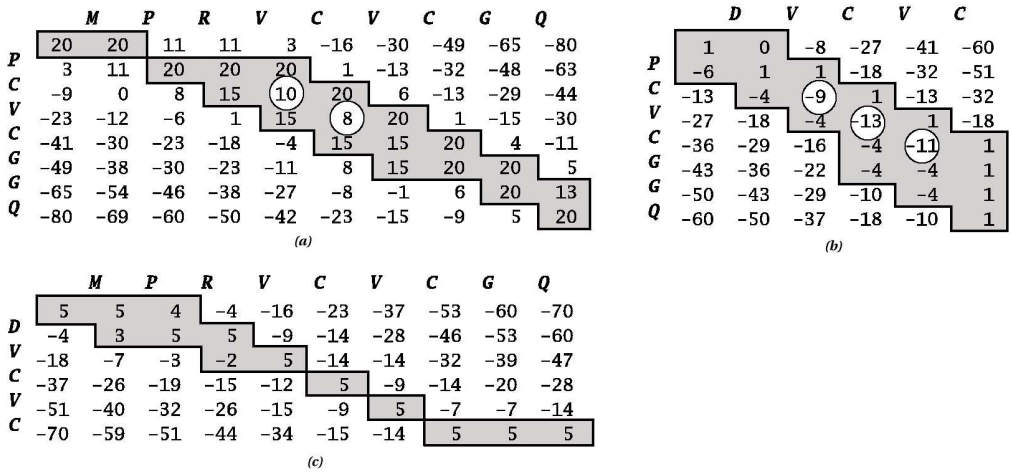

Figure 3: An illustration of constraints on multiple alignment using bounds on its pairwise projections. The matrices correspond to the each pair of the sequences: $S_1 = \text{PCVCGGQ}$, $S_2 = \text{MPRVCVCGQ}$, and $S_3 = \text{DVCVC}$; (a) is S_1 versus S_2, (b) is S_1 versus S_3 and (c) is S_2 versus S_3. The score in each cell of the matrices denotes the optimal path between the end corners through that cell. For this example, $L - U$ was computed to be -7. The shaded regions denote Y_{12}, Y_{13}, and Y_{23} respectively.

⇑

A. S. Konagurthu

Figure 3 (of 4)

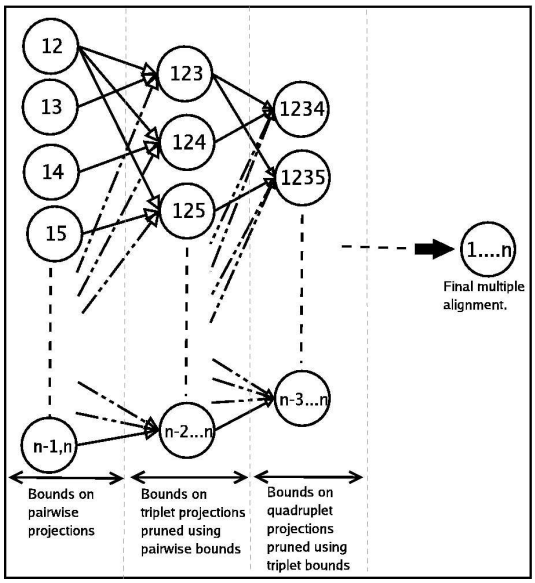

Figure 4: The basic architecture of the Incremental Approach.

⇑

A. S. Konagurthu

Figure 4 (of 4)

List of Tables

Table 1: Comparisons between Carrillo and Lipman's approach and Incremental Approach over HOMSTRAD datasets. PSU, $TOps$, $Time$, Acc represents Peak Space Usage, Total Operations, Time, and Accuracy respectively. The definitions of these metrics can be found in Section 7. (For the last row, the entries corresponding to the PSU, $TOps$, and $Time$ columns indicates their respective geometric means while the entry corresponding to Acc indicates its arithmetic mean.)

