2008

TR-2008015: An Efficient Heuristic for the Tree Alignment Problem

Andrés Varón

Ward Wheeler

Amotz Bar-Noy

An Efficient Heuristic for the Tree Alignment Problem

Andrés Varón[1,2], Ward Wheeler[1], and Amotz Bar-Noy[2]

[1] Division of Invertebrate Zoology, American Museum of Natural History, Central Park West at 79th Street, New York, US.

[2] Computer Science Department, The Graduate School and University Center, The City University of New York, 365 Fifth Avenue, New York, US.

Abstract. Phylogeny and alignment estimation are two important and closely related biological problems. One approach to the alignment question is known in combinatorial optimization as the Tree Alignment Problem (TAP). A number of heuristic solutions exist, with the most competitive algorithms using iterative methods to find solutions. However, these methods are either non-scalable, the bound is too loose, or the time and memory complexity are difficult to predict. For the simplest distance functions, a widely used quick heuristic to find a solution to an instance of the problem is Direct Optimization (DO). Unfortunately, this algorithm has not been formally described, and it is currently unable to find correct solutions under an edition distance function called *affine indel cost*, of great importance for biologists analyzing DNA sequences. In this paper, we describe DO formally, and present a new algorithm, Affine-DO, which heuristically bounds an instance of the TAP under the affine indel cost. Affine-DO shows superior results with real biological sequences, a point that we illustrate using a published data set example.

Draft 8, December 18, 2008

1 Introduction

The inference of homologies between DNA sequences, that is, positions in multiple genomes that share a common evolutionary origin, is a crucial and yet difficult task that biologists face. Its computational counterpart is known as the multiple sequence alignment problem. There are various criteria and methods available to perform multiple sequence alignments (e.g. [34,18,13,7,8,23,6,47,20]). Among these, given a distance function, *minimize the overall cost of the alignment on a phylogenetic tree* [24,26,25,11,12,42] is one of the most important, and is known in combinatorial optimization as the Tree Alignment Problem (TAP). In reality, the TAP occurs as a subproblem of the Generalized Tree Alignment Problem (GTAP) [27], which looks for the tree with the lowest alignment cost among all possible trees [24]. The GTAP is equivalent to the Maximum Parsimony problem when the input sequences are not aligned, that is, when phylogeny and alignments are simultaneously inferred [24].

The computer program POY [45,36] is the most popular tool available for biologists interested in heuristically solving the GTAP, and implements a number of algorithms for the TAP grouped as follows: a strong version of the Lifted Assignment (also known as Fixed States) [40,43], Direct Optimization (DO) [42], and iterative improvement [26,46]. POY is known to produce competitive tree length estimations. As an example, the alignment for the Sankoff *et al.* data set [26] produced by POY has cost 302.25 under its default DO fast tree length estimation algorithm, matching that of GESTALT [16] and SALSA [15]. Using a weak iterative method on top of DO, POY finds an alignment of cost 298.75, very close to the best known cost of PRODALI (295.25) [30].

Prior to an analysis, biologists select a sequence edition distance function. One of its most important elements is the cost $G(k)$ of an indel of length k, a function that could have a significant impact in the overall analysis [3,17]. There are only four plausible indel cost functions in the literature: $G(k) = bk$ (non-affine) [41], $G(k) = a+bk$ (affine) [41], $G(k) = a+b\log k$ (logarithmic) [2,10,48,5,3],

Andrés Varón: avaron@amnh.org, pp. 1–14, 2008.

and $G(k) = a + bk + c \log k$ (affine-logarithmic) [3]. Simmulations and theoretical work has found evidence that affine-logarithmic yield the best results, but provide marginal benefits over the affine function, while its time complexity is much larger [3]. For this reason, many biologists continue to rely in the affine indel cost function, although the topic is still a subject of controversy.

For large data sets and the non-affine function, DO is the most important heuristic of practical use because it has the lowest time complexity while producing competitive alignments. All other (competitive) algorithms require a greater or difficult to predict time complexities with close practical approximation levels (e.g [16,15,30]). Unfortunately, DO performs poorly for most of the potential parameters of the affine distance: under many of them, the tree cost estimation of DO can even be unbounded, producing alignments of greater costs than those of the basic 2-approximation algorithm: (lifted assignment). Regardless this difficulty, there are no other practical algorithms for the GTAP that can be employed on datasets of the size that biologists are interested in (such as entire genomes), thus DO has remained the heuristic of choice when using affine gap costs (e.g. [4,1,33,31,21,14,28,32]).

1.1 Our Contribution

In this paper, we introduce and present experiments using our new algorithm, Affine-DO (Section 3). Affine-DO has the same time complexity of DO, but computes dramatically lower tree costs while maintaining a bounded time complexity, a feature that we experimentally show using its implementation in POY 4.0.2900 (Section 4). Our experiments show that for real datasets, Affine-DO coupled with iterative improvement strategies, produce solutions that are very close to the lower bound inferred from an LP solution. Moreover, iterating over a solution produced using Affine-DO has very little impact in the overall solution, a sign of the excellent performance of the heuristic.

Although we build Affine-DO on top of the successful aspects of DO, DO has never been formally described, nor have its basic properties been demonstrated. To describe Affine-DO, we first formally define DO and demonstrate some of its properties (Section 2).

1.2 Formal Definition and Related Work

Given a binary (phylogenetic) tree $T = (V, E)$ with leaf vertex set $L \subseteq V$, an assignment of sequences $\chi : L \to \Sigma^*$ for some alphabet Σ, and an edition distance function $e : \Sigma^* \times \Sigma^* \to \mathbb{R}$, the Tree Alignment Problem (TAP) [24] is to find an assignment of sequences $\chi' : V \to \Sigma^*$ such that for all $v \in L$, $\chi(v) = \chi'(v)$, and the total cost $\sum_{(u,v)\in E} e(\chi'(u), \chi'(v))$ is minimized. The TAP is known to be NP-Hard [38]. Due to its difficulty, a number of different methods are usually applied to produce reasonable (but most likely suboptimal) solutions.

The first heuristic techniques [26,25], consisted of iteratively improving the assignment of each interior vertex as a median between the sequences assigned to its three neighbors. This method can be applied to any initial assignment of sequences and adjust them to improve the overall tree cost.

