Available online at www.sciencedirect.com
SCIENCE DIRECT®
Journal of Algorithms 51 (2004) 1–14
Journal of Algorithms
www.elsevier.com/locate/jalgor

Selecting the branches for an evolutionary tree. A polynomial time approximation scheme

Jonathan Badger,[a,1] Paul Kearney,[a,*,2] Ming Li,[a,3] John Tsang,[a,4] and Tao Jiang [b,5]

[a] *Department of Computer Science, University of Waterloo, Waterloo, ON N2L3G1, Canada*
[b] *Department of Computer Science, University of California, Riverside, CA 92521, USA*

Received 24 August 2000

Abstract

Many fundamental questions in evolution remain unresolved despite the abundance of genetic sequence data that is now available. This state of affairs is partly due to the lack of simultaneously efficient and accurate computational methods for inferring evolutionary trees. Efficient methods are critical since the abundance of sequence data has resulted in the need to analyze large datasets. Methods with guaranteed accuracy are important since biologists require proof that results are meaningful. In this paper the first polynomial time approximation scheme (PTAS) for selecting the branches of an evolutionary trees from a list of candidate branches is presented. PTAS's are highly desirable since they allow the approximation of an optimal solution with arbitrary precision in polynomial time. This PTAS is based upon recent advances in the approximation of smooth polynomial integer programs.

Keywords: Approximation; Character compatibility; Evolution; Phylogeny; PTAS

* Corresponding author.
E-mail addresses: jhbadger@caprion.com (J. Badger), pkearney@caprion.com (P. Kearney), mli@math.uwaterloo.ca (M. Li), jsctsang@math.uwaterloo.ca (J. Tsang), jiang@cs.ucr.edu (T. Jiang).

[1] Supported by CITO.
[2] Supported by CITO and NSERC grant 160321.
[3] Supported by CITO, NSERC grant OGP0046506, CGAT and a Steacie Fellowship.
[4] Supported by NSERC.
[5] Supported by NSERC Research Grant OGP0046613, CITO and a UCR startup grant.

0196-6774/$ – see front matter
doi:10.1016/S0196-6774(03)00086-5

1. Introduction

In recent years the amount of DNA sequence data available has grown exponentially.[6] This data are both deep and broad in the sense that entire genomes of several organisms have been sequenced (such as yeast) and several genes shared by the majority of species on the planet have been widely sequenced (such as HSP70).

This has had, and will continue to have, a major impact on biological and medical science. In particular, the breadth and accuracy of these data has made it possible to address evolutionary questions once out of reach. For example, the Ribosomal Database Project[7] at Michigan State University now contains evolutionary trees describing the evolutionary history of over 33000 small sub unit rRNA sequences [16].

An obstacle to constructing large evolutionary trees is not the availability of data but the current state of the computational methods for inferring evolutionary trees. The problem of inferring an evolutionary tree from sequence data has many formulations including maximum likelihood [9], maximum parsimony [10], minimum distance [18] and quartet recombination [4,5,7,15,20]. Virtually all of these formulations are NP-complete, and so, methods for inferring evolutionary trees tend to be either heuristic or exhaustive in nature. This is frustrating for biologists since popular heuristic methods are often statistically inconsistent or slow to converge [14] whereas the growing size of datasets render exhaustive methods such as maximum likelihood infeasible.

The challenge for computer scientists is to develop methods for inferring evolutionary trees that are both efficient and give accuracy guarantees. Given that most formulations of the evolutionary tree inference problem are NP-hard, the task is to develop *polynomial time approximation schemes* (*PTAS's*) for inferring evolutionary trees. A PTAS approximates the optimal solution to a problem with arbitrary precision in polynomial time.

In this paper an advance of this nature is presented: the first PTAS for obtaining an evolutionary tree from a list of candidate branches.

2. From branches to trees: character compatibility

An evolutionary tree is modeled by a rooted binary tree T where the leaves of T are labeled by a set S. The edges of an evolutionary tree have an associated length that represents the amount of mutation along that lineage. Labels in S are typically species or genes. If e is an edge of T then e induces the bipartition (A, B) of S if $T - \{e\}$ consists of a tree labeled by A and a tree labeled by B. This is denoted $e = (A, B)$ and (A, B) is called a *bipartition* of T. A bipartition (A, B) is *trivial* if either $|A| = 1$ or $|B| = 1$. Trivial bipartitions are induced by leaf edges of T. The set *bipartitions*(T) is the set of edge induced bipartitions of T.

