Hindawi Publishing Corporation
BioMed Research International
Volume 2013, Article ID 472163, 10 pages
http://dx.doi.org/10.1155/2013/472163

Research Article

Algorithms of Ancestral Gene Length Reconstruction

Alexander Bolshoy[1,2] and Valery M. Kirzhner[3]

[1] *Department of Evolutionary and Environmental Biology, Institute of Evolution, University of Haifa, 199 Aba-Hushi Avenue, Mount Carmel, Haifa 3498838, Israel*
[2] *Institute of Evolution, University of Haifa, Mount Carmel., Haifa 39105, Israel*
[3] *The Tauber Bioinformatics Center, University of Haifa, 199 Aba-Hushi Avenue, Mount Carmel, Haifa 3498838, Israel*

Correspondence should be addressed to Alexander Bolshoy; bolshoy@research.haifa.ac.il

Received 30 August 2013; Accepted 24 September 2013

Academic Editor: Vassily Lyubetsky

Ancestral sequence reconstruction is a well-known problem in molecular evolution. The problem presented in this study is inspired by sequence reconstruction, but instead of leaf-associated sequences we consider only their lengths. We call this problem ancestral gene length reconstruction. It is a problem of finding an optimal labeling which minimizes the total length's sum of the edges, where both a tree and nonnegative integers associated with corresponding leaves of the tree are the input. In this paper we give a linear algorithm to solve the problem on binary trees for the Manhattan cost function $s(v, w) = |\pi(v) - \pi(w)|$.

1. Introduction

Ancestral sequence reconstruction (ASR) is a well-recognized problem in molecular evolution [1]. Let **G** be a (phylogenetic) tree with **n** leaf nodes, and k strings over one alphabet (gene sequences) assigned to k leaves ($k \leq n$). ASR may be defined in the following way: assignment of strings to inner nodes "in the best possible way." There are two main paradigms for ASR: maximum parsimony (MP) and probabilistic-based reconstruction. The latter includes maximum likelihood (ML) and Bayesian reconstructions. MP reconstruction has a time complexity linear in the number of sequences analyzed. The problem of the parsimonious reconstruction of ancestral states for the given tree with the given states of its leaves (the most parsimonious assignment of the labels of internal nodes for a fixed tree topology) is a well-studied problem [2–4]. Efficient algorithms have also been developed for different types of ML-based reconstructions (reviewed in [5]). ASR methods require as input both a phylogenetic tree and a set of gene sequences associated with corresponding leaves of the tree [6].

ASR is related to gene sequence evolution while the problem presented in this paper, being inspired by ASR, deals with gene length variation. Instead of considering leaf-associated sequences we take into account only their lengths. Instead of the reconstruction of ancestral sequences, we search for the optimal reconstruction of ancestral gene lengths. The problem may be called ancestral gene length reconstruction (AGLR). AGLR is actually a problem of finding an optimal labeling which minimizes the total "length" sum of the edges, the minimum sum problem where both a tree and nonnegative integers associated with corresponding leaves of the tree are the input.

In the graph theory vertex labeling related problems were intensively studied [8]. Typically, the problems can be described as follows: for a given graph, find the optimal way of labeling the vertices with *distinct* integers. The problems and their solutions were described in [9–12]. In [13] we presented the algorithms to solve the minimum sum problem where both a tree and *positive* integers associated with *all leaves* of the tree are the input (finding the optimal way of labeling the vertices with *positive* integers). Here we would like to formulate the minimum sum problem where both a tree and *positive* integers associated with *some of the leaves* of the tree are the input (finding the optimal way of labeling the vertices with *nonnegative* integers). This problem reflects a situation in which the genome tree is constructed by one or another method for a set of genomes, the leaves of the tree

are linked with the corresponding genomes of the set, and the leaves are labeled by integers designating lengths of genes of a chosen gene family. Some leaves would be labeled *zero* because corresponding genomes have no genes of the chosen gene family. Alternatively, it may be a case of a missing value but in this study we do not consider this case: in the problem definition that we bring here zero means "no value."

In this paper we provide a linear algorithm to solve max sum problem on binary trees for the Manhattan cost function $s(v, w) = |\pi(v) - \pi(w)|$. The algorithm uses dynamic programming technique and the properties of the Manhattan distance.

2. Preliminaries

Let **G** be a tree with **n** leaf nodes, vertex set **V**(**G**), and edge set **E**(**G**). $N = |V(G)|$. Let us number the leaf nodes of G:$1, 2, \ldots, n$. Let us number the root of $G : N$. An *integer labeling* $\boldsymbol{\pi}$ of G is a mapping π from G to a set of nonnegative integers, where label **0** is an out-of-the ordinary label meaning "absent value." Let us denote integer labeling of the leaf nodes of $G(\pi(1) = p_1, \ldots, \pi(n) = p_n)$. Let us denote by $g_{\min}$ and $g_{\max}$ minimum and maximum *positive* integers labeling leaf nodes: $g_{\min} = \min p_i : p_i > 0$; $g_{\max} = \max p_i$; $m = g_{\max} - g_{\min} + 1$.

Let us introduce a cost function φ of the edge $vw \in E(G)$:

$$\varphi(x, y) = \begin{cases} 0 & \text{if } x = y \text{ else} \\ C_1 & \text{if } x = 0 \text{ else} \\ C_2 & \text{if } y = 0 \text{ else} \\ \theta(x, y), \end{cases} \tag{1}$$

where the nonnegative cost function $\theta(x, y)$ has the following distance properties:

(i)

$$\theta(x, y) \geq 0 \quad x = y \longleftrightarrow \theta(x, y) = 0 / * \text{ function is equal to zero if and only if its arguments are equal } * / \tag{2}$$

(ii)

$$\theta(x, y) = \theta(y, x) / * \text{ symmetry } * / \tag{3}$$

(iii)

$$x > y \longrightarrow [(\theta(x, y) < \theta(x, y - 1)), (\theta(x, y) < \theta(x + 1, y))]; \tag{4}$$

(iv)

$$x < y \longrightarrow [(\theta(x, y) < \theta(x - 1, y)), (\theta(x, y) < \theta(x, y + 1))]. \tag{5}$$

$C_1 > C_2 > m = g_{\max} - g_{\min} + 1$. C_1 is a gain penalty, C_2 is a loss penalty, and θ is a length change penalty function. Since the likelihoods of loss and gain events are likely to differ, we may need to weight them differently. This is achieved by introducing different penalties $C_1 > C_2$; the loss penalty is normally assigned a value close to $g_{\max} - g_{\min}$, whereas the gain penalty should be larger due to biological considerations. They suggest that, on average, gene loss might be a more likely event than gene gain. Therefore, different gain penalties were used in our study similarly to as it was done in [14].

An example of a function $\theta(x, y)$ is $|\pi(v) - \pi(w)|^{\lambda}$. In case of $\lambda = 1$ we obtain an absolute value of the difference between labelings v and w: $|\pi(v) - \pi(w)|$. In case of $\lambda = 2$ we obtain a square of the difference between labelings v and w: $(\pi(v) - \pi(w))^2$.

