GESTALT: Genomic Steiner Alignments

Giuseppe Lancia*[1] and R. Ravi**[2]

[1] Dipartimento Elettronica ed Informatica, University of Padova, lancia@dei.unipd.it
[2] GSIA, Carnegie Mellon University, ravi@cmu.edu

Abstract. We describe GESTALT (GEnomic sequences STeiner ALignmenT), a public–domain suite of programs for generating multiple alignments of a set of biosequences. We allow the use of either of the two popular objectives, Tree Alignment or Sum-of-Pairs. The main distinguishing feature of our method is that the alignment is obtained via a tree in which the internal nodes (ancestors) are labeled by Steiner sequences for triples of the input sequences. Given lists of candidate labels for the ancestral sequences, we use dynamic programming to choose an optimal labeling under either objective function. Finally, the fully labeled tree of sequences is turned into into a multiple alignment. Enhancements in our implementation include the traditional space-saving ideas of Hirschberg as well as new data-packing techniques. The running-time bottleneck of computing exact Steiner sequences is handled by a highly effective but much faster heuristic alternative. Finally, other modules in the suite allow automatic generation of linear-program input files that can be used to compute new lower bounds on the optimal values. We also report on some preliminary computational experiments with GESTALT.

1 Introduction

Comparing genomic sequences drawn from individuals of the same or different species is one of the fundamental problems in computational molecular biology. These comparisons can (i) lead to the identification of highly conserved (and therefore presumably functionally relevant) genomic regions, (ii) spot fatal mutations, (iii) suggest evolutionary relationships, (iv) help in correcting sequencing errors etc. Therefore, the mathematical formulation and solution of the *Multiple Sequence Alignment* problem has been and remains a fundamental challenge for computational molecular biologists.

Aligning a set of sequences consists in arranging them in a matrix having each sequence in a row. This is obtained by possibly inserting spaces (gaps) in each sequence so that they all have the same length. The

* Most of this work was done when this author was visiting CMU during Summer '98, under a grant from the CMU Faculty Development Fund.

** Supported in part by an NSF CAREER grant CCR-9625297

following is a simple example of an alignment of the sequences ATTCGAC, TTCCGTC and ATCGTC.

```
A T T - C G A - C
- T T C C G - T C
A - T - C G - T C
```

There are many popular formulations of the alignment problem. The choice of the objective function for multiple alignments depends mainly on the presence or absence of extra input information in the form of a phylogenetic tree relating the sequences to their unknown ancestors. In fact, when such tree is given, knowledge of the ancestral sequences would imply the possibility of aligning the given sequences by progressively aligning each sequence to its ancestor in the tree all the way to the root and chaining these pairwise alignments together [6]. Hence when a phylogeny is given, the *tree alignment (TA)* objective consists in finding the best ancestral sequences to label this tree and deriving the induced alignment. Guided by parsimony, the best labeling is taken to be one minimizing the total evolutionary change represented in the tree, namely, the total distance of all the edges in the tree. When the phylogenetic tree is not available, a popular multiple alignment objective is the *Sum–of–pairs (SP)* objective, which attempts to minimize the average distance between a pair of sequences in the multiple alignment. This objective results naturally by extending the alignment objective for pairs of sequences, namely, that of minimizing the edit-distance between the pair, to more than two sequences. The SP objective has been popular in the literature and several heuristic implementations addressing it proceed by first finding a heuristic tree spanning the sequences and aligning them progressively as mentioned earlier to obtain the final alignment.

Historically, the SP objective is the one to which more attention has been devoted by computational biologists, and correspondingly a set of programs have been developed which are now widely in use. Among them, the only program that computes optimal SP alignments is MSA by Lipman, Altschul, Kececioglu, Gupta and Schaeffer [2, 8]. A variety of other multiple sequence alignment programs implicitly use the SP objective in guiding heuristic construction of the multi-alignments: An example is CLUSTAL V [11] (see also the various methods described in the surveys [16, 5] for other examples). As for tree alignment, the only implementation that addresses this problem directly that we are aware of is the recent TAAR by Jiang and Liu [13]. This program implements some of the ideas from the approximation algorithms of Jiang, Lawler and Wang [27] to heuristically compute tree alignments, phylogenies and generalized tree alignments.

