

# Large Grain Size Stochastic Optimization Alignment

Perry Ridge<sup>1</sup>, Hyrum Carroll<sup>1</sup>, Dan Sneddon, Mark Clement, Quinn Snell  
Computer Science Department, Brigham Young University  
Provo, Utah 84602, USA  
perry.ridge@gmail.com, {hdc,dsneddon,clement,snell}@cs.byu.edu

## Abstract

*DNA sequence alignment is a critical step in identifying homology between organisms. The most widely used alignment program, ClustalW, is known to suffer from the local minima problem, where suboptimal guide trees produce incorrect gap insertions. The optimization alignment approach, has been shown to be effective in combining alignment and phylogenetic search in order to avoid the problems associated with poor guide trees. The optimization alignment algorithm operates at a small grain size with alignment occurring as each tree is searched, but wastes time producing multiple sequence alignments for suboptimal trees.*

*This research develops and analyzes a large grain size algorithm for optimization alignment that iterates through steps of alignment and phylogeny search, thus improving the quality of guide trees used for computation of multiple sequence alignments and eliminating computation of multiple sequence alignments for sub-optimal guide trees. Local minima are avoided by the use of stochastic search methods. Large Grain Size Stochastic Optimization Alignment (LGA) exploits the relationship between phylogenies and multiple sequence alignments, and in so doing achieves improved alignment accuracy.*

*LGA is licensed under the GNU General Public License. Source code and the data sets referenced in this work are publicly available at <http://csl.cs.byu.edu/lga/>.*

## 1. Introduction

The explosion in DNA sequence data has revolutionized the way scientists perform biological and genetic analysis. Through analyzing sequence data for different species, researchers can determine which species are most closely related and make conservation decisions based on these results [19]. Multiple sequence alignment (MSA) is frequently the first step in determining where active regions in proteins are located and plays a critical role in understanding the function of many genes and how they behave in organisms to govern life. Once active regions have been located, drugs can be designed to assist

---

<sup>1</sup>Equally contributing authors

with protein regulation. Alignment also plays a central role in sequence analysis as the first step in comparing corresponding regions in the genomes of different organisms (comparative genomics). Since a refined multiple sequence alignment is crucial to so many different types of life-saving research, it is surprising that multiple sequence alignment does not receive more attention from the research community.

There are a number of different ideas for handling unaligned sequence data. The most popular method is to perform a MSA and then a phylogeny search. ClustalW [28] is the most popular software used for performing the MSA, and there are a variety of software packages available for performing the subsequent phylogenetic search. Most MSA algorithms either require too much execution time and/or suffer from the classic *chicken and egg* problem: to get an accurate MSA in reasonable time (an exhaustive search would return an optimal MSA) requires an accurate phylogeny; yet, traditional methods for obtaining a refined phylogeny require a refined MSA. When these two problems are solved separately the result is a stepwise solution where the accuracy of the final solution depends on the accuracy of the first step. The quality of the results from a phylogenetic search are dependent on the quality of the MSA; therefore, the worse the multiple sequence alignment, the worse the phylogeny [11, 20, 21].

The processes involved in alignment and phylogeny reconstruction are intertwined and researchers often ignore the inter-relationships.

- Phylogeny estimation relies heavily on the alignment [11]. If gaps are inserted incorrectly, significantly different phylogenies will be selected. In many cases, an alignment based on a neighbor-joining tree [25] is used for the phylogenetic search, which may bias the reconstructed phylogeny towards a neighbor-joining tree topology. This *chicken and egg* problem between phylogeny search and alignment is often ignored in reporting results.
- Many researchers have investigated substitution matrices to create more accurate alignments [16, 28]. Substitution matrices allow the alignment to penalize mutations differently that are less likely to have occurred, based on the probability of a given mutation. Probabilities are calculated by determining the frequency of nucleotide change between ancestors and their closest descendants in a phylogenetic tree. Some alignment algorithms use different probabilities for each column or position in the sequence. These methods depend on the phylogeny and alignment to correctly build an alignment.

The interrelationship between multiple sequence alignment, phylogeny construction and substitution matrices necessitates a combined approach to all of these problems. This research develops a feedback-based scheme, Large Grain Size Stochastic Optimization Alignment (LGA), that creates a MSA based on an optimal phylogeny and substitution matrix. A phylogenetic search is then performed using the improved alignment. Since the alignment is more accurate, the inferred phylogeny will be more accurate. A new substitution matrix is built based on the new phylogeny and the whole process is repeated. LGA avoids local minima by using a simulated annealing stochastic search method. The combination of each of these three processes,

multiple sequence alignment, phylogeny estimation, and calculation of a substitution matrix, yields improved results for each of these three processes.