2.1. An Arbitrary Tree and an Arbitrary Cost Function. Given a tree G, an integer labeling of the leaves of $G(p_1, \ldots, p_n) = 1$, the gain penalty C_1, the loss penalty C_2, and a cost function θ ((1)–(5)), the minimum sum problem is to find a labeling which minimizes the total cost:

$$S(G) = \min_{\pi} \sum_{\forall\{vw\} \in E(G)} \varphi(\pi(v), \pi(w)) \quad \text{over all } \pi. \tag{6}$$

2.2. A Binary Tree Problem . Given a binary tree G, an integer labeling of the leaves of $G(p_1, \ldots, p_n)$, the "gain" penalty C_1, and the "loss" penalty C_2, the *Manhattan* minimum sum problem is to find the labelings which minimize the sum S over all π

$$S(G) = \sum_{\forall\{vw\} \in E(G) \& \pi(v) \neq 0 \& \pi(w) \neq 0} |\pi(v) - \pi(w)| + k_1 \cdot C_1 + k_2 \cdot C_2, \tag{7}$$

where k_1 is a number of edges of type $(\pi(v) = 0 \,\&\, \pi(w) > 0)$, and k_2 is a number of edges of type $(\pi(v) > 0 \,\&\, \pi(w) = 0)$.

3. Problem Solutions

3.1. DP Algorithm (for the Problem (1)*).* Due to the properties ((2)–(5)) of the cost function $\theta(x, y)$ all labels of the optimal labeling must be either equal to 0 or in the interval $[g_{\min}, g_{\max}]$. As a consequence of this, the dynamic programming (DP) method is applicable for the problem. It will be easier to explain the DP method on a binary tree using $\sigma_k(i)$ notation. The quantity $\sigma_k(i)$ will be interpreted as the minimal cost, given that node k is assigned integer i, to the subtree with the node k as a root of the subtree.

3.1.1. DP Algorithm for a Binary Tree

Up Phase. A procedure called DP_up calculates the costs $\sigma_k(i)$ of all nodes $V(G)$ of the tree G, given a cost function φ.

When we compute $\sigma_N(i)$ for the root node (the index of the root is N), then we simply choose the minimum of these values:

$$S(G) = \min_i \sigma_N(i) \tag{8}$$

Initiation. Given labeling of the leaf nodes of $G(p_1, \ldots, p_n\}$ at the tips of the tree the $\sigma_i(j)$ are easy to compute. The cost is 0 if the observed integer p_i is integer j, and infinite otherwise.

$$\sigma_i(j) = \left\{ \begin{array}{l} 0 \text{ if } j = p_i \\ \infty \text{ otherwise} \end{array} \right\}. \tag{9}$$

Iteration. For the immediate common ancestor of the nodes l and r, node a, we have

$$\begin{aligned} \sigma_a(0) &= \min\left(\sigma_l(0), C_1 + \min_j \sigma_l(j)\right) \\ &\quad + \min\left(\sigma_r(0), C_1 + \min_k \sigma_r(k)\right), \\ \sigma_a(i) &= \min\left(\min_j \left[\theta(i,j) + \sigma_l(j)\right], C_2 + \sigma_l(0)\right) \\ &\quad + \min\left(\min_k \left[\theta(i,k) + \sigma_r(k)\right], C_2 + \sigma_r(0)\right), \\ &\qquad \forall i, j, k \in [g_{\min}, g_{\max}]. \end{aligned} \tag{10}$$

The interpretation of this equation is immediate. The smallest possible cost given that node a is assigned zero is either the cost $\sigma_l(0)$ or the "gain" penalty C_1 plus the minimum of $\sigma_l(j)$, the least of the two plus the minima of corresponding values associated with the right descendant tree. The smallest possible cost given that node a is assigned i is a sum of two values: the first one is either the cost $\theta(x, y)$ of the edge from node a to node l, plus the cost $\sigma_l(j)$ of the left descendant subtree given that node l is in state j, or the "loss" penalty C_2 plus $S_l(0)$; the second one is the cost $\theta(i, k)$ of the edge from the node a to the node r, plus the cost $\sigma_r(k)$ of the right descendant subtree given that node r is in state k. We select those values of j and k which minimize that sum. Equation (10) is applied successively to each inner node in the tree, doing a postorder tree traversal. Finally it computes all the $\sigma_N(i)$, and then (8) is used to find the minimum cost for the whole tree. The complexity of the Up_phase of the algorithm is $O(N^* m^* m)$.

Traceback. The procedure calculates the labels $\pi(p)$ of all nodes p of the tree G.

Choose any integer i which provides the minimum of the $\sigma_N(i)$—it is the root label. It may be either zero or a positive i. Doing a preorder tree traversal, successively label each inner node in the tree: for any inner node p, and given that a parent label i was reconstructed, the label $\pi(p) = j$ is easily reconstructed as well.

3.1.2. DP Algorithm for an Arbitrary Tree

Up-Phase. A procedure DP_up calculates the costs $\sigma_k(i)$ of all nodes $V(G)$ of the tree.

Suppose that the k_a descendant nodes of the node a are called b_j. The following equation will therefore be similar to (10) replacing the sum of σ_l and σ_r by the total sum of σ_{j_1}, while j_1 traverses all values of b_j:

$$\sigma_a(0) = \sum_{j_1}^{k_a} \min\left[\sigma_{j_1}(0), C_1 + \min_j \sigma_{j_1}(j)\right], \tag{11}$$

$$\begin{aligned} \sigma_a(i) &= \sum_{j_1}^{k_a} \min\left[\min_j \left(\theta(i,j) + \sigma_{j_1}(j)\right), C_2 + \sigma_{j_1}(0)\right], \\ &\qquad \forall i, j \in [g_{\min}, g_{\max}]. \end{aligned} \tag{12}$$

This equation is applied successively to each node in the tree, doing a postorder tree traversal. Finally it computes all the $\sigma_N(i)$, and then (8) is used to find the minimum cost for the whole tree.

Down Phase. As Traceback above: Consider the following.

3.2. DP Algorithm for a Manhattan Sum for a Binary Tree (Problem (2)). Manhattan distance $\theta(\pi(v), \pi(w))$ is an absolute value of the difference between labelings v and w : $|\pi(v) - \pi(w)|$. This distance measure has the following property: if siblings have positive labels, then all integers that lie between these values may equally serve as optimal labels of a parent.

(i) If $(\pi(l) \le \pi(r))$, then for all $k\pi(l) \le k \le \pi(r)$ the score $\theta(k, \pi(l)) + \theta(k, \pi(r)) = k - \pi(l) + \pi(r) - k = \pi(r) - \pi(l)$.

(ii) If $(\pi(l) \le \pi(r))$, then for all $k < \pi(l) \le \pi(r)$ the score $\theta(k, \pi(l)) + \theta(k, \pi(r)) = \pi(l) - k + \pi(r) - k = \pi(r) - \pi(l) + 2(\pi(l) - k)$.