Hein [11,12], designed a second heuristic solution which is implemented in the TreeAlign program. In TreeAlign, sets of sequences are represended by alignment graphs, which hold *all possible alignments* between a pair of sequences. The complete assignment can be performed in a post-order traversal of a rooted tree, beginning at the root, where each vertex is assigned an alignment graph of the two closest sequences in the alignment graph of its two children vertices. The final assignment can be performed using a backtrace over the pre-order traversal of the tree. Although this method

is powerful, it is not very scalable and it is difficult to predict its time complexity. Moreover, the current implementation of this heuristic does not allow the user to fully specify the distance function. This algorithm was later improved by Schwikowski and Vingron, producing some of the best alignments for the Sankoff data set [29].

In the most important theoretical results for the TAP, several 2-approximation algorithms, and a Polynomial Time Approximation Scheme (PTAS) were developed [40,37,22,39]. These algorithms solve the TAP from a theoretical perspective, but the execution time of the PTAS is of no practical use, while the the 2-approximation algorithms show very poor performance when compared to heuristic methods such as that of TreeAlign.

Direct Optimization (DO) [42] is a heuristic implemented in the computer program POY [45,44,36], which yields good execution time and competitive alignment costs. Given that DNA sequences have a small alphabet (4 elements extended with an indel to represent insertions and deletions), DO represents a large (potentially exponential) number of sequences in a compact way by using sets of bits. In the spirit of the TreeAlign method, DO heuristically assigns to each vertex, during a post order traversal, a set of sequences in an edition path connecting two of the closest sequences assigned in the children vertices. Later on, in a pre-order backtrace, a unique sequence is assigned to the interior vertices to produce an alignment and the instance solution.

Additionally, due to the small alphabet size of nucleotide sequences, DO can be implemented with a time complexity of $O(n^2|V|)$, where n is the length of the longest sequences, and V is the vertex set of T. For larger alphabets the total time complexity is $O(n^2|V||\Sigma|)$, where $\Sigma \ll n$ is the alphabet.

In recent work, Schwikowski and Vignon [30] published the best heuristic algorithm for the tree alignment problem, as implemented in the PRODALI software. Although powerful, this algorithm has a potentially exponential time and memory complexity, which in turn makes it non-scalable and difficult to use for the GTAP. Moreover, PRODALI is not publicly available. It is for these reasons that DO is the algorithm of choice for the GTAP, yielding slightly weaker tree cost approximations when compared to those of PRODALI, but delivering better performance for larger data sets.

2 Direct Optimization

In practice, biologists use DO due to its scalability and competitive costs. However, the DO algorithm was defined for the Non-Affine distance functions ($G(k) = bk$), and does not work correctly for the popular affine indel cost model [41] ($G(k) = a + bk$). Under many parameter sets, DO could produce worse tree cost estimations than those of the Lifted Assignment (non published data). We close this gap with our new algorithm which builds on top of the successfull aspects of DO.

Direct Optimization has only been described informally in the literature [42,44], and to build on it, we must first fill this gap. At the core of the algorithm is the use of an extended alphabet that support the representation of sets of sequences.

2.1 Sets of Sequences, Edition Distance, and Medians

Let Σ be an alphabet (typically $\Sigma = \{\text{A, C, G, T, } \textit{indel}\}$). The special "indel" element is used to represent insertions and deletions indistinctly, but must not occur in an instance solution. Let $d(x, y), x, y \in \Sigma$ be a metric distance between the elements in Σ. Let $P(\Sigma) = \mathscr{P}(\Sigma) \setminus \{\emptyset\}$ (i.e. all the subsets of Σ excepting the empty set), be an extended alphabet, to which we associate the corresponding distance function $d_P(A, B) = \min_{a \in A, b \in B} d(a, b)$.

Given d, the sequence edition distance between $X, Y \in \Sigma^*$[19], is defined as:

$$e(X_{1\ldots i}, Y_{1\ldots j}) = \min \begin{cases} e(X_{1\ldots i-1}, Y_{1\ldots j-1}) + d(X_i, Y_j) & \text{(case 1)} \\ e(X_{1\ldots i-1}, Y_{1\ldots j}) + d(X_i, indel) & \text{(case 2)} \\ e(X_{1\ldots i}, Y_{1\ldots j-1}) + d(Y_j, indel) & \text{(case 3)}, \end{cases} \quad (1)$$

with base cases $e(\langle\rangle, \langle\rangle) = 0$, and $e(\langle\rangle, X) = e(X, \langle\rangle) = \sum_{1 \le i \le |X|} d(X_i, indel)$. Similarly, we define e_P by replacing d with d_P in the expression.

The following observation is by definition:

Observation 1 *For all $A, B \in P(\Sigma)$, there exists an $a \in A$ and $b \in B$ such that $d_P(A, B) = d(a, b)$.*

To simplify notation, let $P(\Sigma)^* = (P(\Sigma))^*$. We will say that $X \in \Sigma^*$ is contained in $\mathscr{A} \in P(\Sigma)^*$ iff $|X| = |\mathscr{A}|$ and for all $x_i \in X$ and the corresponding $\mathscr{A}_i \in \mathscr{A}$, $x_i \in \mathscr{A}_i$. It follows that in this way we can represent a potentially exponential number of sequences:

Observation 2 *There are $\prod_{A \in \mathscr{A}} |A|$ sequences contained in $\mathscr{A}$.*

The following Lemma extends Observation 1 to the sequence edition distance:

Lemma 1. *For all $\mathscr{A}, \mathscr{B} \in P(\Sigma)^*$, there exists $U, V \in \Sigma^*$ such that U is contained in $\mathscr{A}$, V is contained in $\mathscr{B}$, and $e_P(\mathscr{A}, \mathscr{B}) = e(U, V)$.*

Proof. The proof is in the Appendix A.

Let the *median* between $A, B \in P(\Sigma)$ be $m(A, B) = \{x \in A \text{ and } y \in B, d_P(A, B) = d(x, y)$, and the corresponding median between a pair of aligned sequences $\mathscr{A}, \mathscr{B} \in P(\Sigma)^*, |\mathscr{A}| = |\mathscr{B}| = n$ as

$$m_P(\mathscr{A}, \mathscr{B}) = \langle m(\mathscr{A}_1, \mathscr{B}_1), m(\mathscr{A}_2, \mathscr{B}_2), \ldots, m(\mathscr{A}_n, \mathscr{B}_n)\rangle.$$

We will from now and on assume that for all $x \in \Sigma \setminus \{indel\}$, $d(x, indel) = b$ for some constant b.