It is well known that the structure of an evolutionary tree T can be recovered from its set of bipartitions [3]. That is, the bipartitions of an evolutionary tree give specific information

[6] Visit http://www.ebi.ac.uk/sterk/genome-MOT/ for an online progress report on major genome sequencing efforts.

[7] The RDP's URL is http://rdp.cme.msu.edu.

about T. This motivates the following paradigm for inferring an evolutionary tree T' that approximates the unknown true evolutionary tree T:

1. Obtain a set R of bipartitions of S from the data such that R approximates *bipartitions*(T).
2. Recombine the bipartitions in R, under some optimality criterion, to produce a tree T'.
3. Determine the root and edge lengths of T'.

There are many diverse methods that can be used in step 1 to obtain a set R of bipartitions supported by the data.

- Characters that bipartition S can be derived from sequence or morphological data [21]. For example, a gene can be used to separate species into those that possess the gene and those that do not [19].
- Data are often analyzed using several techniques each producing different trees. These trees can be used to generate a list of edge induced bipartitions from which a consensus tree is to be constructed.
- Methods such as hypercleaning [5] can be used to generate a list of bipartitions most highly supported by sequence data.

Step 3 of this paradigm, rooting and determining edges lengths, is well-studied and several methods exist for doing this [21].

A pair of bipartitions are *compatible* if they can exist within the same evolutionary tree. A set of bipartitions is compatible if they are pairwise compatible, or in other words, can all exist within the same tree. Clearly, the largest number of compatible bipartitions is $2|S|-1$ which is the number of edges in an evolutionary tree labeled by S. When R is compatible, obtaining an evolutionary tree such that $R \subseteq$ *bipartitions*(T) can be done in $O(nm)$ time where $n = |S|$ and $m = |R|$ [12].

However, it is typical that $|R| > 2|S| - 1$, and so, R is not *compatible*. For example, Fig. 1 depicts a partial list of bipartitions (displayed as boxes) of a set of HSP70 sequences, ordered left to right by support, generated by hypercleaning. Arcs connect incompatible bipartitions.

Step 2 of the above paradigm has many formulations. Biologists have examined bipartition compatibility for many years [8,17]. A natural generalization of this problem is the well-studied *perfect phylogeny problem* [1,13]. However, this formulation requires that the bipartitions in R meet certain assumptions not often satisfied when the bipartitions are obtained from experimental data. As a result, the recombination of compatible bipartitions has been formulated as the following optimization problem:

Character Compatibility (CC)
Instance: Set R of bipartitions of label set S.
Problem: Find a tree T labeled by S that maximizes $|$*bipartitions*$(T) \cap R|$.

The decision version of *Character Compatibility* is NP-complete [6]. Furthermore, *Character Compatibility* cannot be approximated within ratio n^{ϵ}. This follows from the NP-completeness proof that reduces *Clique* to *Character Compatibility* [6] and the inapproximability of *Clique* [2].

A criticism of the **CC** formulation is that the optimization criterion ignores those bipartitions of R that are not bipartitions of the constructed tree. Hence, bipartitions that