(iii) If $(\pi(l) \le \pi(r))$, then for all $\pi(l) \le \pi(r) < k$ the score $\theta(k, \pi(l)) + \theta(k, \pi(r)) = k - \pi(l) + k - \pi(r) = \pi(r) - \pi(l) + 2(k - \pi(r))$.

So, as it would be proven below, at the bottom-up stage of the DP algorithm it would be sufficient to assign to each node a in the tree G four values: left(a), right(a), $Z(a)$, and $X(a)$. The meanings of the values are as follows: left and right are bounds of an interval associated with the node a, Z is a cost value $\sigma_a(0)$, and X is a cost $\sigma_a(i)$ for any integer i from the interval: left $\le i \le$ right.

Initiation. Given labeling of the leaf nodes of $G(p_1, \ldots, p_n) = 1$ these four values are easy to compute for the leaf nodes:

for ($i = 1$; $i \le n$; $i + +$) if ($p[i] == 0$) {$Z[i] = 0$; left[i] = 0; right[i] = 0; $X[i] = C_1 + C_2$} else {$Z[i] = C_1 + C_2$; left[i] = $p[i]$; right[i] = $p[i]$; $X[i] = 0$}.

3.2.1. Examples. Let us consider the simplest trees with two, three, and four labeled leaves. The simplest tree configuration is presented in Figure 1. There is only one node to label—the root node.

(i) Figure 1(a): no genes are assigned to the leaves $\rightarrow$ no gene is assigned to the root.

(ii) Figure 1(b): the left leaf has no gene, and the right leaf has a gene with the length equal to 136 $\rightarrow$ the root

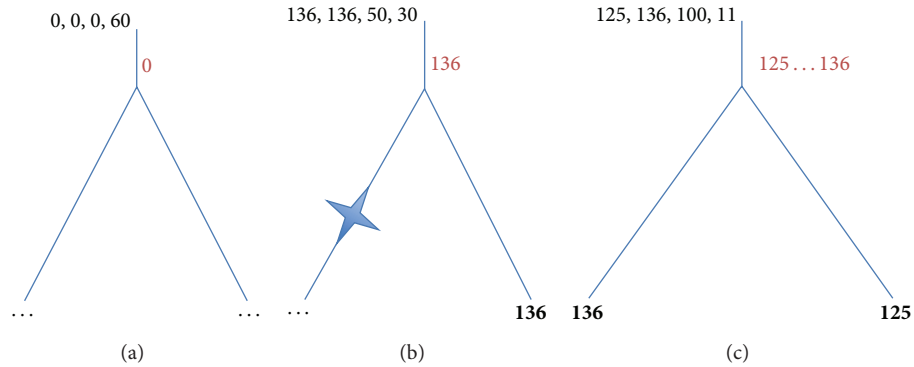

FIGURE 1: Assignment of bottom-up stage values (left, right, Z, and X) in 2-leaf trees. The "gain" penalty $C_1 = 50$; the "loss" penalty $C_2 = 30$. Optimal labels are in red.

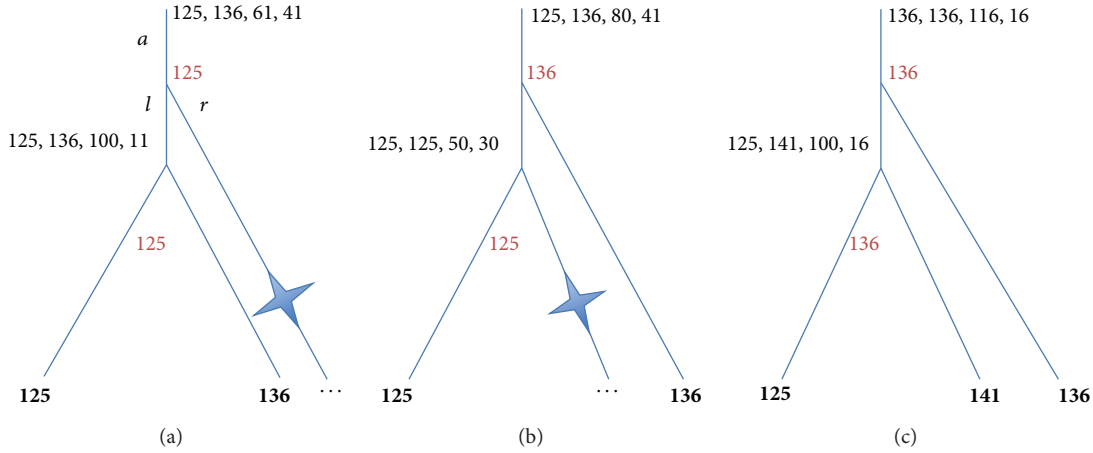

FIGURE 2: Assignment of bottom-up stage values (left, right, Z, and X) in 3-leaf trees. The "gain" penalty $C_1 = 50$; the "loss" penalty $C_2 = 30$. Optimal labels are in red.

is labeled by 136; the score is equal to the loss penalty $C_2 = 30$.

(iii) Figure 1(c): any label $125 \le k \le 136$ is good to label the root; the score is equal to $136 - 125 = 11$.

The next simplest tree topology—three-leaf trees—is presented in Figure 2. There are two nodes to label, the inner node and the root.

(i) Figure 2(a): the inner node is labeled analogically to the root in Figure 1(c): any k $125 \le k \le 136$ is equally good to label the inner node; the root node is labeled analogically to the root in Figure 1(b): $(Z(\text{root}) = C_1 + (136 - 125)) > (X(\text{root}) = C_2 + 11) \rightarrow$ the root is labeled by any k $125 \le k \le 136$, that is, by 125.

(ii) Figure 2(b): labeling is similar to that of Figure 1(a).

(iii) Figure 2(c): the inner node is labeled analogically to Figure 2(a): any label $125 \le k \le 136$ is good to label it; the score is equal to $136-125 = 11$. The root should be labeled by 136 because $125 < 136 < 141$.

Determination of the optimal labeling of the four-leaf trees is very similar to the examples described above. Figure 3 illustrates labeling of the tree where all four leaves have nonzero labels: ((125, 141), (136, 150)). Labeling of the inner nodes is as above (Figure 2(c)): [125, 141] and [136, 150]. All integers of the intersection between these two close intervals are optimal values to label the root: $[125, 141] \cap [136, 150] = [136, 141]$. In Figure 3 we present the value 136 as a chosen suitable label.

Examples of the trees with very distinct subtrees are presented in Figures 4 and 5. In Figure 4 we present a tree obtained by merging two very different subtrees. The left 4-leaf subtree has very obvious intuitive labeling of internal nodes: all nodes should be labeled by zero. The right subtree is identical to the tree presented in Figure 2(c). Merging of these two subtrees produces bottom-up stage values (left, right, Z, and X) to the new root equal to [125, 136, 111, 91]. In spite of assignins the interval [125, 136] to the root only the value 136 provides the optimal solution. (We would like to express our gratitude to the anonymous reviewer for bringing our attention to this situation.) We formulate this rule below describing traceback stage of the algorithm. Figure 4 is chosen to illustrate labeling of nodes similar to the root of the tree.