DATA SET	CARRILLO AND LIPMAN				INCREMENTAL			
	PSU	TOps	Time	Acc	PSU	TOps.	Time	Acc
bowman (5)	2.546e+07	1.578e+08	1704 s	76.0%	3.323e+06	4.563e+07	420 s	76.0%
CBS (4)	6.258e+06	2.347e+07	199 s	44.8%	3.635e+06	1.518e+07	82 s	44.8%
ccH (4)	1.449e+05	5.434e+05	4 s	94.8%	4.512e+04	2.951e+05	2 s	94.8%
ChtBD (5)	6.910e+03	4.284e+04	1 s	95.8%	6.600e+02	2.637e+04	2 s	95.8%
cytb (4)	2.785e+05	1.044e+06	9 s	80.6%	5.212e+04	4.259e+05	3 s	80.6%
dhfr (4)	2.001e+07	7.505e+07	606 s	81.2%	8.570e+06	3.755e+07	220 s	81.2%
GLA (4)	9.753e+04	3.657e+05	3 s	97.1%	2.446e+04	1.900e+05	2 s	97.1%
Glyco_hydro_18_D2 (4)	1.643e+07	6.160e+07	544 s	64.9%	7.277e+06	3.063e+07	183 s	64.9%
hpr (5)	6.227e+05	3.861e+06	26 s	97.4%	8.253e+04	2.152e+06	21 s	97.4%
hr (5)	1.821e+06	1.129e+07	76 s	97.0%	1.664e+05	4.232e+06	43 s	97.0%
HTH_AraC (4)	4.095e+06	1.536e+07	132 s	29.3%	2.533e+06	1.080e+07	62 s	29.3%
LDLa (4)	4.545e+04	1.704e+05	1 s	89.9%	1.830e+04	1.169e+05	<1 s	89.9%
LIM (5)	5.453e+06	3.381e+07	220 s	90.4%	4.427e+05	1.122e+07	110 s	90.4%
myb_DNA-binding (5)	2.280e+06	1.414e+07	98 s	93.6%	1.949e+05	4.621e+06	46 s	93.6%
parv (7)	6.898e+06	1.252e+08	860 s	97.9%	2.643e+05	4.760e+07	696 s	97.9%
plantltp (5)	3.208e+05	1.989e+06	12 s	79.6%	2.302e+04	6.228e+05	7 s	79.6%
Propep_M14 (4)	3.246e+04	1.217e+05	1 s	93.5%	7.041e+03	8.038e+04	1 s	93.5%
protg (4)	7.756e+05	2.909e+06	23 s	48.2%	2.183e+05	1.292e+06	7 s	48.2%
rep (4)	1.100e+06	4.126e+06	35 s	79.2%	1.098e+05	9.985e+05	7 s	79.2%
rub (5)	1.688e+04	1.046e+05	1 s	93.5%	4.128e+03	1.136e+05	2 s	93.5%

...Continued on next page

Table 1: ...continued

DATA SET	CARRILLO AND LIPMAN				INCREMENTAL			
	PSU	TOps	Time	Acc	PSU	TOps.	Time	Acc
seatoxin (5)	1.378e+07	8.543e+07	743 s	67.5%	8.993e+05	2.026e+07	191 s	67.5%
squash (4)	8.488e+03	3.183e+04	1 s	91.1%	2.004e+03	2.422e+04	<1 s	91.1%
tbpc (4)	1.436e+06	5.386e+06	45 s	94.0%	6.347e+05	3.285e+06	24 s	94.0%
tgfb (5)	4.910e+06	3.044e+07	191 s	89.6%	4.444e+05	1.174e+07	125 s	89.6%
TIG (6)	3.658e+07	3.841e+08	4222 s	97.1%	1.951e+06	8.927e+07	1426 s	97.1%
TIL (4)	2.071e+06	7.765e+06	60 s	75.1%	4.066e+05	2.366e+06	15 s	75.1%
WW (4)	2.132e+06	7.997e+06	65 s	61.6%	1.254e+06	5.390e+06	29 s	61.6%
Mean	**9.757e+05**	**4.855e+06**	**41.8**	**81.5%**	**1.744e+05**	**2.220e+06**	**21.9**	**81.5%**

Table 2: Comparisons between Carrillo and Lipman's approach and Incremental Approach both using the optimal as heuristic over HOMSTRAD data sets. PSU, $TOps$, $Time$, Acc represents Peak Space Usage, Total Operations, Time, and Accuracy respectively. The definitions of these metrics can be found in Section 7.(For the last row, the entries corresponding to the PSU, $TOps$, $Time$ columns indicates their respective geometric means while the entry corresponding to Acc indicates its arithmetic mean. For judicious comparisons of these central trends for each column across both the methods, the missing entries (—) in a column corresponding to Carrillo and Lipman Approach and the equivalent entries from a column in Incremental Approach are both excluded from the calculations.)