## 1.1. Alignment

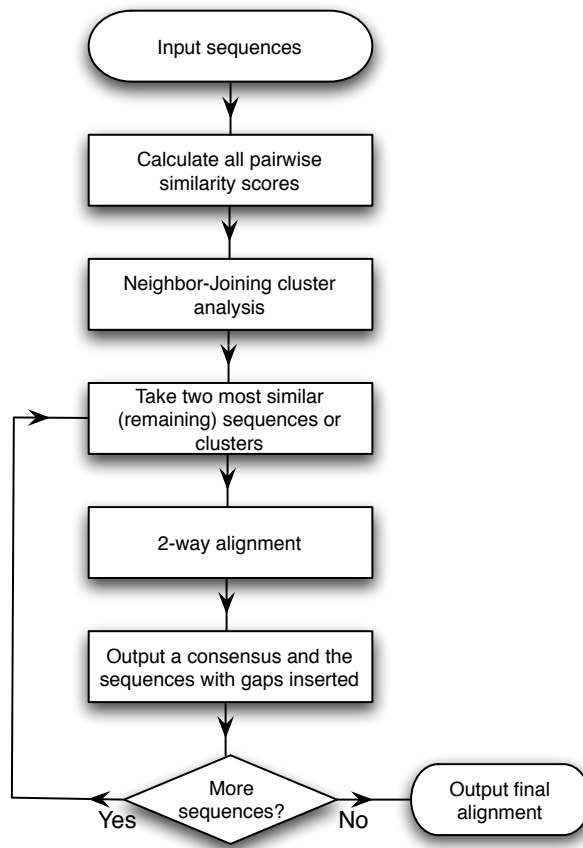
Multiple sequence alignment can be performed with DNA nucleotide or protein (amino acid) sequences. Because multiple sequence alignment is an NP-complete problem [3, 18], the result is an approximation of the true alignment for all but the smallest data sets. MSA algorithms insert gaps in order to align the sequences to maximize similarity according to the evolutionary model summarized in the substitution matrix [28]. Gaps correspond to an insertion or deletion of a substring (sometimes a single residue). Gaps can occur because of single mutations, unequal crossover in meiosis, DNA slippage in the replication process or translocation of DNA between chromosomes.

Multiple sequence alignments are also strongly dependent on the guide tree used. Each alignment is associated with a unique phylogeny. It is not realistic to perform an alignment for every single possible phylogeny in the sample space; there are  $(2n-3)!!$  possible guide trees, where  $n$  is the number of sequences in the data set [10]. Inspection of every single guide tree would cause the execution time to be astronomically long. However, this isn't the only problem: inspection of every possible tree would be wasteful, as most phylogenies are suboptimal and result in weak alignments.

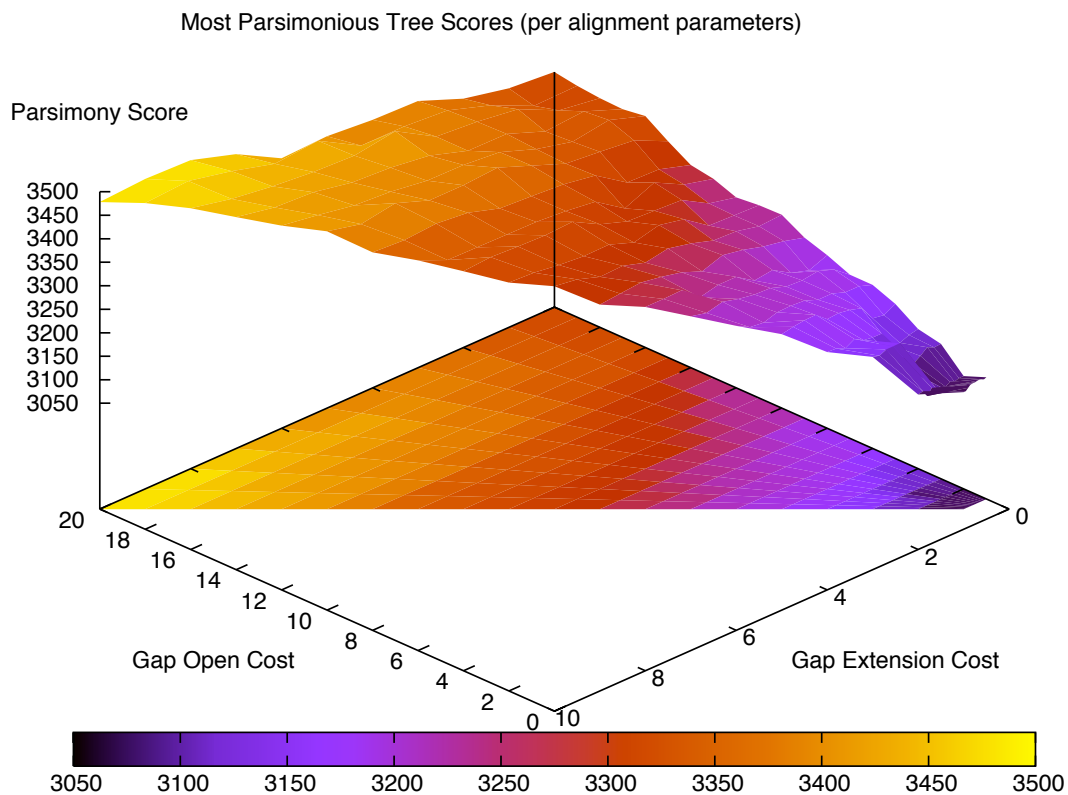
One of the most popular algorithms for MSA is the progressive sequence alignment algorithm [11, 28]. In a progressive sequence alignment algorithm, the substitution matrix is used to determine the likelihood of an observed mismatch (the mismatch may be the result of a mutation or sequencing error). The algorithm then decides to either insert a gap or allow the mismatch to remain in the alignment. In progressive sequence alignment algorithms, inserted gaps are never removed.