After considering these few simple examples, we describe the algorithm.

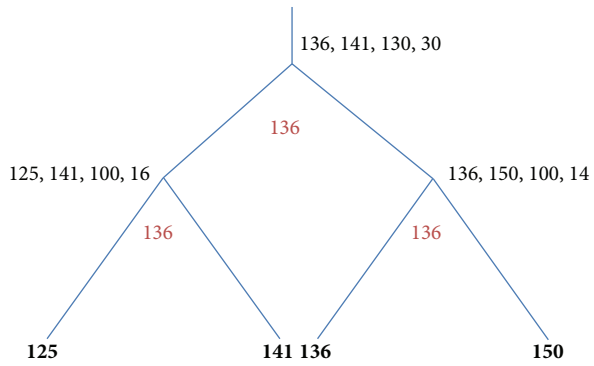

Figure 3: Assignment of bottom-up stage values (left, right, Z, and X) in a 4-leaf tree with all four leaves labeled by positive integers. The "gain" penalty $C_1 = 50$; the "loss" penalty $C_2 = 30$. Optimal labels are in red.

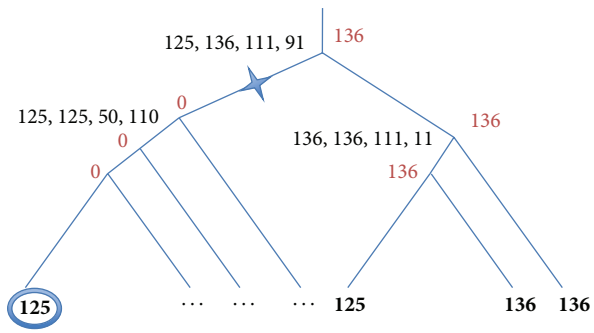

Figure 4: Labeling of a "peculiar" tree. The left subtree has three zero and one nonzero leaf, while the right subtree has three nonzero leaves. The "gain" penalty $C_1 = 50$; the "loss" penalty $C_2 = 30$. Optimal labels are in red.

3.2.2. Bottom-Up Stage

Initiation. Given labeling of the leaf nodes of $G(p_1, \ldots, p_n)$ at the tips of the tree the $\sigma_i(j)$ are easy to compute. The cost is 0 if the observed integer p_i is integer j, and

$$C_1 + C_2 \quad \text{otherwise.} \tag{13}$$

Iteration. Doing a postorder tree traversal assign successively to each node in the tree the abovementioned four values left(a), right(a), $Z(a)$, and $X(a)$. An interval [left(a), right(a)] is assigned according to the following rule: if anyone of two children intervals is not defined, then assign the interval of the other child; otherwise, a parent interval is either an intersection of the intervals of its children or an interval that lies between these intervals if their intersection is empty. Z is a cost value $\sigma_a(0)$, where for the Manhattan distance we can rewrite (10) as

$$\begin{aligned} Z(a) = \sigma_a(0) &= \min\left(\sigma_l(0), C_1 + \sigma_l(j)\right) \\ &+ \min\left(\sigma_r(0), C_1 + \sigma_r(j)\right), \end{aligned}$$

$$\begin{aligned} X(a) = \sigma_a(i) &= \min\left(\min_j\left[\theta(i,j) + \sigma_l(j)\right], C_2 + \sigma_l(0)\right) \\ &+ \min\left(\min_k\left[\theta(i,k) + \sigma_r(k)\right], C_2 + \sigma_r(0)\right) \\ &= \min\left(\min_j\left[|i-j| + \sigma_l(j)\right], C_2 + \sigma_l(0)\right) \\ &+ \min\left(\min_j\left[|i-j| + \sigma_r(j)\right], C_2 + \sigma_r(0)\right). \end{aligned} \tag{14}$$

3.2.3. Pseudocode.

For more details see Pseudocode 1.

3.2.4. Traceback Stage

Interval Correction Rule. Following the bottom-up stage four values left(a), right(a), $Z(a)$, and $X(a)$ are assigned to every internal node a of the tree. An interval (left(a), right(a)) should be diminished if one of the edges connecting the node a with its son becomes of type $(k, 0)$, $k > 0$. Let us denote sons of the node a by $l(a)$ and $r(a)$. Correction condition $\Omega(a)$ would be formulated as

$$\begin{aligned} \Omega(a) = (X(a) \le Z(a)) \quad &\text{and} \\ &((X(l(a)) > Z(l(a)))\, V\, (X(r(a)) > Z(r(a)))). \end{aligned} \tag{15}$$

If $\Omega(a)$ is TRUE, then the bounds of the corrected interval would be obtained by intersection of the interval associated with the node with the corrected interval associated with the corresponding son:

$$\begin{aligned} &\text{if } X(l(a)) > Z(l(a)), \text{then } \left[\text{left}'(a), \text{right}'(a)\right] \\ &\quad = \left[\text{left}(a), \text{right}(a)\right] \\ &\quad \cap \left[\text{left}(r(a)), \text{right}(r(a))\right] \\ &\quad \text{else } \left[\text{left}'(a), \text{right}'(a)\right] \\ &\quad = \left[\text{left}(a), \text{right}(a)\right] \\ &\quad \cap \left[\text{left}(l(a)), \text{right}(l(a))\right]; \end{aligned} \tag{16}$$

Otherwise, the bounds of the corrected interval would not be changed from the original ones:

$$\left[\text{left}'(a), \text{right}'(a)\right] = \left[\text{left}(a), \text{right}(a)\right]. \tag{17}$$

Initiation. Labeling of the root: if $X(N) \le Z(N)$, then correct the root interval according to (15)–(17), and then choose an integer from the corrected interval assigned to the root node otherwise choose 0—it is the root label $\pi(N)$.

Iteration. Doing a preorder tree traversal, successively label each node in the tree either by an integer from the corrected interval assigned to this node which is the nearest to its parent

```
/* Assignment values left[a] and right [a] */
FL = FALSE;
if (left[l] == 0) { left[a] = left[r]; right[a] = right[r]} else
if (left[r] == 0) { left[a] = left[l]; right[a] = right[l]} else
if (left[l] ≤ left[r]) {
  if (right[l] < left[r]) {
        left[a] = right[l]; right[a] = left[r]; FL = TRUE;
  } else {
        left[a] = left[r]; right[a] = min(right[l], right[r]);
  }
} else {
  if (right[r] < left[l]) {
        left[a] = right[r]; right[a] = left[l]; FL = TRUE;
  } else {
        left[a] = left[l]; right[a] = min(right[l], right[r]);
  }
}
/* Assignment values Z[a] and X[a]: Z is a cost of σ_a (0), X is a cost of σ_a (i) for all i:
left ≤ i ≤ right */
if (FL) diff = right[a] − left[a]; else diff = 0;
Z(a) = min(Z(l), C1+X(l)) + min(Z(r),C1+X(r))
X(a) = min(diff + X(l) + X(r), X(r) + C2+ Z(l)), X(l) + C2+ Z(r), 2C2 + Z(l) + Z(r))
```

Pseudocode 1

Table 1: List of archaeal genomes for Figure 4.