Lemma 2. *Let $\mathscr{C} = m_P(\mathscr{A}, \mathscr{B})$. Then for all X contained in $\mathscr{C}$, there exists a Y contained in $\mathscr{A}$ and a Z contained in $\mathscr{B}$ such that $e_P(\mathscr{A}, \mathscr{B}) = e(Y, Z) = e(X, Y) + e(X, Z)$. Moreover, Y and Z are the closest sequences to $\mathscr{C}$ that are contained in $\mathscr{A}$ and $\mathscr{B}$ respectively.*

Proof. Follows directly from the median definition and Lemma 1. □

2.2 The DO Algorithm

DO (Algorithm 1) estimates the cost of a tree by proceeding in a post-order traversal on a *rooted tree*, starting at the root ρ.

Definition 1. *Two assignments $\chi : V \to \Sigma^*$ and $\chi' : V \to \Sigma^*$ are* compatible *if for all $v \in L$, $\chi(v) = \chi'(v)$.*

Theorem 1. *There exists an assignment of sequences χ' compatible with χ such that*

$$DO(T, \chi) \ge \sum_{(u,v) \in E} e(\chi'(u), \chi'(v)).$$

```
Data: A binary tree T with root ρ
Data: An assignment χ : L(T) → Σ of sequences the leaves L of T
Result: cost holds an upper bound of the cost of T, χ
begin
    cost ← 0;
    foreach level of T, with the bottom level first do
        foreach node v at the level do
            if v is a leaf (has no children) then
                S(v) ← ⟨a_i, a_i = {χ(v)_i}⟩;
            else
                Data: v has children u and w
                cost ← cost + e_P(S(u), S(w));
                 𝒰, 𝒲 ← the alignment of S(u) and S(w)) respectively;
                S(v) ← m_P(𝒰, 𝒲);
        return cost
end
```

Algorithm 1: $DO(T, \chi)$, Direct Optimization

Proof. Let T have root vertex ρ. We will call χ' the final assignment of sequences to the vertices of T. Select any X contained in $S(\rho)$ and set $\chi'(\rho) \leftarrow X$. Then for each other vertex v with parent p, following a pre-order traversal starting at ρ, let $\chi(v) \leftarrow X$ where $X \in \Sigma^*$ is contained in $S(v)$ and is closest to $\chi'(p)$. From Lemma 2, we know that for any selection at p there exists a selection in its children that would yield the additional cost computed at p during the DO algorithm. Moreover, at each pre-order traversal step, we assign to each vertex v the closest sequence to $\chi'(p)$ contained in $S(v)$. Again from Lemma 2, we know that the total cost of the two edges connecting p with its children must be greater than or equal to the additional cost computed for vertex p in the DO algorithm. Therefore, $DO(T, \chi) \geq \sum_{(u,v)\in E(T)} e(\chi'(u), \chi'(v))$. □

Lemma 1 and Observation 2 imply that $m_P(\mathscr{A}, \mathscr{B})$ contain a potentially exponential number of sequences in an optimal edition path transforming some of the closest sequences contained in $\mathscr{A}$ and $\mathscr{B}$. This implies that DO is weaker than the alignment graph algorithms [12,29,30], as these techniques hold either all the optimal edition paths or a set that includes the optimal edition paths. However, in these algorithms the overall execution time and memory consumption requirements could grow tremendously, in unpredictable ways that could be exponential [30]. In contrast, DO maintains a polynomial memory and execution time, making it more scalable. Moreover, DO can be efficiently implemented, yielding faster execution time, while providing good results (unpublished data).

3 The Affine Gap Cost Case

Lemma 2 does not hold for the affine gap cost, therefore, DO fails under this gap cost function (see appendix B for an example). To overcome this problem, we first extend Gotoh's algorithm [9] to compute heuristically distances for sequences in $P(\Sigma)^*$, and define a new median sequence. With these tools, we modify DO so that Lemma 2 still holds to compute tree cost bounds.

3.1 Heuristic Pairwise Sequence Alignment in $P(\Sigma)^*$

Let $\mathscr{A}, \mathscr{B} \in P(\Sigma)^*$ be a pair of sequences to be aligned. We define

$$e_{\mathrm{aff}_P}(\mathscr{A}_{1\ldots i}, \mathscr{B}_{1\ldots j}) = \min\{G[i,j], D[i,j], V[i,j], H[i,j]\}, \tag{2}$$

as the cost of aligning the subsequences $\mathscr{A}_{1\ldots i}$ and $\mathscr{B}_{1\ldots j}$. G, D, V, and H are matrices to be filled recursively. Before defining them formally, let us explain the basic intuition of each. G holds the cost of an alignment where $\mathscr{A}_i$ and $\mathscr{B}_j$ align elements other than an indel. D holds at each position i, j the cost of an alignment using indel elements in $\mathscr{A}_i$ and $\mathscr{B}_j$. V holds the cost of an alignment where we use a "vertical" indel block by aligning $\mathscr{B}_j$ with an indel. Finally, H holds the cost of an alignment where we use a "horizontal" indel block by aligning $\mathscr{A}_i$ with an indel.

To compute these values, we need a number of accessory functions. We first define the cost of a pure substitution $subst(A, B) = d_P(A \setminus \{indel\}, B \setminus \{indel\})$. Symmetric to the substitution cost, we need the cost of *extending* a gap when $indel \in A, B \subseteq \Sigma$:

$$diag(A, B) = \begin{cases} 0 & \text{if } indel \in A \text{ and } indel \in B \\ \infty & \text{otherwise.} \end{cases}$$

We have three remaining accessory functions required to compute the matrices G, H, V, and D, all of them handling different cases of where a or b needs to be added. The first function, $go(\mathscr{A}, i)$ evaluates whether or not it is necessary to add a gap opening value when aligning $\mathscr{A}_i$ with a gap:

$$go(\mathscr{A}, i) = \begin{cases} 0 & \text{if } i = 1 \text{ and } indel \in \mathscr{A}_i \\ 0 & \text{if } i > 1 \text{ and } indel \notin \mathscr{A}_{i-1} \text{ and } indel \in \mathscr{A}_i \\ a & \text{otherwise.} \end{cases}$$

The second function $go'(A, B)$ calculates the extra cost incurred when *not* selecting an indel in one of the sequences means splitting an indel block:

$$go'(A, B) = subst(A, B) + \begin{cases} 0 & \text{if } indel \notin A \\ a & \text{otherwise.} \end{cases}$$

The third, and final accessory function, computes what would be the extra cost of *extending* an indel, that is:

$$ge(A) = \begin{cases} 0 & \text{if } indel \in A \\ b & \text{otherwise.} \end{cases}$$