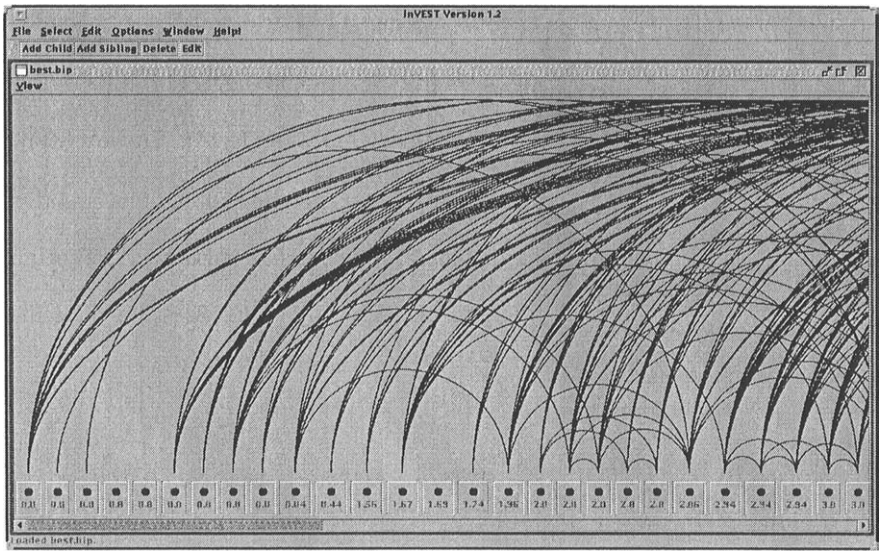

Fig. 1. Typically, bipartitions are not compatible as in this analysis.

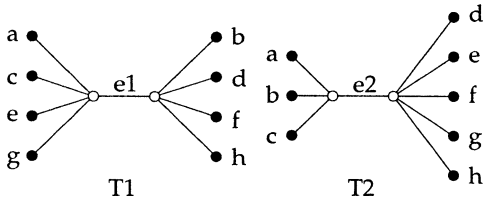

Fig. 2. T_1 and T_2: T_2 fits $(\{a, b, c, d\}, \{e, f, g, h\})$ better.

may be informative, though not perfect, do not contribute to the score of the constructed tree.

A better optimization criterion would attempt to fit a tree to all bipartitions using a more accurate measure. To illustrate, consider the bipartition $(A, B) = (\{a, b, c, d\}, \{e, f, g, h\})$ and the two trees T_1 and T_2 appearing in Fig. 2. Neither of these trees contain the bipartition (A, B) and so the **CC** optimization criterion would not differentiate between these two trees but clearly T_2 is a better approximation of (A, B). In particular, four leaf relocations are required for T_1 to contain the bipartition (A, B) but only one leaf relocation is required for T_2 to contain (A, B).

Motivated by this example, define the similarity between two bipartitions (A, B) and (C, D) to be $s((A, B), (C, D)) = \max\{|A \cap C| + |B \cap D|, |A \cap D| + |B \cap C|\}$. In the above example, $s(e_1, (A, B)) = 4$ whereas $s(e_2, (A, B)) = 7$ (recall that $e_1 = (\{a, c, e, g\}, \{b, d, f, h\})$ and $e_2 = (\{a, b, c\}, \{d, e, f, g, h\})$). The distance between two bipartitions (A, B) and (C, D) is defined to be $d((A, B), (C, D)) = |A \cup B| - s((A, B), (C, D))$.

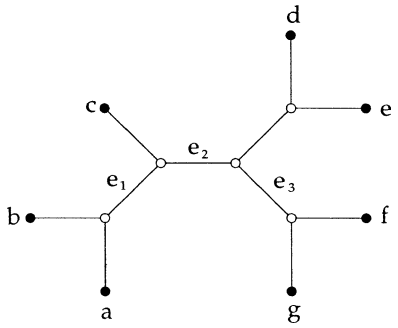

Fig. 3. Illustrating $s(T, R)$.

Extending the notion of similarity to trees, define the similarity between a tree T and a bipartition (A, B), denoted $s(T, (A, B))$, to be $\max_{e \in T}(e, (A, B))$. That is, the similarity is the maximum similarity between (A, B) and a bipartition of T. Finally, the similarity between a set R of bipartitions and a tree T is

$$s(T, R) = \sum_{(A,B) \in R} s\big(T, (A, B)\big).$$

Similarity allows us to define a fractional version of **CC**:

Fractional Character Compatibility (FCC)
Instance: Set R of bipartitions of label set S.
Problem: Find a tree T labeled by S that maximizes $s(T, R)$.

To illustrate, if $R = \{(\{a, b, c, d\}, \{e, f, g\}), (\{a, b, g\}, \{c, d, e, f\}), (\{a, c, d, e, f\}, \{b, g\})\}$ then for the tree T depicted in Fig. 3

$$\begin{aligned} s(T, R) &= s\big(T, \big(\{a, b, c, d\}, \{e, f, g\}\big)\big) + s\big(T, \big(\{a, b, g\}, \{c, d, e, f\}\big)\big) \\ &\quad + s\big(T, \big(\{a, c, d, e, f\}, \{b, g\}\big)\big) \\ &= s\big(e_2, \big(\{a, b, c, d\}, \{e, f, g\}\big)\big) + s\big(e_1, \big(\{a, b, g\}, \{c, d, e, f\}\big)\big) \\ &\quad + s\big(e_3, \big(\{a, c, d, e, f\}, \{b, g\}\big)\big) \\ &= 6 + 6 + 5 = 17. \end{aligned}$$

The remainder of the paper is organized as follows. In Section 3 it is proven that **FCC** is NP-complete and in Section 4 a PTAS for **FCC** is presented.