In this paper we introduce and describe a new public–domain suite of programs for multiple sequence alignment that produce heuristic alignments under both the TA and SP alignment objectives. Like TAAR, Our methods are based on ideas used in an approximation algorithm for tree alignment due to Ravi and Kececioglu [17]. However, unlike the methods of Jiang, Lawler and Wang [27] on which TAAR is based, whose refined heuristics require very high running times, the ideas of Ravi and Kececioglu are based on mainly computing and using Steiner sequences as candidates for the unlabeled ancestral sequences in the tree. Intuitively, a Steiner sequence for a given set of sequences is a "central" sequence to them, one whose sum of distances to all these sequences is minimized. Once these Steiner sequences for appropriate subsets of the input sequences have been computed, dynamic programming can be used to efficiently pick one such sequence for each ancestral node so as to minimize the total resulting distance in the tree, as in [27]. Thus, this method is adaptable for efficient implementation giving us the freedom to specify the subsets of sequences for which the Steiner sequences must be computed. Further, we can effectively adapt this general idea by modifying the dynamic program to provide an efficient heuristic even for the SP objective using the postulated Steiner ancestors.

Further refinements in our implementation include incorporating the traditional space-saving ideas of Hirschberg [12] as well as some new data-packing techniques to reduce the space overhead; The running-time bottleneck in our method of computing exact Steiner sequences is effectively handled by a much faster heuristic alternative that has never shown more than two percent degradation in quality in our extensive preliminary testing. Finally, other programs in the suite allow automatic generation of linear-programming models as files that can be input to the popular commercial CPLEX package. The solution of these programs give lower bounds on the minimum TA and SP alignment values for the given set of sequences, thus providing the deviations from optimality on a case-by-case basis.

We formally describe the various objectives and methods in the remainder of this section. In Sect. 2 we give a high–level description of the algorithms in GESTALT, together with an analysis of the individual steps. In Sect. 3 we report on some experimental results on real data.

1.1 Edit Distance

At the heart of any alignment algorithm lies the procedure for optimally comparing two given sequences. This problem is called pairwise alignment, and is formulated as follows. Given symmetric costs $c(a, b)$ for replacing a symbol a with a symbol b and costs $c(a, -)$ for deleting (inserting) symbol a, find a minimum–cost set of symbol operations that turn a sequence S' into a sequence S''. It is well known that this problem can be solved by dynamic programming in time and space $O(l^2)$, where l is the length of the sequences. The value of an optimal solution is called the *edit distance* of S' and S'' and denoted by $d(S', S'')$.

An *alignment* $\mathcal{A}$ of two (or more) sequences is a way of inserting "$-$" characters (gaps) in the sequences so that the resulting sequences have the same length. For two sequences S' and S'', the value $d_{\mathcal{A}}(S', S'')$ of their alignment is obtained by adding up the costs for the pairs of characters in corresponding positions. It is immediate that $d(S', S'') = \min_{\mathcal{A}} d_{\mathcal{A}}(S', S'')$.

1.2 The Sum–of–Pairs Alignment Problem

The SP score is the generalization to many sequences of the pairwise alignment objective, in which the cost of the alignment is obtained by adding the costs of the symbols matched up at the same positions. Analogously, in a multiple alignment the cost is obtained by adding up the matching characters, over all the positions and for all the pairs of sequences.

Minimizing SP is NP-hard [26]. In [9] Gusfield showed that a tree-based progressive alignment method due to Feng and Doolittle (described below) using the minimum cost star gives a 2–approximation. In the program described in this paper we push this idea further, by considering also trees that are not only stars and also employing alignments with sequences which are not in the original set, but are derived from it as Steiner sequences of some of the original ones.

1.3 The Tree Alignment Problem

In the tree alignment problem, we are given n sequences related by an evolutionary tree T. The sequences label the leaves of the tree, while the internal nodes correspond to the unknown ancestral sequences from which the others have evolved. The problem consists in finding the sequences at the internal nodes which minimize the cost of the tree, defined as $\sum_{(S_i,S_j)\in T} d(S_i, S_j)$. When T is a star, the problem is called a *Steiner* *problem*, and the optimal sequence for the center is called the *Steiner sequence* for the leaves.

The first exact algorithm for tree alignment was proposed by Sankoff in [18], and is based on dynamic programming. Later Altschul and Lipman [1] introduced some bounding rules to reduce the size of the dynamic programming lattice. Due to the prohibitive worst case complexity of exact methods, approximation algorithms for this problem were devised, by Jiang, Lawler and Wang [27] first, and improved by Wang and Gusfield [25] later. In [27] a 2–approximation method is described, based on what are called *lifted alignments*. In lifted alignments, the internal nodes can only be labeled by sequences occurring at the leaves. The running time of their algorithm is $O(n^2l^2 + n^3)$ for a tree of n leaves of length l. For trees of bounded degree d, they also provided the first PTAS for the problem. For any t, their approximation scheme guarantees a solution within a factor $1 + \frac{3}{t}$ of optimal, in time $O(n^{2+d^{t-1}} l^{d^{t-1}-1/d-1})$.