DATA SET	CARRILLO AND LIPMAN				INCREMENTAL			
	PSU	TOps	Time	Acc	PSU	TOps.	Time	Acc
bowman (5)	9.314e+06	5.775e+07	216 s	76.0%	6.410e+05	1.028e+07	96 s	76.0%
CBS (4)	1.662e+06	6.231e+06	16 s	44.8%	2.056e+05	1.423e+06	8 s	44.8%
ccH (4)	3.098e+04	1.162e+05	2 s	94.8%	5.226e+03	6.119e+04	1 s	94.8%
ChtBD (5)	6.645e+03	4.120e+04	1 s	95.8%	5.940e+02	2.089e+04	<1 s	95.8%
cyt5 (6)	4.377e+07	4.596e+08	3199 s	72.9%	2.936e+05	1.736e+07	181 s	72.9%
cytb (4)	1.400e+05	5.251e+05	1 s	80.6%	1.928e+04	1.841e+05	1 s	80.6%
dhfr (4)	1.024e+06	3.842e+06	12 s	81.2%	8.793e+04	1.121e+06	10 s	81.2%
GLA (4)	2.608e+04	9.780e+04	1 s	97.1%	5.994e+03	4.061e+04	1 s	97.1%
Glyco_hydro_18_D2 (4)	2.574e+06	9.651e+06	25 s	64.9%	1.361e+05	1.385e+06	9 s	64.9%
hpr (5)	1.722e+04	1.068e+05	1 s	97.4%	1.722e+03	3.894e+04	1 s	97.4%
hr (5)	2.560e+05	1.587e+06	5 s	97.0%	1.416e+04	4.242e+05	5 s	97.0%
HTH_AraC (4)	4.265e+05	1.599e+06	5 s	29.3%	3.832e+04	4.716e+05	3 s	29.3%
kazal (6)	1.188e+07	1.247e+08	691 s	88.3%	1.089e+05	6.258e+06	62 s	88.3%
LDLa (4)	3.996e+03	1.498e+04	1 s	89.9%	9.120e+02	9.374e+03	<1 s	89.9%
LIM (5)	7.697e+05	4.772e+06	14 s	90.4%	6.217e+04	1.365e+06	13 s	90.4%
myb_DNA-binding (5)	2.330e+05	1.445e+06	4 s	93.6%	7.995e+03	3.023e+05	3 s	93.6%
parv (7)	1.907e+06	3.459e+07	133 s	97.9%	4.564e+04	8.317e+06	93 s	97.9%

...Continued on next page

Table 2: ...continued

DATA SET	CARRILLO AND LIPMAN				INCREMENTAL			
	PSU	TOps	Time	Acc	PSU	TOps.	Time	Acc
plantltp (5)	9.758e+04	6.050e+05	2 s	79.6%	6.477e+03	1.421e+05	2 s	79.6%
Propep_M14 (4)	8.912e+03	3.342e+04	1 s	93.5%	2.055e+03	1.619e+04	<1 s	93.5%
protg (4)	1.817e+05	6.815e+05	2 s	48.2%	2.437e+04	2.166e+05	2 s	48.2%
rep (4)	7.486e+05	2.807e+06	7 s	79.2%	6.365e+04	6.442e+05	4 s	79.2%
rnasemam (6)	—	—	—	—	6.023e+06	3.582e+08	5230 s	87.4%
rub (5)	1.076e+04	6.674e+04	1 s	93.5%	8.460e+02	2.467e+04	<1 s	93.5%
seatoxin (5)	2.328e+06	1.444e+07	48 s	67.5%	1.053e+05	2.054e+06	18 s	67.5%
squash (4)	2.320e+03	8.700e+03	1 s	91.1%	6.240e+02	5.125e+03	<1 s	91.1%
tbpc (4)	3.086e+04	1.157e+05	2 s	94.0%	8.190e+03	6.797e+04	2 s	94.0%
tgfb (5)	2.994e+05	1.856e+06	6 s	89.6%	2.134e+04	5.749e+05	4 s	89.6%
TIG (6)	9.682e+06	1.017e+08	457 s	97.1%	1.602e+05	1.426e+07	159 s	97.1%
TIL (4)	3.667e+05	1.375e+06	4 s	75.1%	3.628e+04	2.812e+05	2 s	75.1%
WW (4)	2.125e+05	7.969e+05	2 s	61.6%	2.062e+04	2.326e+05	1 s	61.6%
Mean	**2.451e+05**	**1.284e+06**	**7.6**	**81.4%**	**1.876e+04**	**3.357e+05**	**4.6**	**81.4%**