A popular alignment program, ClustalW, is used in this research. ClustalW utilizes the progressive sequence alignment algorithm. A multiple sequence alignment performed by ClustalW proceeds as follows: first, pairwise alignments are performed using the Wilbur and Lipman algorithm [32] to obtain pairwise similarity scores for all sequences in the data set [17]. These similarity scores are only very general approximations, but work as a starting point [32]. A similarity matrix is then created to cluster the sequences based on pairwise alignment scores. If a guide tree has not been provided by the user, then a neighbor-joining tree is constructed using the distances in the similarity matrix [4]. In the next step, ClustalW uses a modified version of the Needleman-Wunsch algorithm [22], performing pairwise alignments of the most closely related sequences and generating a consensus sequence for each pair of sequences aligned. ClustalW follows the topology of the guide tree until all sequences have been included in the alignment (see Figure 1).

A more recent alignment package is MUSCLE [6, 5]. MUSCLE generally achieves shorter execution times due to its heuristics for building phylogenies to guide a progressive alignment. It first uses k-mer counting and UPGMA [7] to build an initial phylogeny. With that phylogeny it performs a progressive alignment. Next, it generates another phylogeny using the Kimura distance and UPGMA. After, it computes a progressive alignment with the revised phylogeny and starts to



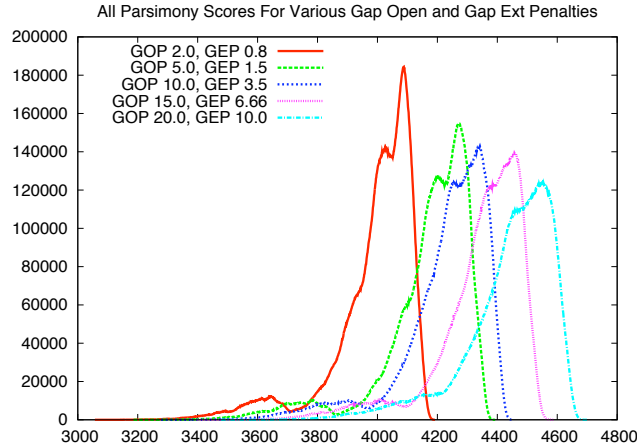
**Figure 1. ClustalW flowchart**



**Figure 2. Optimal parsimony scores for 200 alignments of the P-53 data set. The minimum optimal parsimony score of 3,057 has a gap open penalty of 2.0 and a gap extension penalty of 0.8.**

perform refinement iterations. The iterations consist of splitting the phylogeny and computing the induced profile alignments, followed by aligning the two profiles. MSAs with improved sum-of-pairs scores are used in the next iteration.

Since there are several accepted methods for computing a multiple sequence alignment, it is difficult to evaluate the accuracy of an alignment. The alignment score is dependent on the substitution matrix and gap penalties. ClustalW provides an alignment score for each multiple sequence alignment performed. However, since this score is dependent on the substitution matrix and gap penalties this score cannot be used to compare different alignments of the same data set. The minimum cost for a phylogeny inferred from a given MSA has been suggested as an unbiased measure of the quality of the alignment [31, 29]. Because no better, unbiased, metrics have been commonly used, this research uses the minimum cost phylogeny to determine alignment quality.