No.	Name	Kingdom	Group
0	*Aeropyrum pernix* K1	A	C
1	*Archaeoglobus fulgidus* DSM 4304	A	E
8	*Caldivirga maquilingensis* IC-167	A	C
29	*Haloarcula marismortui* ATCC 43049	A	E
30	*Halobacterium salinarum* R1	A	E
31	*Halobacterium* sp. NRC-1	A	E
32	*Haloquadratum walsbyi* DSM 16790	A	E
35	*Hyperthermus butylicus* DSM 5456	A	C
36	*Ignicoccus hospitalis* KIN4/I	A	C
37	*Metallosphaera sedula* DSM 5348	A	C
38	*Methanobrevibacter smithii* ATCC 35061	A	E
39	*Methanococcoides burtonii* DSM 6242	A	E
40	*Methanococcus aeolicus* Nankai-3	A	E
41	*Methanococcus maripaludis* C5	A	E
42	*Methanococcus maripaludis* C6	A	E
43	*Methanococcus maripaludis* C7	A	E
44	*Methanococcus maripaludis* S2	A	E
45	*Methanosaeta thermophila* PT	A	E
46	*Methanosarcina acetivorans* C2A	A	E
47	*Methanosarcina barkeri* str. fusaro	A	E
48	*Methanosarcina mazei* Go1	A	E
49	*Methanosphaera stadtmanae* DSM 3091	A	E
50	*Methanospirillum hungatei* JF-1	A	E

Notations of the groups: E: *Euryarchaeota*, C: *Crenarchaeota*.

label (it may be either the value equal to the parent label or the boundary value of the interval assigned with the node) or by 0.

The proof of the correctness of a simpler algorithm (without zero-labeled leaves) is published in [15]. In the Appendix there are several lemmas, from which the correctness of the algorithm presented here results.

3.2.5. Example. In Figure 5 of [7] the consensus trees obtained from 100 genome trees were presented. The trees were produced on the basis of 80% randomly chosen COGs, and the right tree was produced on the basis of 15%-jackknifing (the explanations in the text of [7]). This tree possesses phylogenetic reasonableness.

(a) The representatives of both prokaryotic Kingdoms: Eubacteria and Archaea are clustered separately. In other words, Archaeal organisms (genomes 0, 1, 8, 29–32, and 35–50) form a monophyletic group.

(b) Euryarchaeota and Crenarchaeota form monophyletic groups.

A part of this tree was selected to illustrate the algorithm. We took the upper part of the tree related exclusively to Archaea (see A/B marked arrow in Figure 4(b) from [7]) and placed the root at the point dividing all Archaeal genomes into Euryarchaeota and Crenarchaeota (see E/C marked arrow in Figure 4(b) from [7]). Thus, Figure 5 is a part of Figure 4(b) from [7] labeled according to COG0835. This COG was randomly selected as suitable for purposes of illustration. Table 1 presents a list of Archaeal genomes from the whole set of genomes that were used for a genome tree construction (Figure 4(b) from [7]). Table 2 presents the lengths of the Archaeal proteins of this COG.

To assign labels to the leaves of the tree of Figure 5 two preprocessing steps were done: (1) taking off outliers, the lengths 328 of the *H. marismortui* protein and 344 of the *M.*

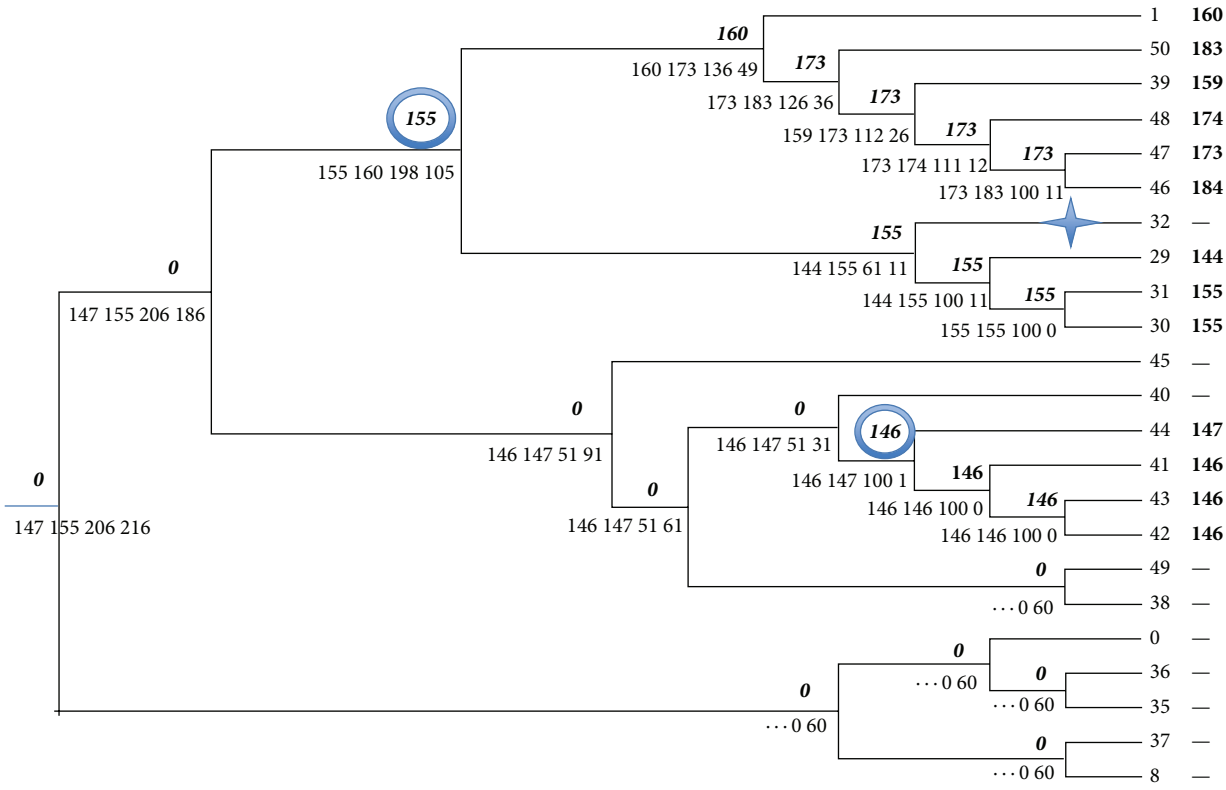

Figure 5: Archaeal part of Figure 4(b) from [7] labeled accordingly to COG0835.

hungatei protein are obvious outliers; (2) taking the median value of paralog's lengths of the genomes 30, 31, 46, 48, and 50. Figure 5 presents results of application of the bottom-up and traceback stages of the algorithm to this tree: a quartet that was assigned to a node a at the bottom upstage is shown under the edge linking the node a and its parent node, a label that was assigned to the node a at the traceback stage, is shown over the same edge.