For regular d–ary trees on n sequences, Ravi and Kececioglu gave in [17] a $\frac{d+1}{d-1}$–approximation algorithm with running time roughly $(O(2kn)^d)$ – the main ideas of their algorithm are briefly described in Sect. 2. The program GESTALT described in this paper is the first implementation of the ideas in [17].

1.4 A Tree-based Progressive Alignment Method

A reasonable requirement on the cost function is that $c(a, a) = 0\ \forall a$, and it obeys triangle inequality. In this case, the edit distance induces a metric over the space of all sequences and, given n sequences, we can talk of graphs having the sequences as vertices and for which an edge is weighted by the edit distance between the endpoints. In this setting, graph theoretical concepts such as spanning trees, stars and Steiner points, have been widely used in the design and analysis of effective alignment algorithms. In particular, a folklore approach to multiple alignments is due to Feng and Doolittle [6] and shows how we can use any tree to align a set of n sequences. The appeal of the approach is that for $n - 1$ out of $n(n - 1)/2$ pairs, the pairwise alignment induced is in fact optimal.

Proposition 1. *For any tree T over a set of sequences, there exists a multiple alignment $\mathcal{A}(T)$ of the sequences such that $d_{\mathcal{A}(T)}(S', S'') = d(S', S'')$ for all the pairs of sequences (S', S'') connected by an edge of the tree.*

Feng and Doolittle's method can be used to turn the solution of the tree alignment problem, namely a labeling of the internal nodes of the given tree, into a multiple alignment of the leaves. Moreover, it is straightforward to upper bound the distance in this alignment of pairs that are not endpoints of a tree edge. In fact, denote by $d(S', S'', T)$ the length of the path in T between two sequences S' and S''. Then, by triangular inequality we have that $d_{\mathcal{A}(T)}(S', S'') \leq d(S', S'', T)$. This inequality suggests that, given a tree with sequences at the leaves for which we want to minimize average pairwise distance in the resulting multiple alignment, a good labeling for the internal nodes is one which minimizes the total inter-leaf distance in the tree. This strategy is adopted in this work to obtain alignments of small SP value, as described in 2.3.

1.5 GESTALT Program Suite

In this paper we describe the program GESTALT (GEnomic sequences STeiner ALignmenT), which can be used for both TA and SP multiple alignments. GESTALT is in fact a program suite, including modules for computing LP-based lower bounds for TA and SP, and optimal alignments of two or three sequences.

The main program takes as input a set $\mathcal{L} = \{S_1, \ldots, S_n\}$ of n sequences and possibly a tree T of which $\mathcal{L}$ are the leaves. If the phylogenetic tree is not available, the algorithm internally computes one, which is then used to find an alignment of small SP value. If the tree is given, then the TA objective is optimized[1]. The output of the algorithm consists of a multiple alignment of the input sequences, plus some extra information, such as the Steiner sequences computed at the internal nodes of the phylogenetic tree.

GESTALT is based on the ideas introduced by Ravi and Kececioglu in [17] of using Steiner sequences of the leaves to label the internal nodes of the tree. While in their paper Ravi and Kececioglu show that if the tree is d–ary the method gives a $\frac{d+1}{d-1}$ approximation for TA, in our work we do not restrict the degree of each node to a constant. Therefore we do not have the same approximation guarantee. However, among all the labelings considered is included the best lifted labeling of [27] and therefore we still have a performance guarantee of 2 for the TA objective. As is typically the case, this bound turns out to be largely pessimistic and our computational results show that the algorithm performs much better in practice.

The 2–approximation guarantee holds also for the SP alignments we output. Recall that we include, among all the labelings considered, one

[1] The choice of the objective in the presence or absence of the tree can also be user-specified

in which the internal nodes of the tree are all labeled with any leaf S. For this particular labeling, the resulting tree is equivalent to a star centered at S, and as remarked before [9], the best star centered at a leaf gives a 2–approximation.

2 Procedure Overview

Our program is largely based on a heuristic procedure by Ravi and Kececioglu ([17]) for solving the tree alignment problem. Their algorithm relies on labeling the internal nodes with Steiner sequences for subsets of p leaves, where p is a parameter. The procedure is divided in two phases. In the first phase a Steiner sequence is computed for every subset of $q \leq p$ leaves, obtaining a set $\mathcal{F}$ of all such Steiner sequences. In the second phase, dynamic programming is used to compute the best labeling of the internal nodes among those in which only labels from $\mathcal{F}$ are allowed.