We are ready to define the recursive functions for the cost matrices:

$$G[i, j] = \min \begin{cases} G[i-1, j-1] + subst(\mathscr{A}_i, \mathscr{B}_j) \\ D[i-1, j-1] + subst(\mathscr{A}_i, \mathscr{B}_j) + go(\mathscr{A}, i) + go(\mathscr{B}, j) \\ V[i-1, j-1] + go'(\mathscr{B}_j, \mathscr{A}_i) \\ H[i-1, j-1] + go'(\mathscr{A}_i, \mathscr{B}_j), \end{cases} \tag{3}$$

$$H[i, j] = \min \begin{cases} H[i, j-1] + ge(\mathscr{B}_j) \\ D[i, j-1] + ge(\mathscr{B}_j) + go(\mathscr{B}, j), \end{cases} \tag{4}$$

$$V[i, j] = \min \begin{cases} V[i-1, j] + ge(\mathscr{A}_i) \\ D[i-1, j] + ge(\mathscr{A}_i) + go(\mathscr{A}, i), \end{cases} \tag{5}$$

$$D[i, j] = diag(\mathscr{A}_i, \mathscr{B}_i) + \min \begin{cases} D[i-1, j-1] \\ G[i-1, j-1] + go(\mathscr{A}, i) + go(\mathscr{B}, j), \end{cases} \tag{6}$$

with base cases $G[0,0] = 0$, $D[0,0] = \infty$, $V[0,0] = go(\mathscr{A}, 1)$, $H[0,0] = go(\mathscr{B}, 1)$, $G = [0,i] = D[0,i] = V[0,i] = \infty, H[0,i] = H[0,i-1] + ge(\mathscr{B}_i)$, $1 \leq i \leq |\mathscr{B}|$, $V[j,0] = V[j-1,0] + ge(\mathscr{A}_j)$, and $G[j,0] = D[j,0] = H[j,0] = \infty, 1 \leq j \leq |\mathscr{A}|$.

Theorem 2. *There exists a sequence X contained in $\mathscr{A}$ and a sequence Y contained in $\mathscr{B}$ such that $e_{aff_P}(\mathscr{A}, \mathscr{B}) \geq e_{aff}(X, Y)$.*

Proof. We are going to create a pair of sequences contained in $\mathscr{A}$ and $\mathscr{B}$ that have edition cost at most $e_{\text{aff}_P}(\mathscr{A}, \mathscr{B})$. To do so, we follow the backtrace that yields $e_{\text{aff}_P}(\mathscr{A}, \mathscr{B})$, and at each position i and j in the aligned $\mathscr{A}$ and $\mathscr{B}$ we assign X_k and Y_k, where k is the alignment position corresponding to the aligned X_i and Y_j as follows:

1. The matrix G holds on each cell i, j the cost of aligning $\mathscr{A}_{1\ldots i}$ and $\mathscr{B}_{1\ldots j}$ when a non-indel element of $\mathscr{A}_i$ and $\mathscr{B}_j$ is aligned. If the backtrace uses $G[i,j]$ then assign to X_i and Y_j the closest elements in $\mathscr{A}_i \setminus indel$ and $\mathscr{B}_j \setminus indel$. Observe that all the cases in Equation 3 align a non-indel element from $\mathscr{A}_i$ and $\mathscr{B}_j$, and add a cost that is always greater than or equal to $subst(\mathscr{A}_i, \mathscr{B}_j) = d(X_i, Y_j)$.
2. The matrix H holds on each cell the cost of extending a indel in the horizontal direction. Therefore, select $X_k = indel$, and

$$Y_k = \begin{cases} indel & \text{if } indel \in \mathscr{B}_j \\ y, y \in \mathscr{B}_j & \text{otherwise.} \end{cases}$$

   If $Y_k = indel$, then the alignment of X_k and Y_k causes no additional cost in the particular alignment being built between X and Y. Otherwise, then there is an extra cost, of at least the b parameter, which both cases of Equation 4 account for. Additionally, if the previous pair of aligned elements are a pair of indels (second case in 4, see below for the treatment of this option), then an extra indel opening cost is added.
3. Similarly, the matrix V holds on each cell the cost of extending a indel block in the vertical direction. The treatment is symmetric to that of H, with $Y_k = indel$ and

$$X_k = \begin{cases} indel & \text{if } indel \in \mathscr{A}_i \\ x, x \in \mathscr{A}_i & \text{otherwise.} \end{cases}$$

4. The matrix D holds the cost of extending a indel in the *diagonal direction*, that is, when both $\mathscr{A}$ and $\mathscr{B}$ hold indels in their corresponding aligned elements, and those indels are being selected to generate the alignment of $\mathscr{A}$ and $\mathscr{B}$. Therefore, Equation 6 ensures that this choice is only possible by assigning ∞ whenever at least one of $\mathscr{A}_i$ or $\mathscr{B}_j$ does not contain a indel. Otherwise, if this option is selected, then simply assign *indel* to both X_k and Y_k with no extra cost for the alignment of X and Y. □

Theorem 2 tells us that if we align a pair of sequences in $\mathscr{A}, \mathscr{B}$ then we can bound the cost of the closest pair of sequences contained in them.

3.2 The Main Algorithm: Affine-DO

We will now use $e_{\text{aff}_P}(\mathscr{A}, \mathscr{B})$ to bound the cost of a tree using a post-order traversal, in the same way we did with DO (Algorithm 1). In order to do so, we must first define a valid median to be

assigned on each step (that is the function m_P in Algorithm 1). To create the median $\mathscr{M}$ at each position k during the backtrace, we use the method depicted in the proof of Theorem 2, with the following changes:

1. If we selected two indels in X_k and Y_k, $\mathscr{M}'_k = \{indel\}$.
2. If $X_k = indel$ and $Y_k \neq indel$, then $\mathscr{M}'_k = \{indel\} + \mathscr{B}_j$.
3. If $X_k \neq indel$ and $Y_k = indel$, then $\mathscr{M}'_k = \{indel\} + \mathscr{A}_i$.
4. If $X_k \neq indel$ and $Y_k \neq indel$, then $\mathscr{M}'_k = \{x \in \mathscr{A}_i, \text{for some } y \in \mathscr{B}_j, d(x,y) = d(X_k, Y_k)\} + \{y \in \mathscr{B}_j, \text{for some } x \in \mathscr{A}_i, d(x,y) = d(X_k, Y_k)\}$.
5. Once the complete $\mathscr{M}'$ is created, remove all the elements $\mathscr{M}_i = indel$ to create the indel-less sequence $\mathscr{M}$. We call $\mathscr{M}$ the affine median $m_{\text{aff}_P}(A, B)$.