As we can see the root is labeled by zero. There are two gene-birth events and one gene-death event. One gene was born with the length of 155 and another gene birth is labeled by 146. Genome number 32 (*Haloquadratum walsbyi*) has no protein from COG0835, while other *Haloarchaea* (genomes 29–31) do have. Thus, the edge connecting with leaf labeled by 32 is marked with a gene-loss symbol.

4. Discussion

In [15] the algorithms to find the optimal labeling of the vertices of the tree under Wagner parsimony were presented. A simple extension of the problem could be finding the optimal labeling of the vertices of the tree with nonnegative integers. This more realistic approach requests special consideration of zero labeling. Wedges of type $(k, 0)$, $k > 0$, should be scored differently from wedges of type $(0, k)$, $k > 0$, because the $(k, 0)$ notes gene loss, while $(0, k)$ notes gene gain. These events should be scored differently. Interestingly, this differentiated scoring in addition to tree labeling resulted in reconstruction of "parsimonious" evolutionary scenario. Reconstruction of a gene evolution along a species tree is an interesting and principal problem. Lyubetsky and his coworkers contributed a lot to formulation and solving this problem. In their studies [16–22] the authors tackled mainly two important and sophisticated phylogenetic problems. The obtained results are partially reviewed in the first section of [22] which also provides an extended biological background and relevant references. Reconstruction of a gene evolution along a species tree (to build the evolutionary scenario), following the approach of Lyubetsky et al., is to find an optimal mapping of a gene tree into a species tree. (An example of a different approach was presented in [14].) The second problem is to construct a supertree from the given set of gene trees.

As it was mentioned in [22], the first problem, stated as a tree-into-tree mapping, is solved in polynomial (often linear, and at maximum cubic) time even for the case of time slices and horizontal gene transfers. The algorithms presented in our study are polynomial as well.

Choosing C_1 (a gain penalty), C_2 (a loss penalty), and θ (a label change penalty) is crucial for reconstruction of trustworthy evolutionary scenario. However, it is very difficult task and we cannot claim categorically that choosing "correct" parameters of the model will result in truly reliable reconstruction. We do plan to make a comparison between results obtained by abovementioned methods of Lyubetsky and ours (work in progress).

Table 2: Protein lengths of the chemotaxis signal transduction proteins. Archaeal part of COG0835.

Number	COG	Length	Genome name
1	835	160	*Archaeoglobus fulgidus* DSM 4304
29	835	144	*Haloarcula marismortui* ATCC 43049
29	835	328	*Haloarcula marismortui* ATCC 43049
30	835	132	*Halobacterium salinarum* R1
30	835	178	*Halobacterium salinarum* R1
31	835	132	*Halobacterium* sp. NRC-1
31	835	178	*Halobacterium* sp. NRC-1
39	835	159	*Methanococcoides burtonii* DSM 6242
41	835	146	*Methanococcus maripaludis* C5
42	835	146	*Methanococcus maripaludis* C6
43	835	146	*Methanococcus maripaludis* C7
44	835	147	*Methanococcus maripaludis* S2
46	835	182	*Methanosarcina acetivorans* C2A
46	835	184	*Methanosarcina acetivorans* C2A
47	835	173	*Methanosarcina barkeri* str. fusaro
48	835	159	*Methanosarcina mazei* Go1
48	835	189	*Methanosarcina mazei* Go1
50	835	124	*Methanospirillum hungatei* JF-1
50	835	167	*Methanospirillum hungatei* JF-1
50	835	169	*Methanospirillumhungatei* JF-1
50	835	169	*Methanospirillum hungatei* JF-1
50	835	174	*Methanospirillumhungatei* JF-1
50	835	176	*Methanospirillum hungatei* JF-1
50	835	183	*Methanospirillum hungatei* JF-1
50	835	187	*Methanospirillum hungatei* JF-1
50	835	189	*Methanospirillum hungatei* JF-1
50	835	190	*Methanospirillum hungatei* JF-1
50	835	198	*Methanospirillum hungatei* JF-1
50	835	200	*Methanospirillum hungatei* JF-1
50	835	344	*Methanospirillum hungatei* JF-1
50	835	779	*Methanospirillum hungatei* JF-1

To prepare input for the algorithm, as it was done above for 3.2.5, the original data is to be transformed to the following format: to each (genome, COG) pair one standardized protein length should be assigned (as we described in [7]). For a given COG, each organism is represented by a calculated length—a median length of all paralogous proteins. A natural extension would be to formulate the labeling problem taking into account existence of paralogs.

We may define a *k-tuple* integer labeling Π of G as a mapping Π from G to a set of k-tuples composed of integers $\Pi(v) = \{\pi_1(v), \pi_2(v), \ldots, \pi_{k(v)}(v)\}$, where $\pi_i(v) \leq \pi_{i+1}(v)$ for all $1 \leq i < k(v)$. The simplest extension would be to introduce the case with *identical* sizes of k-tuples composed of *nonnegative* integers. A *uniform* k-tuple integer labeling Π_c of G is characterized by a constant $k(v)$ for all v. The stretch of the edge vw in a $\Pi_c(G)$ is a simple sum $c_{vw} = \sum_{i=1}^{k} \varphi(\pi_i(v), \pi_i(w)) \cdot \varphi(x, y)$ is defined as in (1). Given a uniform k-tuple integer labeling of the leaves of G the minimum sum problem is to find a labeling which minimizes the total sum of the stretches of the edges. Some $\boldsymbol{\pi_i(v)} = \mathbf{0}$. The minimum sum problem is that of minimizing $\mathbf{s(G)} = \sum_{\forall\{vw\}\in E(G)} c_{vw}$ over all Π_c for given k. By some modifications of the algorithms presented in this paper the minimizing k-tuple labeling can be found. This model again is a gain-loss model. More sophisticated extension must provide more realistic definition of distance between two k-tuples composed of positive integers by introducing duplication events.

Appendix

Lemma A.1 (root optimal label). *Suppose that the root node is called rt and suppose that its children are called l and r. The claim is*

(1) *if* $(\pi(l) = \pi(r) = 0)$*, then* $\pi(rt) = 0$ *else*

(2) *if* $(\pi(l) = 0)$*, then* $\pi(rt) = \pi(r)$ *else*

(3) *if* $(\pi(r) = 0)$*, then* $\pi(rt) = \pi(l)$ *else*

(4) *if* $(\pi(l) \leq \pi(r))$*, then* $\pi(l) \leq \pi(rt) \leq \pi(r)$ *else*

(5) $\pi(l) \geq \pi(rt) \geq \pi(r)$.