In this work, we have decided to solve the TA problem by employing Ravi and Kececioglu's algorithm, with the following variants: (i) Because computing exact Steiner sequences is expensive, we have limited the size of the subsets for which a Steiner problem is solved to $p = 3$. (ii) In addition to Sankoff's exact algorithm for Steiner sequences, with complexity $O(l^3)$, we also use a heuristic algorithm, with average (empirical) complexity $O(l^2)$. (iii) We do not necessarily compute the Steiner sequences for all the $\binom{n}{3}$ possible triples of leaves, but provide alternate, heuristic methods of sampling significant triples. (iv) We also perform a final re-optimization step, as introduced by Sankoff et al ([20]).

Our program can be used also to optimize SP. In this case, we first compute a tree having the given sequences for leaves and then assign tentative labels to the internal nodes by using Steiner sequences, as for the TA objective. In choosing the best label at each node, however, we use dynamic programming to minimize the total leaf–to–leaf distance in the tree, which is an upper bound on the final SP score. A final reoptimization phase can be run to improve the alignment.

The outline of our multiple alignment heuristic procedure is given below.

1. *Tree computation.*
   - TA: none (the tree is given).
   - SP: We compute a phylogenetic tree having the given sequences as leaves - this is derived from a MST on the sequence graph.

2. *Solution of Steiner problems.* We tentatively assign to each of the internal nodes of the phylogenetic tree a set of labels, given by the Steiner sequences of some subsets of the leaves.
3. *Optimal labeling by Dynamic Programming.* We find for each internal node the best sequence among those in its set of possible labels.
   - TA: The objective is to minimize the total tree-length.
   - SP: The objective is to minimize the total leaf–to-leaf distance in the tree.
4. *Local re-optimization.*
   - TA: At each node of degree three we replace the current sequence by the Steiner sequence of its neighbors. We iterate as long as there are improvements.
   - SP: (after step 5.) We iteratively break up the alignment into two subalignments that are then realigned optimally. The subalignments chosen have a large average difference in the current value versus the edit distance.
5. *Final alignment by Feng and Doolittle.* We compute a multiple alignment of all the resulting sequences (both leaves and internal nodes) by the progressive alignment method of Feng and Doolittle.

We elaborate on some of these steps next.

2.1 Tree Computation.

In order to derive a phylogenetic tree T relating a set of sequences when one is not input, we use a simple greedy approach. We start with T being a minimum cost spanning tree of the edit distance graph. Let (u, v) be the largest cost edge of T. Break up T by deleting edge (u, v) into two trees T_u containing u and T_v containing v. Recursively, apply the same procedure to T_u and T_v, obtaining two new trees, $T_{u'}$ and $T_{v'}$ rooted at new nodes u' and v' respectively. Finally, join these two subtrees by means of edges (u', w) and (v', w) to a new root node w, thus obtaining the final phylogenetic tree.

2.2 Solution of Steiner Problems.

Choice of Steiner Sequences Given a set of possible sequences (labels) for each internal node of the tree, choosing the best label is done by dynamic programming (described in 2.3) and is very fast in practice. On the other hand, computing the labels is very expensive. Therefore once some labels have been computed, it is convenient to store them at every internal node, i.e. all the nodes will have the same set $\mathcal{G}$ of labels. As previously noted, the labels allowed at the internal nodes will only be Steiner sequences for some subsets of $q \leq 3$ leaves. When $q = 1$ or 2, a Steiner sequence is simply a leaf, so that it will always be $\mathcal{G} = \mathcal{L} \cup \mathcal{G}'$, where $\mathcal{G}'$ is a set of Steiner sequences for some triples of leaves. Let us denote by $Y(S_i, S_j, S_k)$ a Steiner sequences for the triple (S_i, S_j, S_k). We allow three possibilities for $\mathcal{G}'$:

- $\mathcal{G}' = \emptyset$. In this case the internal nodes are labeled with leaves sequences only. This option results in the fastest running time, but may produce poor final alignments, especially when the given sequences are very dissimilar. Note that among the alignments based on these labels are included all lifted alignments [27] for TA. Similarly, these labels contain also all star alignments for SP.
- $\mathcal{G}' = \{Y(S_i, S_j, S_k) : i < j < k\}$. This is computationally the most expensive option, since it requires the solution of $\binom{n}{3}$ Steiner problems. On the other hand, the larger set of possible labels at the internal nodes guarantees a better value of the final alignment.
- Let $S_1, S_2, \ldots, S_n$ be the sequence of leaves as encountered by performing a depth–first visit of the tree. Then, $\mathcal{G}' = \{Y(S_j, S_k, S_h) : h = k + 1 = j + 2$ or $h = k + \Delta = j + 2\Delta\}$ where $\Delta = \lfloor \frac{n}{3} \rfloor$. The intention is to heuristically obtain a uniform sampling by selecting triples of leaves from different positions in an Euler tour of the tree. This option is quick –there are only $O(n)$ such triples– but ensures that each sequence is included in some triples, and that all the sequences are given the same representation in the samples.