Table 3: Comparisons between heuristic-Carrillo and Lipman's approach and heuristic-Incremental Approach over HOMSTRAD data sets. PSU, $TOps$, $Time$, Acc represents Peak Space Usage, Total Operations, Time, and Accuracy respectively. The definitions of these metrics can be found in Section 7. (For the last row, the entries corresponding to the PSU, $TOps$, and $Time$ columns indicates their respective geometric means while the entry corresponding to Acc indicates its arithmetic mean. For judicious comparisons of these central trends for each column across both the methods, the missing entries (—) in a column corresponding to Carrillo and Lipman Approach and the equivalent entries from a column in Incremental Approach are both excluded from the calculations.)

DATA SET	CARRILLO AND LIPMAN				INCREMENTAL			
	PSU	TOps	Time	Acc	PSU	TOps.	Time	Acc
bowman (5)	2.023e+06	1.254e+07	80 s	76.0%	7.354e+04	2.017e+06	16 s	76.0%
CBM_20 (8)	—	—	—	—	4.608e+04	3.004e+07	315 s	80.9%
CBS (4)	4.548e+05	1.705e+06	14 s	44.8%	8.539e+04	6.026e+05	5 s	44.8%
ccH (4)	9.272e+04	3.477e+05	3 s	94.8%	1.552e+04	1.513e+05	1 s	94.8%
ChtBD (5)	6.910e+03	4.284e+04	1 s	95.8%	6.600e+02	2.637e+04	1 s	95.8%
cryst (7)	8.089e+07	1.468e+09	—	—	9.001e+04	1.985e+07	228 s	85.7%
cyclo (6)	2.319e+07	2.435e+08	1598 s	77.3%	2.818e+05	1.745e+07	216 s	77.3%
cyt5 (6)	5.348e+06	5.616e+07	353 s	72.9%	3.056e+04	3.138e+06	32 s	72.9%
cytb (4)	1.789e+05	6.707e+05	6 s	80.6%	3.272e+04	2.940e+05	2 s	80.6%
dhfr (4)	3.903e+05	1.464e+06	13 s	81.2%	6.948e+04	6.292e+05	6 s	81.2%
flav (6)	3.574e+07	3.752e+08	2561 s	80.0%	1.976e+05	1.883e+07	220 s	80.0%
ghf11 (5)	1.731e+06	1.073e+07	73 s	91.9%	4.998e+04	1.872e+06	21 s	91.9%
ghf22 (12)	—	—	—	—	8.118e+04	4.817e+08	4091 s	96.4%
GLA (4)	9.753e+04	3.657e+05	3 s	97.1%	1.487e+04	1.690e+05	1 s	97.1%
Glyco_hydro_18_D2 (4)	5.675e+05	2.128e+06	17 s	64.9%	1.349e+05	8.603e+05	6 s	64.9%
hpr (5)	6.227e+05	3.861e+06	23 s	97.4%	1.376e+04	6.652e+05	7 s	97.4%
hr (5)	5.189e+05	3.217e+06	19 s	97.0%	2.428e+04	6.775e+05	6 s	97.0%