Definition 2. *Affine-DO is Algorithm 1, modified by replacing m_P with m_{aff_P}, and e_P with e_{aff_P}.*

We will now show that the Affine-DO algorithm can be used to heuristically bound the cost of an instance of the TAP.

Theorem 3. *Given a rooted tree T with root ρ, and an Affine-DO assignment $S : V(T) \to P(\Sigma)^*$, there exists a assignment $\chi' : V(T) \to \Sigma^*$ such that $X = \chi'(\rho)$ and the cost computed by Affine-DO equals that implied by χ'.*

Proof. The proof is in the Appendix C.

Theorem 4. *If Σ is small, then Affine-DO has time complexity $O(n^2|V|)$ time, otherwise the time complexity is $O(n^2|V||\Sigma|)$.*

Proof. If the alphabet is small, then m_{aff_P} and d_P can be pre-computed in a lookup table with a bit set representation for constant time comparison of the sets. Otherwise, a binary tree representation of the sets would be necessary, adding a $|\Sigma|$ factor to the set comparison. Each heuristic alignment can be performed using dynamic programming, with time complexity $O(n^2)$ where n is the maximum sequence length (Ukkonen's [35] algorithm makes no obvious improvement as insertions and deletions could have cost 0 when aligning sequences in $P(\Sigma)^*$) . Each alignment must be repeated for $|V|$ vertices during the post-order traversal, yielding the claimed time complexity. □

4 Experimental Evaluation

POY implements a number of algorithms to approximate the tree alignment problem. These can be classified in two groups: initial assignment, and iterative improvement.

Initial Assignment includes the Lifted Assignment [40,43] (also known as Fixed States), Direct Optimization [42], and Affine-DO. Each of these algorithms starts with a function χ and creates a χ' compatible with χ which is an instance solution. DO and Affine-DO have already been described. The strong version of the Lifted Assignment is a simple algorithm where the interior vertices are optimally assigned one of the sequences assigned to the leaves of the tree. This solution turns out to be a 2-approximation [40].

Iterative Improvement modifies an existant χ' by readjusting each interior vertex using its three neighbors. This procedure is repeated iteratively, until a (user provided) maximum number of iterations is reached, or no further tree cost improvements can be achieved. The adjustment itself can be done using an *approximated* or an *exact* three dimensional alignment. The approximated assignment uses DO or Affine-DO (the selection depends on which kind of edition distance function is used) to solve the tree alignment problem on the three possible binary rooted trees formed by the three neighbors of the vertex as leaves. The assignment yielding the best cost is selected as the new center. (A graphical depiction can be found in Appendix D.)

In this paper we evaluated the following algorithm combinations: lifted assignment (LA), lifted assignment followed by iterative approximated (LA + IA), affine-DO (ADO), affine-DO followed by iterative approximated (ADO + IA), and affine-DO followed by iterative exact until no better cost was found (ADO + IE). We did not include LA + IE because the execution time is prohibitive (within a 3 day wall time the execution would not finish for most trees). Moreover, the few experiments that we could perform using LA + IE showed that the cost estimation was always higher than that of Affine-DO. To evaluate the closeness of our algorithm to the optimal solution we used an LP instance, with constraints given by the pairwise distance between the input sequences, and one variable per edge in each input tree.

As the subject data set, we used a set of plant genes recently published and analyzed in the literature [28]. We generated a set of 100 trees using the command *build*(100) to create a sample of different but sound phylogenetic trees, for 5 different affine distance functions; each tree was then evaluated using every algorithm combination. For all the distance functions selected in this experiment, DO yields costs which are much higher than those of LA. The results are graphically presented in figure 1.

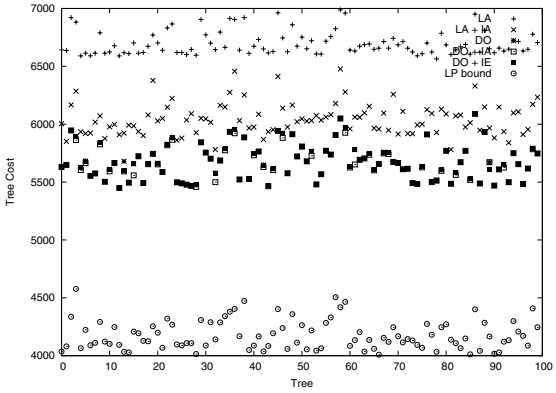
As expected, the best results are found using ADO+IE. However, the improvement in cost over ADO without improvement is minimal, with a tremendous extra time complexity. Surprisingly, ADO+IA remains competitive, outperforming ADO+IE in some cases, as a better solution can be found in a suboptimal path. The cost of ADO+IE is between 15% and 20% lower than the cost of LA (with the same time complexity). When compared with the LP lower bound, ADO is roughtly within 30% and 50% of the (most likely non-realizable) optimal solution.

5 Future Work

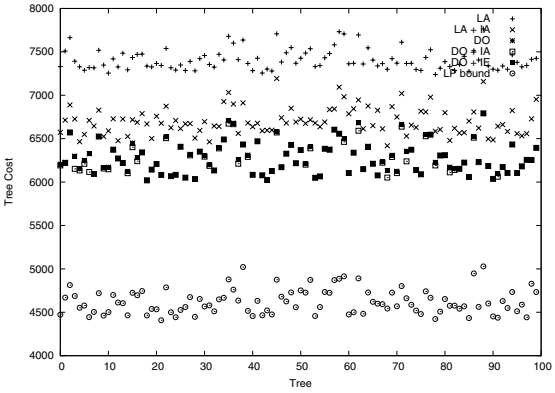
We have presented a new heuristic algorithm that shows excellent experimental results for the TAP. This algorithm is well suited for the GTAP under affine sequence edition distances, and yields significatively better results when complemented with iterative methods. The main question that remains is whether or not there exists a guaranteed bound for DO or Affine-DO, and if the answer is positive, whether or not it is possible to improve the PTAS using these ideas.