**Figure 3. Representative histograms of all parsimony scores for various alignments. Exhaustive search performed with `alltrees` in PAUP\* [27]. The alignment produced using a gap open penalty (GOP) of 2.0 and a gap extension penalty (GEP) of 0.8 has the lowest parsimony score of 3,057. ClustalW defaults to a GOP of 15.0 and GEP of 6.66.**

Using a substitution matrix changes the relative contribution of gap open and gap extension penalties compared to match and mismatch costs. Changing the gap open and gap extension penalties significantly impacts the parsimony score for an alignment. Figure 2 illustrates the optimal phylogenies for various alignments of a single data set, varying only the gap open and gap extension penalties in ClustalW. The nature of an exhaustive search used to find the optimal phylogeny for a given alignment necessitates using an appropriately sized data set. Therefore, this experiment used a data set of eleven sequences obtained from a BLAST search [1] at GenBank [2]. The most parsimonious phylogeny was found with gap open and gap extension penalties of 2.0 and 0.8 respectively, whereas the default settings for ClustalW are 15.0 and 6.66 respectively. Not only does the optimal parsimony score vary with alignment settings, but the distribution of parsimony scores also varies. Figure 3 shows five representative histograms of parsimony scores for different gap open and gap extension penalties. While the total number of phylogenies is the same for each of the histograms the shape of each is noticeably different and the optimal tree score varies significantly. The LGA algorithm varies the substitution matrix in order to explore alignments produced by different gap penalties.

Although most phylogenetic search applications use a given multiple sequence alignment as a starting point [8, 9, 13, 27], multiple sequence alignment has received much less attention than phylogenetic search algorithms [24]. The importance of a quality alignment for the phylogeny search must not be minimized [20, 21]. Morrison *et al.* [20] has even suggested that the resulting phylogeny is affected more by the method used for performing the multiple sequence alignment than the method used to perform the phylogeny search itself. An algorithm that varies the alignment, phylogeny and substitution matrix is necessary in order to effectively explore the search space.

## 1.2. Related Work

Other researchers have addressed the problem of dependencies between alignment and phylogeny search. The most closely related research includes TreeAlign [15], MALIGN [31] and POY [12, 30]. Each of these programs uses the parsimony score of the resulting phylogeny as a global optimality criterion for alignment.

TreeAlign computes pairwise distances between sequences to create an initial alignment topology. This tree is then rearranged to create a better fit to the distance matrix. Multiple sequence alignment is then performed using this tree as a guide tree [15]. Although the guide tree is constructed based on parsimony score, the tree is not based on the multiple sequence alignment, but only on a distance matrix. This process does not allow progressive improvements in alignment to impact the guide tree.

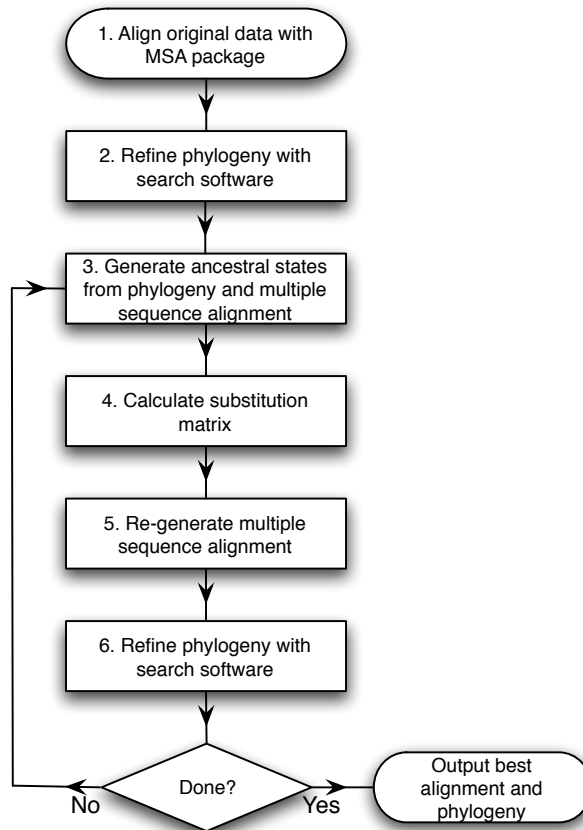
MALIGN and POY construct alignment topologies by constructing a phylogenetic tree and aligning the data concurrently (*optimization alignment*). The combination of tree and alignment that produces the best parsimony score is chosen as the most optimal. By default POY produces a multiple sequence alignment for every tree. This process wastes time because sub-optimal trees will not result in optimal alignments. The majority of possible trees are sub-optimal and therefore, should not be used to create alignments.

There are a number of differences between our research and other optimization alignment approaches:

- The larger-grain size search used in LGA results in a more thorough search of the sample space.
- This iterative approach concentrates on strong phylogenies rather than spending time performing alignments with sub-optimal trees.
- LGA uses a dynamic substitution matrix to define an evolutionary model based on the data set, alignment and phylogeny.
- The simulated annealing approach used in LGA allows the algorithm to avoid local minima.
- More sophisticated algorithms can be used for alignment and phylogeny search since LGA can use a variety of different software packages for each step in the iteration.