...Continued on next page

Table 3: ...continued

DATA SET	CARRILLO AND LIPMAN				INCREMENTAL			
	PSU	TOps	Time	Acc	PSU	TOps.	Time	Acc
HTH_AraC (4)	2.488e+05	9.328e+05	8 s	29.3%	6.595e+04	4.650e+05	3 s	29.3%
igC1 (5)	7.994e+05	4.957e+06	30 s	82.9%	1.798e+04	8.128e+05	8 s	82.9%
il8 (11)	—	—	—	—	4.635e+04	6.125e+08	6389 s	81.7%
kazal (6)	3.490e+06	3.664e+07	260 s	88.3%	2.906e+04	2.544e+06	26 s	88.3%
kringle (9)	—	—	—	—	6.326e+04	1.098e+08	1228 s	94.7%
kunitz (10)	—	—	—	—	1.666e+04	1.518e+08	1600 s	94.0%
LDLa (4)	4.545e+04	1.704e+05	2 s	89.9%	1.061e+04	8.811e+04	1 s	89.9%
LIM (5)	8.261e+05	5.122e+06	29 s	90.4%	4.932e+04	1.098e+06	10 s	90.4%
MHC_II_C (8)	—	—	—	—	3.261e+04	2.719e+07	289 s	98.1%
mmp (6)	7.422e+06	7.793e+07	462 s	95.6%	2.951e+04	4.762e+06	54 s	95.6%
myb_DNA-binding (5)	5.027e+05	3.116e+06	18 s	93.6%	1.530e+04	5.744e+05	5 s	93.6%
parv (7)	6.898e+06	1.252e+08	897 s	97.9%	1.733e+04	5.843e+06	66 s	97.9%
plantltp (5)	3.208e+05	1.989e+06	12 s	79.6%	2.302e+04	6.228e+05	7 s	79.6%
profilin (5)	1.743e+06	1.080e+07	65 s	93.7%	4.786e+04	1.586e+06	18 s	93.7%
Propep_M14 (4)	3.246e+04	1.217e+05	2 s	93.5%	7.041e+03	8.038e+04	1 s	93.5%
protg (4)	2.079e+05	7.796e+05	6 s	48.2%	2.632e+04	2.864e+05	2 s	48.2%
rep (4)	3.900e+05	1.463e+06	12 s	79.2%	4.154e+04	4.848e+05	3 s	79.2%
rnasemam (6)	9.238e+06	9.700e+07	600 s	87.4%	5.233e+04	5.635e+06	60 s	87.4%
rub (5)	1.688e+04	1.046e+05	1 s	93.5%	4.128e+03	1.136e+05	1 s	93.5%
RuBisCO_large_N (6)	1.986e+07	2.085e+08	1416 s	82.5%	1.019e+05	9.173e+06	98 s	82.5%
scorptoxin (8)	1.368e+08	6.485e+07	—	—	3.258e+04	2.648e+07	268 s	88.1%
seatoxin (5)	5.524e+05	3.425e+06	20 s	67.5%	1.932e+04	6.151e+05	5 s	67.5%
serbact (5)	2.045e+06	1.268e+07	85 s	89.5%	5.896e+04	1.996e+06	22 s	89.5%
slectin (5)	1.782e+06	1.105e+07	71 s	92.5%	4.792e+04	1.722e+06	17 s	92.5%

...Continued on next page

Table 3: ...continued

DATA SET	CARRILLO AND LIPMAN				INCREMENTAL			
	PSU	TOps	Time	Acc	PSU	TOps.	Time	Acc
sodcu (7)	9.748e+07	1.769e+09	—	—	1.078e+05	2.883e+07	347 s	85.0%
squash (4)	8.488e+03	3.183e+04	<1 s	91.1%	2.004e+03	2.422e+04	<1 s	91.1%
sti (5)	3.723e+06	2.308e+07	153 s	77.9%	9.487e+04	3.096e+06	33 s	77.9%
tbpc (4)	2.260e+05	8.475e+05	8 s	94.0%	2.965e+04	3.684e+05	5 s	94.0%
tgfb (5)	8.438e+05	5.232e+06	31 s	89.6%	2.957e+04	1.034e+06	10 s	89.6%
TIG (6)	2.812e+06	2.952e+07	179 s	97.1%	1.574e+04	2.046e+06	21 s	97.1%
TIL (4)	2.499e+05	9.372e+05	8 s	75.1%	2.809e+04	3.362e+05	3 s	75.1%
uce (6)	1.349e+07	1.417e+08	965 s	92.4%	6.006e+04	6.989e+06	75 s	92.4%
WW (4)	1.982e+05	7.431e+05	6 s	61.6%	3.969e+04	2.957e+05	2 s	61.6%
Mean	**9.807e+05**	**5.790e+06**	**32.1**	**83.0%**	**3.133e+04**	**1.062e+06**	**8.5**	**83.0%**

Table 4: Comparisons between heuristic-Carrillo and Lipman's approach and heuristic-Incremental Approach using BALIBASE reference 1 data sets. PSU, $TOps$, $Time$, Acc represents Peak Space Usage, Total Operations, Time, and Accuracy respectively. The definitions of these metrics can be found in Section 7. (For the last row, the entries corresponding to the PSU and $TOps$ columns indicates their respective geometric means while the entry corresponding to Acc indicates its arithmetic mean. For judicious comparisons of these central trends for each column across both the methods, the missing entries (—) in a column corresponding to Carrillo and Lipman Approach and the equivalent entries from a column in Incremental Approach are both excluded from the calculations.)