Proof. If we consider a subtree with the node k as a root then $\sigma_k(i)$ designates the minimal cost, given that node k has a label i:

$$\begin{aligned} S(G) &= \min_i \sigma_N(i) \\ &= \min_i \left[\varphi(\pi(rt), \pi(l)) \right. \\ &\qquad \left. + \varphi(\pi(rt), \pi(r)) + \sigma_l(\pi(l)) + \sigma_r(\pi(r))\right]. \end{aligned} \tag{A.1}$$

Proof of the subclaims (1)–(3) is trivial. Case (4) is

$$\begin{aligned} S(G) &= \min_i \sigma_N(i) \\ &= \min_i \left[\, |\pi(rt) - \pi(l)| + |\pi(rt) - \pi(r)| \right. \\ &\qquad \left. + \sigma_l(\pi(l)) + \sigma_r(\pi(r))\right]. \end{aligned} \tag{A.2}$$

Let us introduce a new numbering π' by changing only the root label: $\pi'(rt) = \pi(l)$. It is easy to see that $S_{\pi'}(G) < S_{\pi}(G)$. It means that for optimal integer labeling π the following is correct: $\pi(rt) \geq \pi(l)$. Likewise, we prove that for optimal integer labeling $\pi(rt) \leq \pi(r)$. Let us denote $k = \pi(rt) : \pi(l) \leq k \leq \pi(r)$:

$$\begin{aligned} &\forall k\,(\pi(l) \leq k \leq \pi(r))\, S_\pi(G) \\ &\quad = k - \pi(l) + S_\pi(l) + \pi(r) - k + S_\pi(r), \quad \text{q.e.d.} \end{aligned} \tag{A.3}$$

□

Lemma A.2 (leaf parent optimal interval). *Every node of the optimal integer labeling that all its children are leaf nodes has either a zero label or a label between labels of its children.*

Suppose that the root node is called a and suppose that its children are called l and r. The claim is

(1) *if* $(\pi(l) = \pi(r) = 0)$*, then* $\pi(a) = 0$ *else*

(2) *if* $(\pi(l) = 0)$*, then* $\pi(a) = \pi(r)$ *else*

(3) *if* $(\pi(r) = 0)$, *then* $\pi(a) = \pi(l)$ *else*

(4) *if* $(\pi(l) \le \pi(r))$, *then* $\pi(l) \le \pi(a) \le \pi(r)$ *else*

(5) $\pi(l) \ge \pi(a) \ge \pi(r)$.

Proof. Figure 1 illustrates this lemma. Proof of the subclaims (1)–(3) is trivial. In case of condition (4) let us prove that for optimal integer labeling $\pi(a) \ge \pi(l)$. Suppose $\pi(a) < \pi(l)$. Let us denote $(\pi(l) - \pi(a)) = \delta$; $\pi(r) - \pi(l) = \gamma$. Let us introduce a new numbering π' by changing only the label of the node a: $\pi'(a) = \pi(l)$. It is to show that $S_{\pi'}(G) < S_\pi(G)$. Indeed,

$$\begin{aligned} S_{\pi'}(G) &= S_\pi(G) - \left|\pi(k) - \pi(j)\right| \\ &\quad - \left(p_i - \pi(k)\right) - \left(p_{i+1} - \pi(k)\right) \\ &\quad + \left|\pi'(k) - \pi(j)\right| + \left(p_i - \pi'(k)\right) + \left(p_{i+1} - \pi'(k)\right) \\ &= S_\pi(G) + \left(\left|p_i - \delta - \pi(j)\right| - \left|p_i - \pi(j)\right|\right) \\ &\quad - \left(p_i - \pi(k)\right) - \left(p_{i+1} - \pi(k)\right) + \left(p_i - p_i\right) \\ &\quad + \left(p_{i+1} - p_i\right) = S_\pi(G) \\ &\quad + \left(\left|p_i - \delta - \pi(j)\right| - \left|p_i - \pi(j)\right|\right) \\ &\quad - \delta - (\delta + \gamma) - \gamma = S_\pi(G) - \delta. \end{aligned} \tag{A.4}$$

Likewise, we prove that for optimal integer labeling $\pi(k) \le p_{i+1}$. Q.e.d. $p_i \le \pi(k) \le p_{i+1}$. □

Lemma A.3 (parent optimal interval—(I)). *An optimal label of a parent either is equal to zero or lies between extreme values of optimal labels of its children. If an optimal integer labeling* π *provides the labels of two siblings* i_1 *and* i_2 *satisfying the conditions* $a_1 \le \pi(i_1) \le b_1$ *&* $a_2 \le \pi(i_2) \le b_2$, *then the label of their parent* k *satisfies* $\min(a_1, b_1, a_2, b_2) \le \pi(k) \le \max(a_1, b_1, a_2, b_2)$. *Proof is as for Lemma A.1.*

Lemma A.4 (parent optimal interval—(II)). *An optimal label of a parent in case of the empty intersection of the optimal intervals of its children lies between these intervals.*

If an optimal integer labeling π *provides the labels of two siblings* i_1 *and* i_2 *satisfying the conditions* $(a_1 \le \pi(i_1) \le b_1)$ *&* $(a_2 \le \pi(i_2) \le b_2)$ *then if* $(b_1 \le a_2)$ *then* $b_1 \le \pi(k) \le a_2$ *else if* $(b_2 \le a_1)$ *then* $b_2 \le \pi(k) \le a_1$.

Proof. (1) $b_1 \le a_2$. Let us assume $\pi(k) < b_1$; $\pi(k) = b_1 - \alpha$. Then we introduce a new labeling π' by changing labels for three nodes: $\pi'(k) = b_1$; $\pi'(i_1) = b_1$; $\pi'(i_2) = a_2$:

$$\begin{aligned} &S_\pi(G) - S_{\pi'}(G) \\ &\quad = \left(\left|\pi(j) - \pi(k)\right| - \left|\pi(j) - \pi'(k)\right|\right) \\ &\qquad + \left(\left|\pi(k) - \pi(i_1)\right| - \left|\pi'(k) - \pi'(i_1)\right|\right) \\ &\qquad + \left(\left|\pi(k) - \pi(i_2)\right| - \left|\pi'(k) - \pi'(i_2)\right|\right), \\ &\left(\left|\pi(j) - \pi(k)\right| - \left|\pi(j) - \pi'(k)\right|\right) \\ &\quad = \left|\pi(j) - b_1 + \alpha\right| - \left|\pi(j) - b_1\right| = \alpha, \\ &\left(\left|\pi(k) - \pi(i_1)\right| - \left|\pi'(k) - \pi'(i_1)\right|\right) \\ &\quad = \left(\left(\pi(k) - \pi(i_1)\right) - \left(\pi'(k) - \pi'(i_1)\right)\right) \\ &\quad = b_1 - \alpha - \pi(i_1) - b_1 + b_1 \\ &\quad = b_1 - \alpha - \pi(i_1), \\ &\left(\left|\pi(k) - \pi(i_2)\right| - \left|\pi'(k) - \pi'(i_2)\right|\right) \\ &\quad = \pi(i_2) - b_1 + \alpha - a_2 + b_1, \\ &S_\pi(G) - S_{\pi'}(G) = \alpha + b_1 - \alpha - \pi(i_1) \\ &\qquad + \pi(i_2) + \alpha - a_2 = \left(b_1 - \pi(i_1)\right) \\ &\qquad + \alpha + \left(\pi(i_2) - a_2\right) > 0. \end{aligned} \tag{A.5}$$

From assumption $\pi(k) < b_1$ follows that π is not an optimal labeling. Similarly, we can prove that from assumption $\pi(k) > a_2$ follows that π is not an optimal labeling.