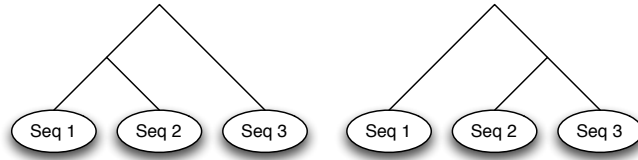
## 2. Large Grain Size Stochastic Optimization Alignment (LGA)

The LGA algorithm iterates through large grain size steps of alignment and phylogeny search, thus eliminating the computation of multiple sequence alignments for suboptimal trees. Local minima are avoided through stochastic search methods (simulated annealing). LGA exploits the relationship between phylogenies and multiple sequence alignments, and in so doing achieves more refined MSAs. The algorithm proceeds as follows:



**Figure 4.** LGA flowchart

1. A multiple sequence alignment algorithm aligns a data set of sequences (without a user-provided guide tree or substitution matrix).
2. A phylogenetic search is performed based on the multiple sequence alignment from the previous step, returning a refined phylogeny.
3. Ancestral states are generated from the inferred phylogeny and multiple sequence alignment.
4. A substitution matrix is calculated by comparing mutations between neighboring nodes in the tree.
5. Multiple sequence alignment is performed on the original unaligned data using the refined phylogeny from step 2 as a guide tree and the substitution matrix from step 4.
6. A phylogenetic search is performed based on the MSA from step 5.
7. If done, output the best multiple sequence alignment and phylogeny found, otherwise, go to step 3.



**Figure 5.** Two trees that each have the same three sequences, but would result in different substitution matrices.

LGA iterates through steps 3-7 (see Figure 4 for a complete flow chart of LGA). The algorithm uses simulated annealing to adjust the substitution matrix (a complete explanation of creation of the substitution matrix follows). Simulated annealing iterations occur for a pre-determined number of times (see Table 1).

Only refined alignments and phylogenies are considered as possible candidates for later iterations. Because of the large granularity of the search, a larger area of the search space is covered in less time than other alignment algorithms, while achieving better results than programs like ClustalW.

The substitution matrix is created by examining data set specific mutations in resulting phylogenies. Therefore, it defines a specific, unique evolutionary model for each data set. The method used to create the substitution matrix is as important as a good guide tree in converging on a more refined alignment. The use of a simulated annealing search for the proper substitution matrix results in convergence on alignments with more refined phylogeny scores.

## 2.1. Creating the Substitution Matrix

The substitution matrix is computed by calculating the relative frequency of nucleotide changes in the phylogenetic tree. A refined phylogeny with estimated ancestral nodes is critical to creating a more parsimonious alignment. For example, Figure 5 shows two different phylogenies, each with different ancestral sequences. As detailed below, different ancestral sequences will result in a different substitution matrix.

The matrix is calculated by comparing nucleotides in corresponding columns of the two sequences. Results are summed between each pair of neighboring nodes (sequences) throughout the tree. For cases where the nucleotides are different (a substitution has occurred), no assumption is made about which nucleotide is the original. For example, the change  $A \Rightarrow T$  is the same as the change  $T \Rightarrow A$ . Figure 6 illustrates the calculation of a substitution matrix. In the example, the two sequences are assumed to be adjacent in a phylogeny.

In the example, in column 1, two adenine(A) nucleotides are present, resulting in an increment of the count for  $A \Rightarrow A$  substitution also shown in row[A], column[A] of Figure 7. The last column (column 18) of each sequence contains a cytosine(C) and thymine(T) respectively. Therefore, the  $C \Rightarrow T$  result is incremented. All matrix counts are shown in Figure 6.

Multiple sequence alignment packages normally use a default substitution matrix to compute the penalty for transitions and transversions. This matrix may not be appropriate for a given data set. Generalizing alignment scoring in this way

```

000000000111111111
123456789012345678
sequence 1: AGTCGGGTTGAACTAGAC
sequence 2: AGTCATCGACGTACTGCT

```

```

A ⇒ A = 1
G ⇒ G = 2
C ⇒ C = 1
T ⇒ T = 1
A ⇒ T or T ⇒ A = 3
A ⇒ C or C ⇒ A = 2
A ⇒ G or G ⇒ A = 2
G ⇒ T or T ⇒ G = 2
G ⇒ C or C ⇒ G = 2
C ⇒ T or T ⇒ C = 2

```

**Figure 6. Sample sequences and resulting calculations for the substitution matrix.**

	A	G	C	T
A	1	2	2	3
G	2	2	2	2
C	2	2	1	2
T	3	2	2	1

**Figure 7. Sample substitution matrix for the sequences in Figure 6.**

does not take into account differences between data sets. Use of the substitution matrix allows multiple sequence alignment algorithms to penalize different substitutions based on observed mutation frequencies. The substitution matrix defines a unique evolutionary model for each data set.

Without scaling (or normalization) of the substitution matrix, values in the matrix will be very high relative to the gap penalties. As a result, the algorithm will prefer gaps to any mismatch and therefore, the final alignment will be sub-optimal with a disproportionately high number of gaps. To avoid sub-optimal alignments, and use the substitution matrix to accurately represent an evolutionary model, each matrix is normalized.