3. FCC is NP-complete

In this section we prove the following:

Theorem 3.1. ***FCC*** *is NP-complete.*

Proof. We show that **FCC** is NP-hard by a reduction from the problem **X3C** [11].

Exact Cover by 3-Sets (X3C)

Instance: Set S with $|S| = 3q$ for some integer q and a collection $C = \{S_1, \ldots, S_m\}$, where each S_i, is 3-element subset of S.

Question: Does there exist a subcollection C' of C such every element of S occurs in exactly one subset in C'?

For the reduction from **X3C** to **FCC**, we need an extra constraint that for all $i \neq j$, $|S_i \cap S_j| \leqslant 1$.

Lemma 3.1. *The problem **X3C** remains NP-hard even if we assume that all $i \neq j$, $|S_i \cap S_j| \leqslant 1$.*

Proof. We reduce the general **X3C** to our restricted version. Let set S and collection $C = \{S_1, \ldots, S_m\}$ be an instance of **X3C**. For each $S_i = \{x, y, z\}$, construct sets $X_i = \{x, a_i, b_i\}$, $Y_i = \{y, c_i, d_i\}$, $Z_i = \{z, e_i, f_i\}$, $A_i = \{a_i, c_i, e_i\}$, and $B_i = \{b_i, d_i, f_i\}$, where $a_i, b_i, c_i, d_i, e_i, f_i \notin S$ are new elements. Let

$$S' = S \cup \bigcup_{i=1}^{m} \{a_i, b_i, c_i, d_i, e_i, f_i\}, \qquad C' = C \cup \bigcup_{i=1}^{m} \{X_i, Y_i, Z_i, A_i, B_i\}.$$

Then clearly, the subsets in C' satisfy the constraint. Moreover, for each i, any exact cover $C'' \subseteq C'$ of S' contains either all subsets X_i, Y_i, Z_i or none of them. Hence, C contains an exact cover of S if and only if C' contains an exact cover of S'. □

We reduce the above restricted version of **X3C** to **FCC**. Let set S and collection $C = \{S_1, \ldots, S_m\}$ be an instance of the restricted **X3C** such that for all $i \neq j$, $|S_i \cap S_j| \leqslant 1$. Suppose that $|S| = 3q$. We construct an instance of **FCC** as follows: for each $S_i \in C$, define a bipartition $(S_i, S - S_i)$. Then we claim that S has an exact cover in C if and only if the **FCC** instance has a solution of cost $2(m - q)$.

To prove the "only if" part of the above claim, suppose that $C' = \{S_{i_1}, \ldots, S_{i_q}\}$ forms an exact cover of S. Then, the tree in Fig. 4 obviously has a cost of at most $2(m - q)$.

Now suppose that the **FCC** instance has a solution tree T of cost $2(m - q)$. Let $(S_{i_1}, S - S_{i_1}), \ldots, (S_{i_p}, S - S_{i_p})$ be the bipartitions that are contained in T (i.e., each of them is induced by some edge of T) and $(S_{i_{p+1}}, S - S_{i_{p+1}}), \ldots, (S_{i_{p+r}}, S - S_{i_{p+r}})$ the bipartitions with distance 1 from T. Then

$$2q = 2p + r.$$

Clearly, the subsets $S_{i_1}, \ldots, S_{i_p}$ are pairwise disjoint. Moreover, for each j, where $1 \leqslant j \leqslant r$, since the bipartition $(S_{i_{p+j}}, S - S_{i_{p+j}})$ has distance 1 from the tree T, T must contain

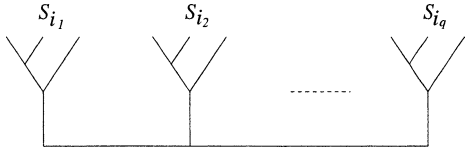

Fig. 4. The optimal tree for **FCC** corresponding to an exact cover of S.