Exact Steiner Sequences Assume we are interested in finding a Steiner sequence for three sequences U_1, U_2 and U_3. The dynamic programming procedure computes the optimal alignment of the variable Steiner sequence and U_1, U_2 and U_3. This is done backwards from the final column of the alignment, which will be of the form $(x_1, x_2, x_3, y)'$, where each x_i is either the last letter of the sequence U_i or a blank (but at least one x_i must be nonblank), and y is any nonblank letter of the alphabet Σ (representing the metter in the Steiner sequence being constructed). For any letter x, define $1 \cdot x = x$ and $0 \cdot x = -$. Let $B^+ = \{0, 1\}^3 \setminus (0, 0, 0)$ be the set of nonnull binary 3–vectors and let $V(l_1, l_2, l_3)$ be the cost of an optimal Steiner sequence for the the first l_1, l_2 and l_3 characters respectively of U_1, U_2 and U_3. The recursive dynamic programming relation is then

$$V(l_1, l_2, l_3) = \min_{b \in B^+} \left\{ V(l_1 - b_1, l_2 - b_2, l_3 - b_3) + \min_{y \in \Sigma} \sum_{i=1}^{3} c(b_i \cdot U_i[l_i], y) \right\}$$

The Steiner sequence is given, as customary in dynamic programming, by backtracking through the values $V(l_1, l_2, l_3)$ along the path for an optimal solution and listing the letters $\widehat{y} = \arg\min \sum_{i=1}^{3} c(b_i \cdot S_i[l_i], y)$ which achieve the minimum in the above expression. Note that the above recurrence requires time and space complexity of $O(7l^3)$, provided that for all $(x_1, x_2, x_3) \in \Sigma^3$, the values $C(x_1, x_2, x_3) := \min_{y \in \Sigma} \sum_{i=1}^{3} c(x_i, y)$ have been computed in a preliminary step and stored in a look-up table. In our implementation we have reduced the space complexity to $O(l^2)$ for the matrix $V(i, j, k)$ using ideas from [12].

Heuristic Steiner Sequences Computing exact Steiner sequences is very time consuming. For instance, the solution of a problem on sequences of about 200 letters each takes roughly half minute on a Pentium PC. Considering that for aligning 10 sequences we may have to solve $\binom{10}{3} = 120$ such problems, we see that speeding up the computation of Steiner sequences would be greatly beneficial. Therefore, we have devised an alternative, heuristic way of computing Steiner sequences which is extremely fast and turns out to be almost–optimal after extensive testing (see Sect. 3).

The idea is to first find all optimal alignments of two of the three sequences, say S_1 and S_2. They correspond to all the shortest paths from $(0, 0)$ to $(|S_1|, |S_2|)$ in the $|S_1| \times |S_2|$ dynamic programming lattice used for the pairwise alignment, and can be represented in a compact form as the subgraph of the lattice of all the edges on some optimal path. Note that this subgraph is typically much smaller than the whole lattice (empirically, $O(l)$ versus $O(l^2)$). Then, we perform a graph–to–sequence alignment, i.e. we find the best completion of an optimal alignment of S_1 and S_2 with S_3. In this case, "best" is taken with respect to the Steiner objective.

The value of the final solution may depend on the ordering of the sequences, since S_3 is clearly used differently than S_1 and S_2. We have observed in our experiments that choosing S_1 and S_2 to be the two closest sequences results in the best Steiner sequences over the three possible choices. However, since the algorithm is very fast, we compute all three possibilities of first aligning together two sequences and then versus the third, and return the best solution found. We conclude this section by remarking that the computation of heuristic Steiner sequences takes on the average one second for sequences of length 200, while returning a solution whose value was never more than 2% larger than the optimum in our extensive testing.

2.3 Optimal Labeling by Dynamic Programming.

In this section we consider the problem of optimally assigning a sequence from a given set $\mathcal{G}$ to each internal node of the tree. Denote by $w_1, \ldots, w_t$ the nodes which are immediate descendants of a node i. Let $V(i, S)$ be the optimal value for the subtree rooted at i when node i is labeled with a sequence $S \in \mathcal{G}$. We have the following dynamic programming recurrence:

$$V(i,S) = \begin{cases} 0 \text{ if } i \text{ is a leaf} \\ \min_{L_1,\ldots,L_t \in \mathcal{G}} \sum_{j=1}^{t} \left(\lambda(i,w_j) d(S,L_j) + V(w_j,L_j)\right) \text{ otherwise} \end{cases}$$