(2) $b_2 \le a_1$. Similarly to (1) let us assume $\pi(k) < b_2$; $\pi(k) = b_2 - \alpha$. Then we introduce a new labeling π' by changing labels for three nodes: $\pi'(k) = b_2$; $\pi'(i_1) = a_1$; $\pi'(i_2) = b_2$. $S_\pi(G) - S_{\pi'}(G) > 0$, so we have a contradiction with the statement that $\pi(G)$ is an optimal labeling. □

Lemma A.5 (parent optimal interval—(III)). *An optimal interval of a parent is either an intersection of the optimal intervals of its children or an interval that lies between these intervals in case that their intersection is empty.*

If an optimal integer labeling π *provides the labels of two siblings* i_1 *and* i_2 *satisfying the conditions* $a_1 \le \pi(i_1) \le b_1$ *&* $a_2 \le \pi(i_2) \le b_2$, *then the label of their parent* k *satisfies the following condition:*

if $([a_1, b_1] \cap [a_2, b_2]) \ne \emptyset$ *then* $\pi(k) \in [a_1, b_1] \cap [a_2, b_2])$ *else*

if $b_1 < a_2$ *then* $\pi(k) \in [b_1, a_2]$ *else* $\pi(k) \in [b_2, a_1]$.

For example, if $a_1 \le a_2 \le b_1 \le b_2$, *then Lemma A.5 states that* $\pi(k)$ *satisfies the following condition* $a_2 \le \pi(k) \le b_1$. *Proof is similar to proof in Lemma A.4.*

References

[1] D. A. Liberles, Ed., *Ancestral Sequence Reconstruction*, Oxford University Press, Oxford, UK, 2007.

[2] W. M. Fitch, "Towards defining the course of evolution: minimum change for a specific tree topology," *Systematic Zoology*, vol. 20, pp. 406–416, 1971.

[3] D. Sankoff and P. Rousseau, "Locating the vertices of a steiner tree in an arbitrary metric space," *Mathematical Programming*, vol. 9, no. 1, pp. 240–246, 1975.

[4] D. Sankoff, "Minimal mutation trees of sequences," *SIAM Journal on Applied Mathematics*, vol. 28, no. 1, pp. 35–42, 1975.

[5] T. Pupko, A. Doron-Faigenboim, D. A. Liberles, and G. M. Cannarozzi, "Probabilistic models and their impact on the accuracy of reconstructed ancestral protein sequences," in *Ancestral Sequence Reconsruction*, D. A. Liberles, Ed., Oxford Oxford University Press, 2007.

[6] H. Ashkenazy, O. Penn, A. Doron-Faigenboim et al., "FastML: a web server for probabilistic reconstruction of ancestral sequences," *Nucleic Acids Research*, vol. 40, pp. W580–W584, 2012.

[7] A. Bolshoy and Z. Volkovich, "Whole-genome prokaryotic clustering based on gene lengths," *Discrete Applied Mathematics*, vol. 157, no. 10, pp. 2370–2377, 2009.

[8] F. R. K. Chung, "Some problems and results in labelings of graphs," in *The Theory and Applications of Graphs*, G. Chartland, Ed., pp. 255–263, John Wiley & Sons, New York, NY, USA, 1981.

[9] M. A. Iordanskii, "Minimal numberings of the vertices of trees," *Soviet Mathematics Doklady*, vol. 15, pp. 1311–1315, 1974.

[10] M. K. Goldberg and I. Klipker, "An algorithm for a minimal placement of a tree on a line," *Sakartvelos Mecnierebata Akademiis Moambe*, vol. 83, pp. 553–556, 1976 (Russian).

[11] F. R. K. Chung, "On optimal linear arrangements of trees," *Computers and Mathematics with Applications*, vol. 10, no. 1, pp. 43–60, 1984.

[12] F. R. K. Chung, "On the cutwidth and the topological bandwidth of a tree," *SIAM Journal on Algebraic and Discrete Methods*, vol. 6, pp. 268–277, 1985.

[13] A. Bolshoy and V. Kirzhner, "Algorithms of an optimal integer tree labeling," http://arxiv.org/abs/1305.5551.

[14] B. G. Mirkin, T. I. Fenner, M. Y. Galperin, and E. V. Koonin, "Algorithms for computing parsimonious evolutionary scenarios for genome evolution, the last universal common ancestor and dominance of horizontal gene transfer in the evolution of prokaryotes," *BMC Evolutionary Biology*, vol. 3, no. 1, article 2, 2003.

[15] J. S. Farris, "Methods for computing Wagner trees," *Systematic Zoology*, vol. 19, pp. 83–92, 1970.

[16] K. Y. Gorbunov and V. A. Lyubetsky, "Reconstructing the evolution of genes along the species tree," *Molecular Biology*, vol. 43, no. 5, pp. 881–893, 2009.

[17] K. Y. Gorbunov and V. A. Lyubetsky, "An algorithm of reconciliation of gene and species trees and inferring gene duplications, losses and horizontal transfers," *Information Processes*, vol. 10, pp. 140–144, 2010 (Russian).

[18] J.-P Doyon, C. Scornavacca, K. Y. Gorbunov, G. J. Szeollosi, V. Ranwez, and V. Berry, "An efficient algorithm for gene/species trees parsimonious reconciliation with losses, duplications and transfers," in *Lecture Notes in Bioinformatics*, S. Istrail, P. Pevzner, and M. Waterman, Eds., vol. 6398 of *Subseries of Lecture Notes in Computer Science*, pp. 93–108, Springer, Berlin, Germany, 2010.

[19] K. Y. Gorbunov and V. A. Lyubetsky, "The tree nearest on average to a given set of trees," *Problems of Information Transmission*, vol. 47, no. 3, pp. 274–288, 2011.

[20] K. Y. Gorbunov and V. A. Lyubetsky, "Fast algorithm to reconstruct a species supertree from a set of protein trees," *Molecular Biology*, vol. 46, no. 1, pp. 161–167, 2012.

[21] V. A. Lyubetsky, L. I. Rubanov, L. Y. Rusin, and K. Y. Gorbunov, "Cubic time algorithms of amalgamating gene trees and building evolutionary scenarios," *Biology Direct*, vol. 7, pp. 1–20, 2012.

[22] L. Y. Rusin, E. V. Lyubetskaya, K. Y. Gorbunov, and V. A. Lyubetsky, "Reconciliation of gene and species trees," *BioMed Research International*. In press.