two of the three elements of $S_{i_{p+j}}$ as siblings. Observe that because $|S_{i_{p+j}} \cap S_{i_k}| \leqslant 1$ for any $1 \leqslant k \leqslant p$, none of these two sibling elements is contained in the subset S_{i_k} for any $1 \leqslant k \leqslant p$. Since $|S_{i_{p+j}} \cap S_{i_{p+k}}| \leqslant 1$ for any $j \neq k$ and T is fully resolved (i.e., T is degree-3), the r pairs of sibling elements corresponding to the subsets $S_{i_{p+1}}, \ldots, S_{i_{p+r}}$ must be pairwise disjoint. Hence,

$$r \leqslant 3(q-p)/2.$$

Therefore, we have $2(q-p) \leqslant 3(q-p)/2$, which implies $p = q$. That is, $S_i, \ldots, S_{i_p}$ form an exact cover of S. □

4. A PTAS for FCC

Let R be an instance of **FCC** with label set S where $n = |S|$ and $m = |R|$. Let T_{OPT} denote an optimal solution for this instance. Our main result is a polynomial time approximation scheme that, for any $\epsilon > 0$, produces a tree T_{APX} in polynomial time such that

$$s(T_{\mathrm{APX}}, R) \geqslant (1-\epsilon)(T_{\mathrm{OPT}}, R).$$

This result requires the caveat that R contains $\Theta(n)$ bipartitions. That this requirement emerges from the analysis is not surprising since a tree T on n leaves has $n-3$ nontrivial bipartitions. Now R is a set of bipartitions from which T is to be recovered. Then $\Omega(n)$ bipartitions are required to recover the nontrivial bipartitions of T and we expect that $O(n)$ bipartitions are actually informative. Consequently, the requirement that $|R| = \Theta(n)$ is quite natural.

Our PTAS for **FCC** utilizes two concepts: k-bin contractions and smooth polynomial integer programs. These two concepts and their relationship to **FCC** are overviewed below. They will be examined in detail in Sections 4.1 and 4.2.

Definition 4.1. A tree T_k is a *k-bin contraction* of T_{OPT} if there is a partition of S into *bins* $S_1, S_2, \ldots, S_k$ such that

- for each S_i, $|S_i| \leqslant 6n/k$. Furthermore, there is a vertex v_i in T_k of degree $|S_i|+1$, called the *bin root*, that is adjacent to each vertex in S_i;
- for each edge e of T_{OPT} there is an edge e' of T_k such that $s(e, e') \geqslant n - 6n/k$.

In Section 4.1 it is shown that there is a k-bin contraction T_k of T_{OPT} that is a good approximation of T_{OPT}, i.e, $s(T_k, R) \geqslant (T_{\mathrm{OPT}}, R) - (6/k)mn$ where $m = |R|$ and $n = |S|$. Our approach to solving **FCC** is to approximate T_{OPT} indirectly by approximating T_k. The approximation of T_k is described below.

Fix k and let K be T_k with all leaves removed (and thus the leaves of K are the bin roots of T_k). K is called the *kernel* of T_k and T_k is called a *completion* of K. K is completed by providing a label-to-bin assignment: labels in S are assigned as children to bin roots of K (see Fig. 5).

An example of a k-bin contraction appears in Fig. 5.

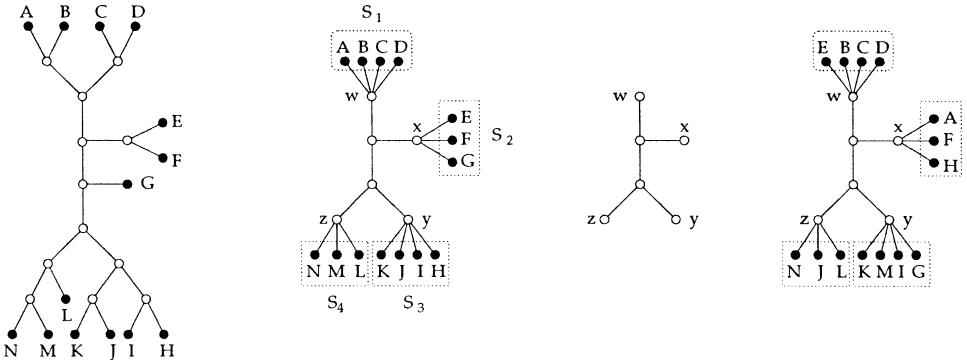

Fig. 5. From left to right: a tree T; a 4-bin contraction T_4 of T with bin roots w, x, y, and z; the kernel K of T_4; and a completion of K.