The coefficients $\lambda(i, w_j)$ allow us to distinguish between the two objective functions - TA and SP. For the TA objective, $V(i, S)$ represents the minimum total length of the subtree, among the labelings that assigns S to i. This is obtained by setting all the λ equal to 1. For the SP objective, we want to find the labels which minimize the total leaf–to–leaf distance. For any edge (u, v) of T, we set $\lambda(u, v)$ to be the number of pairs of leaves whose connecting path in the tree goes through (u, v). This value, called the load of the edge, is equal to $k(n - k)$, where k is the number of leaves on one shore of the cut identified by (u, v). By using the loads, the total leaf–to–leaf distance can be rewritten as $\sum_{S_i,S_j} d(S_i, S_j, T) = \sum_{(u,v)\in T} \lambda(u, v)d(L(u), L(v))$, where $L(u)$ and $L(v)$ are the sequences labeling nodes u and v.

Using the above relation, first the value of each label at each node is computed bottom–up, and later, proceeding top–down from the root, it is determined which label to pick at each node for obtaining an optimal solution. The overall complexity is $O(n|\mathcal{G}|^2)$, i.e. a very fast procedure.

2.4 Reoptimization

The reoptimization for TA objective is the same as in Sankoff et al [20]. For SP, however, we use a new approach. As in other works (e.g. [7]) we repeatedly break up the alignment into two pieces that are then realigned optimally via the basic dynamic program for edit distance. The new idea relies in how these alignments are chosen. Since for each pair of sequences in the same subalignment the distance remains the same, the only improvement can be for sequences that are in different subalignments. Let $\delta(S, S') = d_{\mathcal{A}}(S, S') - d(S, S')$. If $\mathcal{A}_1$ and $\mathcal{A}_2$ are the subalignments, $\delta(\mathcal{A}_1, \mathcal{A}_2) = \sum_{S \in \mathcal{A}_1, S' \in \mathcal{A}_2} \delta(S, S')$ is the δ–value of the cut $(\mathcal{A}_1, \mathcal{A}_2)$ in the graph of all sequences, and $\delta(\mathcal{A}_1, \mathcal{A}_2)/|\mathcal{A}_1||\mathcal{A}_2|$ is a per–sequence measure of how bad the alignment currently is versus the lower bound given by the edit distance. Hence we want to reoptimize some cuts of high (per–sequence) value, which we find through standard greedy heuristics. We have different settings on how far the reoptimization phase can be pushed. In the most expensive setting, for each pair (S, S') of sequences we find a large–value cut separating them and relign it. We iterate as long as there are improvements.

3 Computational Experiences

For our preliminary tests, we used two popular data sets. First, we obtained the sets of protein sequences of Mc Clure [16], used extensively to benchmark programs guided by the SP objective. For the Tree Alignment problem, we have used a famous instance by Sankoff et al [20], used as a benchmark in [10, 13].

As for the cost matrix, in our experiments we have used a distance matrix due to Taylor [23] for amino acid sequences, and the matrix in Sankoff [20] for DNA sequences. Our program also works with all the common score matrices (e.g. PAM, BLOSUM, etc).

1. **Lower Bounds.** A unique feature of the GESTALT suite is a procedure to generate linear programming (LP) based lower bounds on the TA and SP objective values of the given instance by using the Steiner sequences for triples computed so far. We describe the LP for the TA problem. We use a nonnegative variable for the length of every edge of the tree, and the objective is to minimize the sum of lengths of all tree edges. A distance of d between a pair of leaves S_i and S_j allows us to add the constraint that the sum of the values of the edge lengths on the path between S_i and S_j in the tree must be at least D. Similarly, given a value of $TA(i, j, k)$ for the minimum sum of the distances from an optimal Steiner sequence for the triple (S_i, S_j, S_k) to the three sequences S_i, S_j and S_k, we add the constraint that the sum of the lengths of all the edges in the tree induced by the three leaves S_i, S_j and S_k must be at least $TA(i, j, k)$. The objective function in the LP is to minimize the sum of the values of the edge variables. The set of constraints for distances between

Table 1. Heuristic vs exact Steiner sequences. Times in seconds, Pentium 133Mhz

instance	tot seqs	tot triples	relative error			time exact		time heuristic	
			avg	min	max	min	max	min	max
sank	9	84	0.003	0	0.02	15.8	41.0	0.6	1.9
mc582x6	6	20	0.004	0	0.01	52.3	75.6	0.5	3.0
mc586x6	6	20	0.007	0	0.017	17.8	42.5	0.6	2.1
mc587x6	6	20	0.01	0.003	0.019	29.2	71.9	0.8	2.7

pairs of leaves was experimented with in [10], while the strengthening to triples gives better bounds as reported below.