6 Acknowledgements

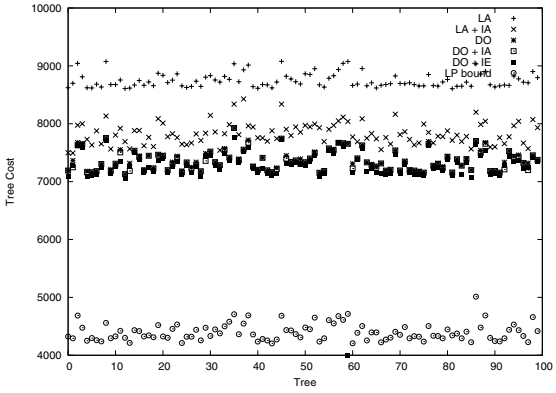
Andrés Varón and Ward C. Wheeler where supported by the U.S. Army Research Laboratory and the U.S. Army Research Office [W911NF-05-1-0271], and the NSF-ITR grant "Building the tree of life: A national resource for phyloinformatics and computational phylogenetics" [NSF EF 03-31495]. Ward Wheeler was also supported by the National Science Foundation grants "An Integrated approach to the origina and diversification of protostomes" [NSF DEB 05-31677], and "Assembling the tree of life: phylogeny of spiders" [NSF EAR 02-28699].



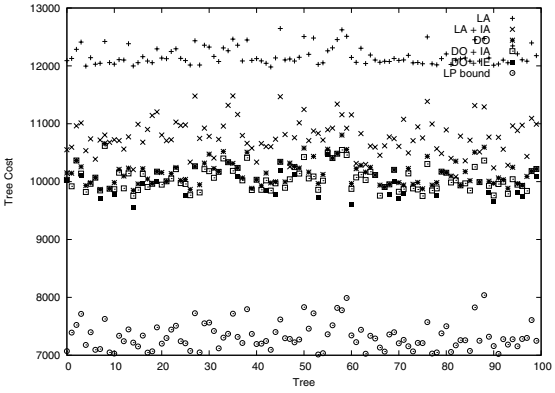
(a) substitutions = 2, $a = 2$, $b = 1$



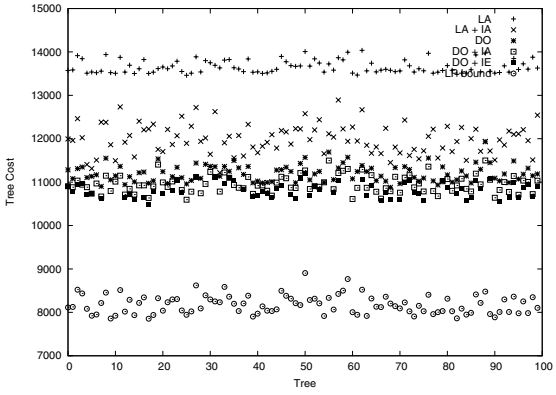
(b) substitutions = 2, $a = 4$, $b = 1$



(c) substitutions = 3, $a = 3$, $b = 1$



(d) substitutions = 5, $a = 5$, $b = 1$



(e) substitutions = 6, $a = 6$, $b = 1$

Fig. 1. Algorithm performance for 6 different edition distance parameters. In every case, the best cost is achieved by Affine-DO + IE. Notice that in many instances the iterative method could not improve the cost found using Affine-DO (almost none for substitutions = 2, $a = 2$, and $b = 1$.

References

1. L Aagesen, G Petersen, and O Seberg. Sequence length variation, indel costs, and congruence in sensitivity analysis. *Cladistics*, 21:15–30, 2005.
2. S A Benner and M A Cohen. Empirical and structural models for insertions and deletions in the divergent evolution of proteins. *Journal of Molecular Evolution*, 229:1065–1082, 1993.
3. R A Cartwright. Logarithmic gap costs decrease alignment accuracy. *BMC Bioinformatics*, 7:527–539, 2006.
4. M S Caterino and A P Vogler. The phylogeny of the histeroidea (coleoptera: Staphyliniformia). *Cladistics*, 18:394–415, 2002.
5. M S S Chang and S A Benner. Empirical analysis of protein insertions and deletions determining parameters for the correct placement of gaps in protein sequence alignments. *Journal of Molecular Biology*, 341(2):617–631, 2004.
6. C B Do, M S P Mahabhashyam, M Brudno, and S Batzoglou. ProbCons: Probabilistic consistency-based multiple sequence alignment. *Genome Res.*, 15:330–340, 2005.
7. R C Edgar. MUSCLE: a multiple sequence alignment method with reduced time and space complexity. *BMC Bioinformatics*, 5:113, 2004.
8. R Fleissner, D Metzler, and A von Haeseler. Simultaneous statistical multiple alignment and phylogeny reconstruction. *Systematic Biology*, 54(4):548–561, 2005.
9. O Gotoh. An improved algorithm for matching biological sequences. *Journal of Molecular Biology*, 162:705 – 708, 1982.
10. Xun Gu and Wen-Hsiung Li. The size distribution of insertions and deletions in human and rodent pseudogenes suggests the logarithmic gap penalty for sequence alignment. *Journal of Molecular Evolution*, 40(4):464–473, 1995.
11. H Hein. A new method that simultaneously aligns and reconstructs ancestral sequences for any number of homologous sequences, when the phylogeny is given. *Molecular Biology and Evolution*, 6(6):649–668, November 1989.
12. J Hein. Unified approach to alignment and phylogenies. *Methods in Enzymology*, 183, 1990.
13. K Katoh, K Misawa, K Kuma, and T Miyata. MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform. *Nucl. Acids Res.*, 30:3059–3066, 2002.
14. S N Kutty, M V Bernasconi, F Šifner, and R Meier. Sensitivity analysis, molecular systematics and natural history evolution of scathophagidae (diptera: Cyclorrhapha: Calyptratae). *Cladistics*, 23:64–83, 2007.
15. G Lancia and R Ravi. Salsa: Sequence alignment via steiner ancestors.
16. G Lancia and R Ravi. Gestalt: Genomic steiner alignments. *Lecture Notes in Computer Science*, 1645:101, 1999.
17. K Liu, S Nelesen, S Raghavan, C Randal Linder, and T Warnow. Barking up the wrong treelength: the impact of gap penalty on alignment and tree accuracy. *IEEE Transactions on Computational Biology and Bioinformatics*, 2008.
18. B Morgenstern. DIALIGN 2: improvement of the segment-to-segment approach to multiple sequence alignment. *Bioinformatics*, 15:211–218, 1999.
19. 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.
20. S Nelesen, K Liu, D Zhao, C R Linder, and T Warnow. The effect of the guide tree on multiple sequence alignments and subsequenct phylogenetic analyses. *Pacific Symposium on Biocomputing*, 13:25–36, 2008.
21. J Pons and A P Vogled. Size, frequency, and phylogenetic signal of multiple-residue indels in sequence alignment of introns. *Cladistics*, 22:144–156, 2006.
22. R Ravi and J D Kececioglu. Approximation algorithms for multiple sequence alignment under a fixed evolutionary tree. *Discret. Appl. Math.*, 88:355–366, 1998.
23. Benjamin D. Redelings and Marc A. Suchard. Joint Bayesian estimation of alignment and phylogeny. *Syst. Biol.*, 54:401–418, 2005.
24. D Sankoff. Minimal mutation trees of sequences. *SIAM Journal on Applied Mathematics*, 28(1):35–42, January 1975.
25. D Sankoff and R J Cedergren. *Simultaneous Comparison of Three or more Sequences Related by a Tree*, pages 253–263. Addison-Wesley, Reading, MA, 1983.
26. D Sankoff, R J Cedergren, and G Lapalme. Frequency of insertion-deletion, transversion, and transition in the evolution of 5s ribosomal rna. *Journal of Molecular Evolution*, 7:133–149, 1976.
27. D Sankoff and P Rousseau. Locating the vertices of a steiner tree in an arbitrary space. *Mathematical Programming*, 9:240–246, 1975.