DATA SET	CARRILLO AND LIPMAN				INCREMENTAL			
	PSU	TOps	Time	Acc	PSU	TOps.	Time	Acc
TEST1 DATA SET								
1aab.msf.ali (4)	2.857e+05	1.072e+06	9 s	72.6%	5.808e+04	4.387e+05	3 s	72.6%
1aboA.msf.ali (5)	8.636e+06	5.354e+07	372 s	38.9%	2.858e+05	6.461e+06	54 s	38.9%
1aho.msf.ali (5)	8.600e+05	5.332e+06	32 s	88.3%	3.814e+04	9.721e+05	10 s	88.3%
1csp.msf.ali (5)	4.555e+05	2.824e+06	16 s	93.2%	1.424e+04	5.706e+05	6 s	93.2%
1csy.msf.ali (5)	2.020e+06	1.252e+07	78 s	79.4%	5.909e+04	1.873e+06	18 s	79.4%
1dox.msf.ali (4)	1.094e+05	4.101e+05	4 s	93.1%	1.647e+04	1.781e+05	2 s	93.1%
1fjlA.msf.ali (6)	9.681e+06	1.016e+08	659 s	93.3%	8.095e+04	5.116e+06	51 s	93.3%
1fkj.msf.ali (5)	1.344e+06	8.330e+06	53 s	90.0%	3.748e+04	1.254e+06	12 s	90.0%
1fmb.msf.ali (4)	2.649e+04	9.934e+04	2 s	86.0%	5.958e+03	6.700e+04	1 s	86.0%
1hfh.msf.ali (5)	2.823e+06	1.750e+07	113 s	84.3%	6.723e+04	2.233e+06	22 s	84.3%
1hpi.msf.ali (4)	1.699e+05	6.371e+05	6 s	76.1%	3.581e+04	2.890e+05	2 s	76.1%
1idy.msf.ali (5)	5.117e+06	3.173e+07	247 s	53.5%	2.311e+05	3.958e+06	33 s	53.5%
1krn.msf.ali (5)	1.045e+06	6.479e+06	41 s	94.6%	5.739e+04	1.304e+06	11 s	94.6%
1pfc.msf.ali (5)	1.918e+06	1.189e+07	85 s	79.5%	7.806e+04	1.919e+06	18 s	79.5%
1plc.msf.ali (5)	1.220e+06	7.561e+06	49 s	88.8%	3.782e+04	1.174e+06	10 s	88.8%
1r69.msf.ali (4)	7.085e+05	2.657e+06	20 s	20.8%	1.218e+05	8.311e+05	6 s	20.8%
1tgxA.msf.ali (4)	1.612e+05	6.044e+05	5 s	72.3%	2.983e+04	2.496e+05	2 s	72.3%