The substitution matrix is normalized by multiplying each entry by the scaling factor (ratio of desired maximum to actual maximum value in the matrix, see Figure 8).

	A	G	C	T	*
A	.33	.66	.66	1	.33
G	.66	.66	.66	.66	.33
C	.66	.66	.33	.66	.33
T	1	.66	.66	.33	.33
*	.33	.33	.33	.33	.33

**Figure 8. Result of normalizing substitution matrix from Figure 7, with a desired maximum value of 1.**

## 2.2. Normalization Algorithm

Normalization of the substitution matrix is critical to the quality of the alignment. As previously stated, parsimony scores from inferred phylogenies from a given MSA are used to measure refinement of MSAs; therefore, reported scores are parsimony scores. As shown in Figure 2, different normalizations of the substitution matrix have a profound effect on the quality of the final alignment. The normalization algorithm is also necessary to find an appropriate normalization of the substitution matrix. The best alignments are achieved when each substitution matrix is normalized according to the specific data set.

The normalization algorithm inspects the parsimony score for the phylogenetic tree after each iteration in order to determine whether the scaling factor should be increased or decreased. Modifications to the scaling factor are made by changing the desired maximum value in the matrix. Frequently, several iterations are required using a given scaling factor before the alignment improves. Several iterations are run for each normalization of the substitution matrix before modification.

The normalization algorithm is the combination of a binary search and simulated annealing. In the first phase of the algorithm, all changes are exponential and in the second phase changes are linear. Simulated annealing begins in the second phase of the algorithm allowing LGA to avoid local minima.

### 1. First Phase - Find a reasonable scaling factor:

Continue until a MSA more refined than the initial MSA from ClustalW is computed:

- If the MSA is less refined than the MSA computed in the previous iteration, exponentially decrease the scaling factor
- Otherwise additively increase the scaling factor

### 2. Second Phase - Refine the scaling factor:

Continue for the specified number of simulated annealing jumps:

- If the MSA is less refined than the best MSA thus far, set the scaling factor midway between the current factor and the factor where the most refined MSA was computed
- Otherwise, linearly increase/decrease (do the same as the previous iteration) the scaling factor

The normalization algorithm refines the substitution matrix as LGA converges on a refined MSA.

Program	Arguments
LGA	-b4jump=6 -times2jump=6
ClustalW	(defaults)
MUSCLE	-maxiters 99 -maxhours 99.0 -maxtrees 99

**Table 1. Arguments for LGA, ClustalW and MUSCLE**

### 3. Results

#### 3.1. Experimental Setup

To evaluate the utility of LGA in light of existing algorithms, we compared the best parsimony score found for 54 nucleotide sequence data sets. Results from the fourteen data sets illustrated in this section are representative of results found overall. In terms of size, the data sets range from 100HIV, with 100 sequences with an average length of 616.6 characters to MP with 332 sequences with an average length of 2158.0 characters. The data sets are from a variety of sources and details about each one are publicly available at <http://csl.cs.byu.edu/lga/>.