28. A C Scheen, C Lindqvist, C G Fossdal, and V A Albert. Molecular phylogenetics of tribe synandraeae, a north american lineage of lamioid mints (lamiaceae). *Cladistics*, 24:299–314, 2008.
29. B Schwikowski and M Vingron. The deferred path heuristic for the generalized tree alignment problem. In *RECOMB '97: Proceedings of the first annual international conference on Computational molecular biology*, pages 257–266, New York, NY, USA, 1997. ACM Press.
30. B Schwikowski and M Vingron. Weighted sequence graphs: boosting iterated dynamic programming using locally suboptimal solutions. *Discrete Appl. Math.*, 127(1):95–117, 2003.
31. M J Sharkey, N M Laurenne, B Sharanowski, D L J Quicke, and D Murray. Revision of the agathidinae (hymenoptera: Braconidae) with comparisons of static and dynamic alignments. *Cladistics*, 22:546–567, 2006.
32. J C Spagna and F Álvarez-Padilla. Finding an upper limit for gap costs in direct optimization. *Cladistics*, 24:1–15, 2008.
33. M D Terry and M F Whiting. Comparison of two alignment techniques within a single complex data set: Poy versus clustal. *Cladistics*, 21:272–281, 2005.
34. J D Thompson, D G Higgins, and T J Gibson. CLUSTAL W: Improving the sensitivity of progressive multiple sequence alignment through sequence weighting, positions–specific gap penalties and weight matrix choice. *Nucleic Acids Res.*, 22:4673–4680, 1994.
35. E Ukkonen. Algorithms for approximate string matching. *Inf. Control*, 64(1-3):100–118, 1985.
36. A Varón, L S Vinh, I Bomash, and W C Wheeler. Poy 4.0.2870. http://research.amnh.org/scicomp/, 2008.
37. L Wang and D Gusfield. Impoved approximation algorithms for tree alignment. *Journal of Algorithms*, 25(2):255–273, November 1997.
38. L Wang and T Jiang. On the complexity of multiple sequence alignment. *Journal of Computational Biology*, 1:337–348, 1994.
39. L Wang, T Jiang, and D Gusfield. A more efficient approximation scheme for tree alignment. *SIAM Journal on Computing*, 30(1):283–299, 2000.
40. L Wang, T Jiang, and E L Lawler. Approximation algorithms for tree alignment with a given phylogeny. *Algorithmica*, 16:302–315, 1996.
41. M S Waterman, T F Smith, and W A Beyer. Some biological sequence metrics. *Advances in Mathematics*, 20(3):367–387, 1976.
42. W Wheeler. Optimization alignment: The end of multiple sequence alignment in phylogenetics? *Cladistics*, 12:1–9, 1996.
43. W Wheeler. Fixed character states and the optimization of molecular sequence data. *Cladistics*, 15:379 – 385, 1999.
44. W Wheeler, L Aagensen, C P Arango, J Faivovich, T Grant, C D'Haese, D Janies, W L Smith, A Varón, and G Giribet. *Dynamic Homology and Phylogenetic Systematics: A Unified Approach using POY*. American Museum of Natural History, 2006.
45. W Wheeler, D Gladstein, and J De-Laet. *POY, Phylogeny Reconstruction via Optimization of DNA and other Data version 3.0.11 (May 6 of 2003)*. American Museum of Natural History, May 2003.
46. W C Wheeler. Iterative pass optimization of sequence data. *Cladistics*, 19:254–260, 2003.
47. W C Wheeler. Dynamic homology and the likelihood criterion. *Cladistics*, 22:157–170, 2006.
48. Z Zhang and M Gerstein. Patterns of nucleotide substitution, insertion and deletion in the human genome inferred from pseudogenes. *Nucl. Acids Res.*, 31(18):5338–5348, 2003.

A Proof of Lemma 1

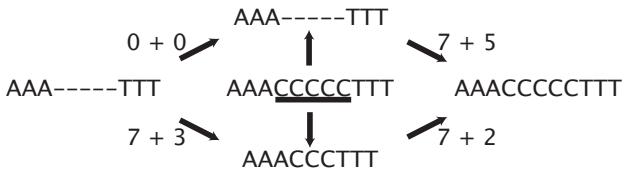
Proof. We will define a procedure to produce U and V. Start with an empty U and V, and follow the backtrace of Equation 1. For each case, prepend the following to U and V:

case 1 Select an element $x \in X_i$ that holds Observation 1 and prepend it to U. Then find an element $y \in Y_k$ that is closest to x and prepend it to V. From Observation 1 we know that $d(x, y) = d_P(X_i, Y_j)$.

case 2 Select an element $x \in X_i$ closest to *indel* and prepend x to U and *indel* to V. Again from Observation 1 we know that $d(x, indel) = d_P(X_i, \{indel\})$.

case 3 Symmetric to case 2. □

B Example of Lemma 2 not holding when $G(k) = a + bk$



Let $G(k) = 7 + k$. The center sequence is the median for the alignment of the left and right sequences. (The underscored C represents {C, *indel*}.) Although the upper and lower sequences are contained in the median, the lower one is not in an optimal edition path connecting left and right. This example shows Lemma 2 does not hold for affine gap costs.

C Proof of Theorem 3

Proof. If there are no indels involved in the tree alignment, then the arguments of Theorem 1 would suffice. We should therefore concentrate in the cases that involve indels.