...Continued on next page

Table 4: ...continued

DATA SET	CARRILLO AND LIPMAN				INCREMENTAL			
	PSU	TOps	Time	Acc	PSU	TOps.	Time	Acc
1tvxA.msf.ali (4)	5.973e+05	2.240e+06	17 s	33.7%	1.952e+05	1.083e+06	6 s	33.7%
1ubi.msf.ali (4)	1.063e+06	3.985e+06	33 s	20.2%	2.445e+05	1.488e+06	9 s	20.2%
1wit.msf.ali (5)	5.752e+06	3.566e+07	239 s	63.4%	1.962e+05	3.903e+06	35 s	63.4%
1ycc.msf.ali (4)	5.071e+05	1.902e+06	15 s	69.0%	8.638e+04	7.318e+05	5 s	69.0%
2fxb.msf.ali (5)	2.896e+04	1.796e+05	1 s	97.1%	6.116e+03	1.606e+05	2 s	97.1%
2mhr.msf.ali (5)	1.191e+06	7.383e+06	47 s	96.9%	3.427e+04	1.242e+06	13 s	96.9%
2trx.msf.ali (4)	7.223e+05	2.709e+06	21 s	67.3%	1.137e+05	8.724e+05	6 s	67.3%
3cyr.msf.ali (4)	2.692e+05	1.010e+06	9 s	70.7%	6.026e+04	4.575e+05	4 s	70.7%
451c.msf.ali (5)	3.559e+06	2.206e+07	142 s	61.3%	1.381e+05	3.066e+06	26 s	61.3%
9rnt.msf.ali (5)	9.193e+05	5.700e+06	36 s	95.0%	2.094e+04	9.397e+05	9 s	95.0%
TEST2 DATA SET								
1ad2.msf.ali (4)	5.421e+05	2.033e+06	18 s	87.4%	1.063e+05	8.760e+05	9 s	87.4%
1amk.msf.ali (5)	2.126e+06	1.318e+07	82 s	97.9%	4.202e+04	2.124e+06	29 s	97.9%
1ar5A.msf.ali (4)	3.190e+05	1.196e+06	11 s	87.7%	6.170e+04	5.451e+05	7 s	87.7%
1aym3.msf.ali (4)	6.050e+05	2.269e+06	20 s	86.5%	1.258e+05	1.005e+06	11 s	86.5%
1bbt3.msf.ali (5)	5.068e+07	3.142e+08	—	—	1.232e+06	2.412e+07	251 s	38.2%
1ezm.msf.ali (5)	2.299e+06	1.425e+07	93 s	95.5%	5.212e+04	2.578e+06	38 s	95.5%
1gdoA.msf.ali (4)	1.043e+06	3.911e+06	32 s	78.4%	2.146e+05	1.582e+06	18 s	78.4%
1havA.msf.ali (5)	—	—	—	—	2.329e+06	3.984e+07	447 s	20.5%
1ldg.msf.ali (4)	9.241e+05	3.465e+06	37 s	92.1%	1.532e+05	1.341e+06	17 s	92.1%
1led.msf.ali (4)	4.155e+05	1.558e+06	15 s	46.1%	8.343e+04	7.163e+05	9 s	46.1%
1mrj.msf.ali (4)	6.045e+05	2.267e+06	21 s	88.8%	1.124e+05	9.605e+05	12 s	88.8%
1pgtA.msf.ali (4)	6.720e+05	2.520e+06	20 s	83.6%	8.219e+04	9.807e+05	10 s	83.6%
1pii.msf.ali (4)	1.076e+06	4.034e+06	34 s	79.7%	1.899e+05	1.536e+06	15 s	79.7%

...Continued on next page

Table 4: ...continued

DATA SET	CARRILLO AND LIPMAN				INCREMENTAL			
	PSU	TOps	Time	Acc	PSU	TOps.	Time	Acc
....TEST2 DATA SET continued								
1ppn.msf.ali (5)	1.748e+06	1.084e+07	69 s	62.1%	5.105e+04	1.988e+06	26 s	62.1%
1pysA.msf.ali (4)	5.001e+05	1.875e+06	17 s	91.5%	9.470e+04	8.382e+05	10 s	91.5%
1sbp.msf.ali (5)	5.622e+07	3.485e+08	—	—	3.777e+06	6.602e+07	712 s	51.1%
1thm.msf.ali (4)	3.981e+05	1.493e+06	15 s	89.8%	5.278e+04	6.734e+05	10 s	89.8%
1tis.msf.ali (5)	4.477e+06	2.776e+07	178 s	94.2%	1.470e+05	4.557e+06	56 s	94.2%
1ton.msf.ali (5)	1.079e+07	6.688e+07	451 s	76.8%	3.672e+05	8.285e+06	100 s	76.8%
1uky.msf.ali (4)	2.863e+06	1.074e+07	83 s	25.5%	3.617e+05	2.819e+06	22 s	25.5%
1zin.msf.ali (4)	3.197e+05	1.199e+06	10 s	91.1%	5.846e+04	5.421e+05	7 s	91.1%
2cba.msf.ali (5)	1.967e+07	1.220e+08	—	—	6.781e+05	1.445e+07	170 s	62.2%
2hsdA.msf.ali (4)	3.186e+06	1.195e+07	100 s	53.9%	5.493e+05	3.843e+06	36 s	53.9%
2pia.msf.ali (4)	4.180e+06	1.567e+07	126 s	54.5%	5.522e+05	4.233e+06	37 s	54.5%
3grs.msf.ali (4)	4.158e+06	1.559e+07	118 s	31.5%	5.899e+05	4.249e+06	34 s	31.5%
5ptp.msf.ali (5)	3.910e+06	2.424e+07	151 s	82.3%	9.144e+04	3.274e+06	42 s	82.3%
kinase.msf.ali (5)	2.180e+07	1.352e+08	982 s	56.0%	6.593e+05	1.339e+07	150 s	56.0%
TEST3 DATA SET								
1ac5.msf.ali (4)	3.869e+06	1.451e+07	141 s	71.8%	3.803e+05	4.090e+06	51 s	71.8%
1ad3.msf.ali (4)	6.387e+05	2.395e+06	27 s	94.9%	1.171e+05	1.091e+06	22 s	94.9%
1adj.msf.ali (4)	8.992e+05	3.372e+06	33 s	94.4%	9.403e+04	1.278e+06	25 s	94.4%
1ajsA.msf.ali (4)	5.709e+06	2.141e+07	174 s	31.2%	4.826e+05	5.281e+06	54 s	31.2%
1cpt.msf.ali (4)	2.942e+06	1.103e+07	103 s	72.0%	5.132e+05	3.785e+06	43 s	72.0%
1dlc.msf.ali (4)	—	—	—	—	4.372e+05	3.473e+06	57 s	79.9%
1eft.msf.ali (4)	1.566e+06	5.871e+06	55 s	81.2%	2.883e+05	2.302e+06	28 s	81.2%
1fieA.msf.ali (4)	1.394e+06	5.226e+06	67 s	91.6%	2.251e+05	2.078e+06	54 s	91.6%