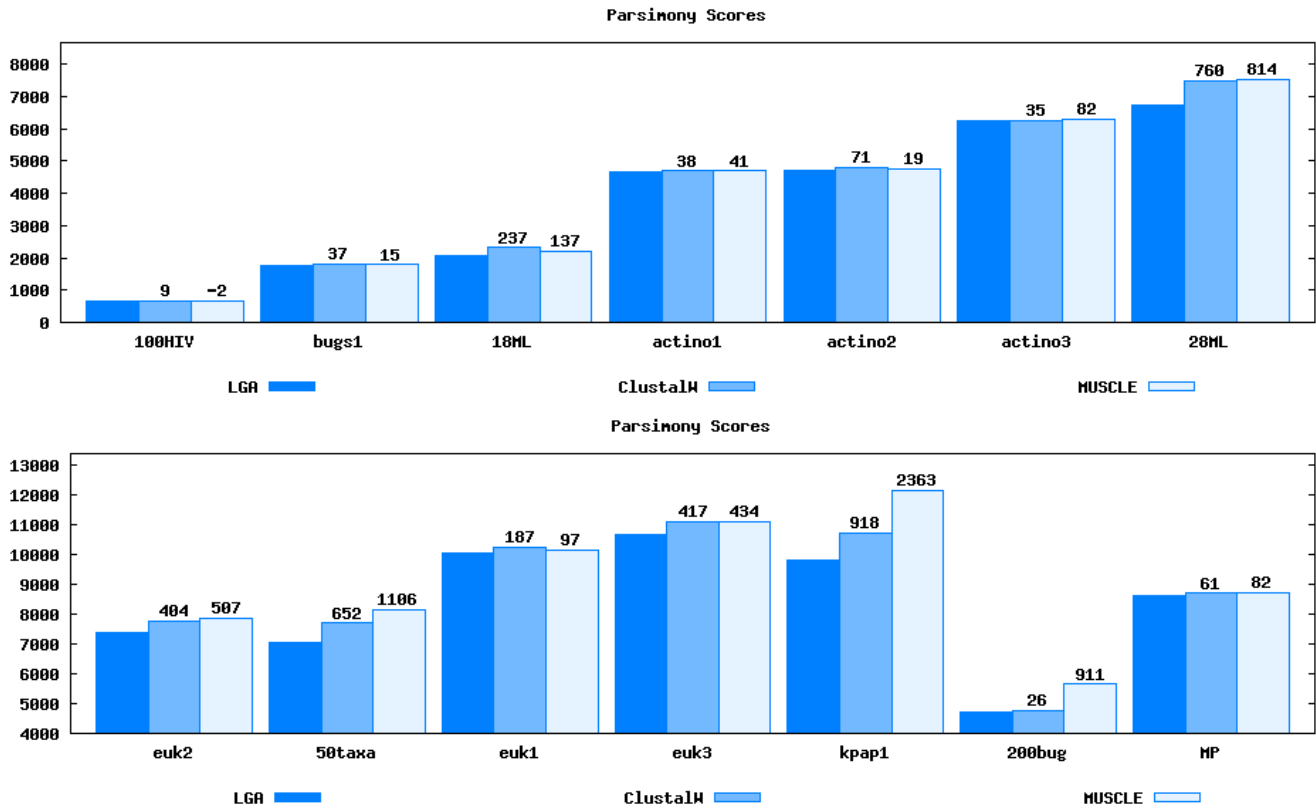
In this paper we compare the alignment produced by LGA, ClustalW and MUSCLE. The parameters used for each algorithm are detailed in Table 1.

All of the experiments were executed with a single processor on a dedicated node (dual 3.6 GHz Xeon processors with 4 GB of memory) at the Fulton Supercomputing Laboratory at Brigham Young University.

All parsimony scores were calculated by PAUP\*, treating a gap as a new character state (lower values are superior). The parsimony scores reported for ClustalW and MUSCLE reflect the best score found after a ratchet [23] search using PAUP\*.

#### 3.2. Algorithm Comparisons

The LGA algorithm produces results that are much better than ClustalW. Figures 9 show the best parsimony score found using LGA, ClustalW and MUSCLE. The differences in parsimony score between LGA and ClustalW, and LGA and MUSCLE are noted in the graph above the respective bar. For all data sets, LGA improves the parsimony score found by ClustalW. Improvements achieved by LGA in parsimony score over ClustalW range from nine to 918 steps and 0.55% to 10.26%. For all but the smallest data set, LGA finds a better phylogeny score than MUSCLE. Improvements range from fifteen to 2,363 steps and 0.40% to 19.43%.



**Figure 9. Parsimony scores for LGA, ClustalW and MUSCLE. The numbers above the bars show the difference in the parsimony scores between LGA and the respective algorithm. Positive values indicate that LGA achieved a more parsimonious alignment.**

#### 4. Conclusions

Many researchers have investigated algorithms for phylogenetic analysis for large data sets without examining multiple sequence alignment. The LGA algorithm described here provides a large grain size approach to optimizing phylogenetic search and alignment concurrently. The LGA algorithm has the following features:

1. LGA achieves an alignment superior to the alignments produced by ClustalW. The phylogenetic trees produced in these alignments can be hundreds of steps better than the trees generated from MSAs produced by ClustalW alone.
2. An analysis of large data sets can be performed with LGA, even when existing optimization alignment algorithms fail to complete. For these data sets, LGA may be the only way to generate optimization alignment results.
3. Intermediate results can be sampled as the computation converges when LGA is used.
4. More sophisticated algorithms can be used for alignment and phylogeny search since LGA can use a variety of different software packages for each step in the iteration.

Future work will examine the impact of more sophisticated alignment and phylogeny search packages in the LGA framework. These improvements can only enhance the LGA results, whereas less flexible alignment and phylogeny search packages are more constrained. Preliminary results from different combinations of ClustalW, MUSCLE, TNT [14] and Soda [26] indicate that greater improvements can be obtained from those reported in this work.