To prove those remaining cases, we will use induction on the vertices of the tree. To do so, we will count the *credits* that each vertex adds to the subtree it roots as added by the Affine-DO algorithm. The credits represent the maximum total cost of the indels involved in a particular subtree; we will compare them with the *debits* incurred by a set of indels, and verify that the *credits* are always greater than or equal to the *debits*. To simplify the description, we will call type A subsequences of maximal size holding only indels, and type B subsequences of maximal size holding sets that include, but are not limited to, indels, and type C maximal subsequences holding sets with no indel. We will count without loss of generality the *credits* and *debits* within those subsequences. In figures 2 and 3, Type A is represented as a line, type B as a box with a center line, and type C as an empty box.

For the inductive step, consider the leaves of the tree. By definition, for all $v \in L$, $S(v)$ cannot contain subsequences of type A nor B, as there are no indels allowed. Therefore, the theorem holds true, with a $credits = debits = 0$.

Consider now the interior vertex v, with children u and v. In figure 2 all the simple cases where the limits of the subsequences in $S(u)$ and $S(v)$ match those of $S(p)$. It is easy to see that in all those cases $credits = debits$.

Consider now the more difficult case when the blocks do not have exact limits. Assume without loss of generality that $S(u)$ and $S(v)$ have a segment of type B, and $S(p)$ has in the corresponding segment a series of blocks of type A and C (figure 3). (There can be no subsequences of type B in $S(p)$ aligned with those of type B in $S(u)$ and $S(v)$ as m_{aff_P} does not allow it.)

The total credit granted by Equation 3 is $c \geq 2ma + 2b(s_1 + s_2 + \cdots + s_{m-1} + s_m)$. We can transfer $c/2$ to u (v), so that in one edge rooted by u (v), a series of insertions corresponding to the subsequences $s_1, s_2, \ldots, s_m$ can occur (figure 3.b, solid boxes), while the other branch supports a single deletion of length $l - (s_1 + s_2 + \cdots + s_m)$ (figure 3.b, upper dashed box). The total debit of these events now rooted in u would be

$$ma + b(s_1 + s_2 + \cdots + s_{m-1} + s_m) + a + b(l - (s_1 + s_2 + \cdots + s_m)) \leq c/2 + a + bl. \qquad (7)$$

By the inductive hypothesis, the subtree rooted by u (v) has $credits \geq debits$, and from Equation 7 we also have that $credits > debits$ in p, therefore the theorem holds, and the overall tree rooted by p has a sequence assignment of cost at most that computed by the Affine-DO algorithm. □

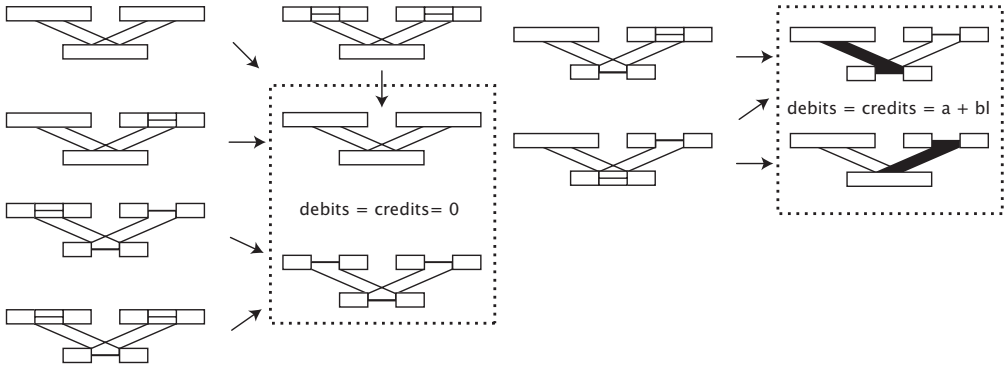


Fig. 2. *credits* and *debits* incurred by the different possible arrangements of subsegments with matching limits in $S(p)$, $S(u)$, and $S(v)$. The only cases with $credits = debits > 0$ (in the right box) represents with filled boxes the assignments that would yield an indel block.

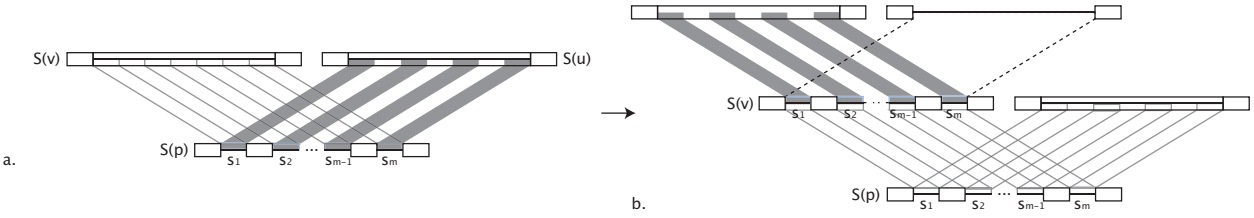
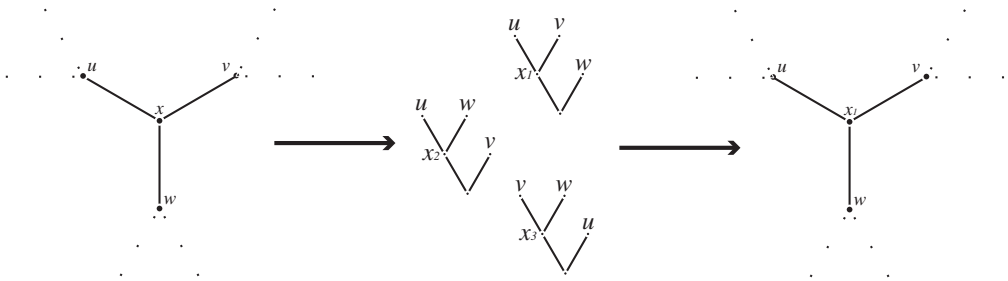


Fig. 3. a. Overlapping blocks of type B in $S(u)$ and $S(v)$, with a complex pattern of insertions and deletions in $S(p)$. The total *credits* added at $S(p)$ by Affine-DO can be transferred to u and v. b. The credits transferred to v can be assigned to m individual insertion blocks (solid boxes), and one deletion block (dashed empty box) which maintain $debits > credits$.

D Graphical Represenation of the Approximated Iterative Algorithm



An iteration of the approximated iterative improvement. To improve x, Affine-DO is used to produce x_1, x_2, and x_3 in the three possible rooted trees with leaves u, v, and w. If the best assignment x_1 yields better cost than the original x, then it is replaced, otherwise no change is made.