For the SP objective for multiple alignment, a simple averaging argument using the usage of Steiner triples yields a simple lower bound of $\sum_{i,j,k} SP(i,j,k)/(n-2)$ for n sequences, where $SP(i,j,k)$ denotes the optimal sum-of-pair value for the triple S_i, S_j and S_k. This may be further extended to a LP lower bound with one nonnegative variable for the distance between every pair of sequences in the multiple alignment. The constraints now require that for every triple S_i, S_j, S_k of distinct sequences, the sum of the values of the three variables involving the three pairs from this triple must be at least $SP(i,j,k)$. The objective is to minimize the sum of all the variables over all pairs of sequences.

2. **Steiner Sequences.** First, we determined the quality of heuristic vs exact Steiner sequences. The results are reported in Table 1. For these tests, we used four data sets, i.e. the sequences from Sankoff and three sets of sequences from McClure. These sequences have between one hundred and two hundred letters each. For each set, we have computed for each triple the exact and heuristic Steiner sequences, and compared the relative errors. It should be noted that on these sequences, the heuristic is roughly thirty times faster than the exact procedure, while the average error is less than one percent. A striking result was that in 41 out of 84 triples for the sank instance, the heuristic solution was in fact optimal.

3. **Tree Alignment.** A second experiment was performed to access the quality of the solution to the Tree Alignment problem, and the relative performance with different settings of the program. We have run GESTALT on Sankoff's problem with all possible combinations of user choices. The results are reported in Table 2. Again, it should be noted that using heuristic Steiner sequences is greatly beneficial to the computing time, and, since the whole procedure is heuristic in nature, can even

lead to better solutions than the exact option. This is indeed the case here.

In order to evaluate the quality of the results, we have computed the lower bound on the problem by using our LP module. The LP lower bound based on all the Steiner sequences of triples for the TA objective is 266.375 improving over the best bound of 253.5 previously known [10]. The optimal lifted alignment finds a value of 364, as also reported in [10]. Using heuristic Steiner sequences, we find a solution of value about 302 in about 7 minutes. Contrast this with the best upper bound of 295.5 by Sankoff et al. [20]. Our improved lower bound shows that Sankoff's solution is within 11% of optimal.

4. **Sum of Pairs.** For the SP objective, we report some results for the McClure data sets (Table 3). For each problem, we have computed the trivial lower bound given by the sum of edit distances, and two lower bounds based on the optimal SP alignment of triples of sequences - one uses a simple averaging argument (LB triples) and the other the solution to an LP relaxation (LB lp). We ran GESTALT with heuristic Steiner sequences, sampling all triples. Our solutions are in an interval of 2 to 9 percent from the lower bound. The table shows also the effectiveness of local reoptimization. For comparison, we also report the SP value of the star alignment (Gusfield, [9]).

Table 2. TA results on the instance `sank`. Times in seconds, Pentium PC

Triples	Steiner	Reopt	Value	Time
ALL	HEUR	EXACT	302	592
ALL	HEUR	HEUR	302.25	424
ALL	EXACT	EXACT	303.25	2802
SOME	EXACT	EXACT	304	493
ALL	EXACT	HEUR	304.25	2599
SOME	EXACT	HEUR	304.5	267
SOME	HEUR	EXACT	314	201
SOME	HEUR	HEUR	315.75	23
NONE	-	EXACT	320	152
NONE	-	HEUR	320.5	6
ALL	HEUR	NONE	322.25	298
ALL	EXACT	NONE	322.5	2387
SOME	EXACT	NONE	333.5	258
SOME	HEUR	NONE	333.75	15
NONE	-	NONE	364	1

Table 3. SP lower and upper bounds for McClure data sets

Instance	LB pairs	LB triples	LB lp	Star align.	GESTALT	Err %	GESTALT+reop	Err %
mc582x6	25411	26056	26100	28444	27647	0.06	26963	0.03
mc586x6	25191	25979	26029	29307	28605	0.10	27498	0.05
mc587x6	29914	30802	30864	34085	34152	0.11	32664	0.05
mc582x10	70718	72274	72757	82011	77676	0.07	75131	0.03
mc586x10	81745	84211	84662	99140	97725	0.15	91754	0.08
mc587x10	95002	97889	98349	115918	110463	0.12	105806	0.07
mc582x12	98810	100720	101464	113328	105674	0.04	103803	0.02
mc586x12	116889	120409	121130	143792	139398	0.15	131980	0.08
mc587x12	140679	145043	145804	174270	164883	0.13	160256	0.09