...Continued on next page

Table 4: ...continued

DATA SET	CARRILLO AND LIPMAN				INCREMENTAL			
	PSU	TOps	Time	Acc	PSU	TOps.	Time	Acc
1gowA.msf.ali (4)	5.217e+06	1.956e+07	171 s	61.2%	5.559e+05	4.577e+06	63 s	61.2%
1gpb.msf.ali (5)	7.963e+06	4.937e+07	428 s	95.9%	2.126e+05	7.938e+06	226 s	95.9%
1gtr.msf.ali (5)	5.924e+06	3.673e+07	271 s	94.0%	1.534e+05	5.605e+06	99 s	94.0%
1lcf.msf.ali (6)	5.629e+07	5.910e+08	—	—	6.604e+05	3.935e+07	1120 s	90.7%
1lvl.msf.ali (4)	4.465e+06	1.675e+07	170 s	32.5%	7.320e+05	7.945e+06	86 s	32.5%
1ped.msf.ali (3)	2.604e+05	6.076e+05	10 s	59.6%	2.097e+05	4.894e+05	5 s	59.6%
1pkm.msf.ali (4)	1.320e+06	4.948e+06	47 s	83.5%	2.239e+05	1.898e+06	31 s	83.5%
1rthA.msf.ali (5)	6.312e+06	3.913e+07	286 s	92.0%	1.523e+05	5.488e+06	118 s	92.0%
1sesA.msf.ali (5)	9.923e+06	6.152e+07	424 s	90.1%	1.932e+05	7.006e+06	108 s	90.1%
1taq.msf.ali (5)	—	—	—	—	2.508e+06	4.349e+07	945 s	84.5%
2ack.msf.ali (5)	4.276e+07	2.651e+08	—	—	1.238e+06	2.446e+07	356 s	74.8%
2myr.msf.ali (4)	3.577e+07	1.341e+08	—	—	4.450e+06	2.781e+07	242 s	27.4%
3lad.msf.ali (4)	1.446e+06	5.424e+06	54 s	86.7%	2.447e+05	2.128e+06	33 s	86.7%
3pmg.msf.ali (4)	9.325e+05	3.497e+06	42 s	93.5%	1.807e+05	1.582e+06	35 s	93.5%
4enl.msf.ali (3)	5.686e+05	1.327e+06	20 s	41.3%	5.184e+05	1.210e+06	8 s	41.3%
actin.msf.ali (5)	3.765e+06	2.334e+07	164 s	93.6%	1.157e+05	4.091e+06	66 s	93.6%
arp.msf.ali (5)	1.750e+07	1.085e+08	753 s	80.2%	8.815e+05	1.700e+07	226 s	80.2%
gal4.msf.ali (5)	—	—	—	—	2.835e+06	5.879e+07	738 s	36.4%
glg.msf.ali (5)	2.791e+07	1.730e+08	1248 s	76.3%	4.495e+05	1.447e+07	221 s	76.3%
Mean	**1.831e+06**	**8.644e+06**	**52.8**	**75.1%**	**1.469e+05**	**2.145e+06**	**19.2**	**75.1%**