## References

- [1] S. F. Altschul, W. Gish, W. Miller, E. W. Meyers, and D. J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215:403–410, 1990.
- [2] D. A. Benson, I. Karsch-Mizrachi, D. J. Lipman, J. Ostell, and D. L. Wheeler. GenBank. *Nucleic Acids Research*, 33:D34–38, 2005.
- [3] H. Bodlaender, R. Downey, M. Fellows, and H. Wareham. The parameterized complexity of sequence alignment and consensus. *Theoretical Computer Science*, 147:31–54, August 1995.
- [4] R. Chenna, H. Sugawara, T. Koike, R. Lopez, T. Gibson, D. Higgins, and J. Thompson. Multiple sequence alignment with the clustal series of programs. *Nucleic Acids Research*, 31(13):3497–3500, 2003.
- [5] R. C. Edgar. MUSCLE: a multiple sequence alignment method with reduced time and space complexity. *BMC Bioinformatics*, 5(1):113–131, 2004.
- [6] R. C. Edgar. MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Research*, 32(5):1792–1797, 2004.
- [7] I. Eidhammer, I. Jonassen, and W. R. Taylor. *Protein Bioinformatics: An Algorithmic Approach to Sequence and Structure Analysis*. John Wiley and Sons Ltd, 2004.
- [8] J. S. Farris. *HENNIG86, version 1.5*. Program and Documentation: Port Jefferson Station, New York, 1988.
- [9] J. Felsenstein. PHYLIP – phylogeny inference package (version 3.2). *Cladistics*, 5:164–166, 1989.
- [10] J. Felsenstein. *Inferring Phylogenies*. Sinauer Associates, Sunderland, MA, 2004.
- [11] D. Feng and R. F. Doolittle. Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *J. Mol. Evol.*, 60:351–360, 1987.
- [12] D. S. Gladstein and W. C. Wheeler. *POY: The Optimization of Alignment Characters*. New York, NY, 1997-2000. Program and documentation. Available at <ftp://ftp.amnh.org/pub/molecular/poy/>.
- [13] P. Goloboff. Analyzing large datasets in reasonable times: Solutions for composite optima. *Cladistics*, 15:415–428, 1999.
- [14] P. Goloboff, S. Farris, and K. Nixon. TNT: Tree analysis using new technology. <http://www.cladistics.com/webtnt.html>, 2001.
- [15] J. Hein. A tree reconstruction method that is economical in the number of pairwise comparisons used. *Mol. Evol. Biol.*, 6:669–684, 1989.
- [16] S. Henikoff and J. G. Henikoff. Amino acid substitution matrices from protein blocks. *PNAS*, 89:10915–10919, 1992.
- [17] D. G. Higgins and P. M. Sharp. Clustal: A package for performing multiple sequence alignment on a microcomputer. *Gene*, 73:237–244, 1988.

- [18] T. Jiang, E. Lawler, and L. Wang. Aligning sequences via an evolutionary tree: complexity and approximation. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 760–769. ACM Special Interest Group on Algorithms and Computation Theory, ACM Press New York, NY, USA, 1994.
- [19] W.-H. Li. *Molecular Evolution*. Sinauer Associates, Sunderland, Massachusetts, 1997.
- [20] D. Morrison and J. Ellis. Effects of nucleotide sequence alignment on phylogeny estimation: a case study of 18s rdnas of apicomplexa. *Molecular Biology and Evolution*, 14:428–441, 1997.
- [21] N. Mugridge, D. Morrison, T. Jakel, A. Heckerth, A. Tenter, and A. Johnson. Effects of sequence alignment and structural domains of ribosomal dna on phylogeny reconstruction for the protozoan family sarcocystidae. *Molecular Biology and Evolution*, 17:1842–1853, 2000.
- [22] S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol.*, 48:443–453, 1970.
- [23] K. Nixon. The parsimony ratchet, a new method for rapid parsimony analysis. *Cladistics*, 15:407–414, 1999.
- [24] A. Phillips, D. James, and W. Wheeler. Multiple sequence alignment in phylogenetic analysis. *Molecular Phylogenetics and Evolution*, 16(3):317–330, September 2000.
- [25] N. Saitou and M. Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Mol. Biol. Evol.*, 4:406–425, 1987.
- [26] Soda. <http://csl.cs.byu.edu/soda>, 2006.
- [27] D. L. Swofford. *PAUP\*. Phylogenetic Analysis Using Parsimony (\* and Other Methods). Version 4*. Sinauer Associates, Sunderland, Massachusetts, 2003.
- [28] J. D. Thompson, D. G. Higgins, and T. J. Gibson. Clustal W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position specific gap penalties and weight matrix choice. *Nucleic Acids Research*, 22:4673–4680, 1994.
- [29] W. C. Wheeler. Optimization alignment: the end of multiple sequence alignment in phylogenetics? *Cladistics*, 12:1–9, 1996.
- [30] W. C. Wheeler, D. Gladstein, and J. DeLaet. *POY [3.0]*. American Museum of Natural History, New York, 1997.
- [31] W. C. Wheeler and D. S. Gladstein. MALIGN: A multiple sequence alignment program. *J. Hered.*, 85:417–418, 1994.
- [32] W. Wilbur and D. Lipman. The context dependent comparison of biological sequences. *SIAM J. Appl. Math.*, 44:557–567, 1984.