If the kernel K of T_k is known then, to approximate T_k, it suffices to determine an approximately optimal label-to-bin assignment for K. This problem is formalized as follows:

Label-to-Bin Assignment (LBA)
Instance: Set R of bipartitions of S and a binary kernel K with k leaves.
Problem: Find a completion T' of K that maximizes $s(T', R)$.

In Section 4.2, **LBA** is formulated as a smooth polynomial integer problem and a PTAS for **LBA** is presented. In particular, it is shown that, for any $\epsilon > 0$, $s(T', R) \geqslant s(\widehat{T}, R) - \epsilon(m+n)^2$, where R and K denote the instance of **LBA**, T' is the completion of K produced by our PTAS and $\widehat{T}$ is an optimal completion of K.

The approximation algorithm for **FCC** proceeds by solving (approximately) an instance of **LBA** for every kernel with k leaves. Since k is a constant, this can be done in polynomial time using the PTAS for **LBA**. Let T_{APX} be the completed tree obtained that maximizes $s(T_{\mathrm{APX}}, R)$. Since the kernel K of T_k is one of the trees completed, it follows that $s(T_{\mathrm{APX}}, R) \geqslant s(T', R)$ where T' is the completion of K obtained by the PTAS for **LBA**. The approximation algorithm for **FCC** is summarized in Section 4.3.

4.1. Contracting the optimal tree

An algorithm called *k-Bin Contraction* appears below. This algorithm first appeared as an algorithm for obtaining k-bin decompositions [15] but is adapted here to generate a k-bin contraction of T_{OPT}.

k-Bin Contraction finds a k-bin contraction T_k of T_{OPT}. For convenience, let $T = T_{\mathrm{OPT}}$ and assume that T is binary, without loss of generality.

Algorithm *k-Bin Contraction(T)*.

1. Root T at an arbitrary internal vertex. Let $T(v)$ denote the subtree of T rooted at v.
2. Traverse T, beginning at the root, such that for each vertex v visited:

If $|T(v)| < 6n/k$ then

- contract all internal edges of $T(v)$,
- label v as a bin root and
- continue traversal at v's parent.

Otherwise, continue traversal at an unvisited child of v.

3. For each bin root v with parent v' and sibling u':

If $|T(v)| \leqslant 3n/k$ and u' has a child u with $|T(u)| \leqslant 3n/k$ then

- transfer the leaves in $T(u)$ to the bin of v,
- contract $\{u, u'\}$, and
- contract $\{u', v'\}$.

4. For each leaf u of T not assigned to a bin, bisect the edge between u and its parent with a new vertex v, and mark v as a bin root.

The last step of the algorithm is necessary since a leaf cannot be a bin root.

Theorem 4.1. *There is a k-bin contraction of T_{OPT}.*

Proof. Let T_k be the tree produced by the k-bin contraction algorithm. It is shown that T_k satisfies both properties that a k-bin decomposition must satisfy (see Definition 4.1).

Property 1 (*Bounded Bin Size*). Call a bin of T_k *small* if it has size less than $3n/k$. A bin root is *small* if its bin is small. Call a bin of T_k is *large* if it has size between $3n/k$ and $6n/k$, inclusive. A bin root is *large* if its bin is large. Let *sbin* denote the number of small bins in T_k and *lbin* the number of large bins in T_k.

The following lemma is required:

Lemma 4.1 [15]. *$sbin < 2lbin$.*

Proof. We prove the following by induction: For every h, if u is a vertex of height h and is not a bin root then the lemma holds for the subtree $T(u)$.

For the base case, assume that u has children p and q. It cannot be that both p and q are small otherwise $|T(u)| < 6n/k$ and the algorithm would not have visited p and q. Hence, the lemma is true for $T(u)$.

In general, assume that u has children p and q. If both p and q are bin roots then the argument for the base case applies. If neither p nor q are bin roots then the inductive hypothesis applies to $T(p)$ and $T(q)$, and hence the lemma holds for $T(u)$. Otherwise, suppose that p is not a bin root but q is.