References

1. S. Altschul and D. Lipman, Trees, Stars and Multiple Sequence Alignment, *SIAM J. Appl. Math.* **49** (1989) 197–209
2. S. Altschul, D. Lipman and J. D. Kececioglu, A tool for multiple sequence alignment. *Proc. Natl. Acad. Sci. USA* **86** (1989) 4412–4415
3. V. Bafna, E. L. Lawler and P. Pevzner. Approximation Algorithms for Multiple Sequence Alignment. *Proceedings of the 5th Combinatorial Pattern Matching conference* LNCS **807** (1994) 43–53
4. H. Carrillo and D. Lipman. The multiple sequence alignment problem in biology. *SIAM J. Appl. Math.* **49:1** (1989) 197–209
5. S. C. Chan, A. K. C. Wong and D. K. Y. Chiu, "A survey of multiple sequence comparison methods," *Bull. Math. Biol.* **54** (1992) 563-598
6. D. Feng and R. Doolittle. Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *J. Molec. Evol.* **25** (1987) 351–360
7. O. Gotoh, Optimal alignment between groups of sequences and its application to multiple sequence alignment, *CABIOS* **9:3** (1993) 361–370
8. S. K. Gupta, J. Kececioglu, and A. A. Schaffer, Making the Shortest-Paths Approach to Sum-of-Pairs Multiple Sequence Alignment More Space Efficient in Practice, (extended abstract) *Proceedings of the 6th Combinatorial Pattern Matching conference* (1995)
9. D. Gusfield, Efficient methods for multiple sequence alignment with guaranteed error bounds, *Bulletin of Mathematical Biology* **55** (1993) 141–154
10. D. Gusfield and L. Wang, New Uses for Uniform Lifted Alignments, Submitted for publication (1996)
11. D. G. Higgins, A. J. Bleasby and R. Fuchs, Clustal V: Improved software for multiple sequence alignment, *CABIOS* **8** (1992) 189-191
12. D. Hirschberg, A linear space algorithm for computing maximal common subsequences, *Communications of the ACM* **18** (1975) 341–343
13. T. Jiang and F. Liu, Tree Alignment And Reconstruction application software, Version 1.0, February 1998. Available from `http://www.dcss.mcmaster.ca/~fliu`.
14. D. Lipman, S. Altschul and J. D. Kececioglu, A tool for multiple sequence alignment. *Proc. Natl. Acad. Sci. USA* **86** (1989) 4412–4415

15. S. B. Needleman and C. D. Wunsch. A general method applicable to search the similarities in the amino acid sequences of two proteins. *J. Mol. Biol.*, **48** (1970) 444
16. M. A. McClure, T. K. Vasi and W. M. Fitch. Comparative analysis of multiple protein–sequence alignment methods, *Mol. Biol. Evol.* **11** (1994) 571–592
17. R. Ravi and J. Kececioglu. Approximation algorithms for multiple sequence alignment under a fixed evolutionary tree, *Proceedings of the 6th Combinatorial Pattern Matching conference* (1995) 330–339
18. D. Sankoff, Minimal mutation trees of sequences, *SIAM J. Applied Math.* **28(1)** (1975) 35–42
19. D. Sankoff and R. Cedergren, Simultaneous comparison of three or more sequences related by a tree, in D. Sankoff and J. Kruskal editors, *Time warps, string edits and macromolecules: the theory and practice of sequence comparison*, Addison Wesley (1983) 253–264
20. D. Sankoff, R. Cedergren and G. Laplame, Frequency of insertion-deletion, transversion, and transition in the evolution of the 5s ribosomal rna, *J. Mol. Evol.* **7** (1976) 133-149
21. D. Sankoff, Analytical approaches to genomic evolution, *Biochimie* **75** (1993) 409–413
22. T. F. Smith and M. S. Waterman. Comparison of Biosequences. *Adv. Appl. Math.* (1981) 482–489
23. W. R. Taylor and D. T. Jones. Deriving an Amino Acid Distance Matrix, *J. Theor. Biol.* **164** (1993) 65–83
24. M. Vingron and P. Argos. A fast and sensitive multiple sequence alignment algorithm. *Comput. Appl. Biosci.* **5** (1989) 115–121
25. L. Wang and D. Gusfield. Improved Approximation Algorithms for Tree Alignment, *Proceedings of the 7th Combinatorial Pattern Matching conference* (1996) 220–233
26. L. Wang and T. Jiang. On the complexity of multiple sequence alignment, *J. Comp. Biol.* **1** (1994) 337–348
27. L. Wang, T. Jiang and E. L. Lawler. Aligning sequences via an evolutionary tree: complexity and approximation, *Algorithmica*, to appear. Also presented at the *26th ACM Symp. on Theory of Computing* (1994)