If q is large then the induction can be applied to $T(p)$ and the claim follows. Otherwise, suppose that q is small. Let p_1 and p_2 be the children of p. Neither p_1 nor p_2 are small, since otherwise the algorithm would have merged one of $T(p_1)$ and $T(p_2)$ with $T(q)$ in step 3. By the inductive hypothesis, $sbin_1 < 2lbin_1$ and $sbin_2 < 2lbin_2$ where $sbin_1$, $lbin_1$, $sbin_2$, and $lbin_2$ are the numbers of small and large bins in $T(p_1)$ and $T(p_1)$, respectively. It follows that $sbin_1 + sbin_2 + 1 < 2(lbin_1 + lbin_2)$, and hence the claim holds for $T(u)$. □

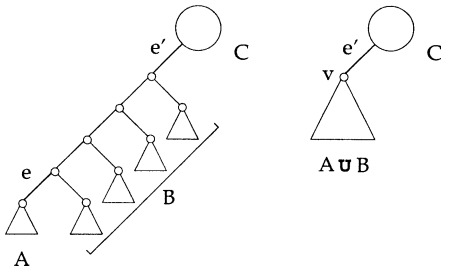

Fig. 6. Case 1.

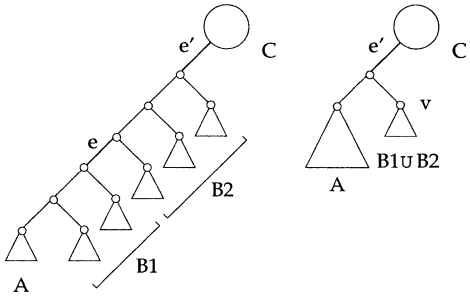

Fig. 7. Case 2.

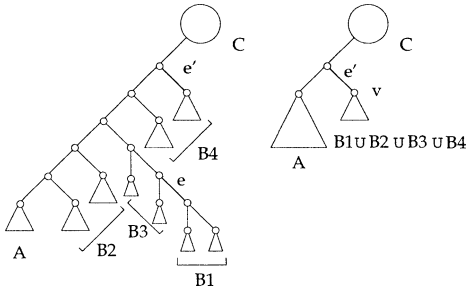

Fig. 8. Case 3.

Since each large bin of T_k has size at least $3n/k$, $lbin \leqslant k/3$. T_k has $lbin + sbin$ bins, and so, $lbin + sbin < lbin + 2lbin = 3lbin \leqslant k$, by Lemma 4.1. We conclude that T_k has less than k bins each with size bounded by $6n/k$.

Property 2 (*Edge Preservation*). Let e be an edge of T_{OPT}. If the algorithm does not contract e then there is an edge e' in T_k such that $s(e, e') = n$ and we are done. Hence, now consider only edges of T_{OPT} that have been contracted.

There are three cases to be considered. These three cases are depicted in Figs. 6–8.

In the first case, e is part of a subtree that is contracted to form a bin with bin root v. Let e' be the edge from v to v' parent in T_k. Referring to Fig. 6, $e = (A, B \cup C)$ and $e' = (A \cup B, C)$. Since $|B| \leqslant 6n/k$, it follows that $s(e, e') \geqslant n - 6n/k$.

In the second case, e is an edge, as depicted in Fig. 7, that is contracted in step 3 of the algorithm to form a bin with bin root v. Let e' be the edge from the parent of v to the grandparent of v. Referring to Fig. 7, $e = (A \cup B_1, B_2 \cup C)$ and $e' = (A \cup B_1 \cup B_2, C)$. Since $|B_2| \leqslant |B_1 \cup B_2| \leqslant 6n/k$, it follows that $s(e, e') \geqslant n - 6n/k$.

In the third case, e is an edge, as depicted in Fig. 8, that is contracted in step 3 of the algorithm to form a bin with bin root v. Let e' be the edge from v to the parent of v. Referring to Fig. 8, $e = (A \cup B_2 \cup B_3 \cup B_4 \cup C, B_1)$ and $e' = (A \cup C, B_1 \cup B_2 \cup B_3 \cup B_4)$. Since $|B_2 \cup B_3| \leqslant |B_1 \cup B_2 \cup B_3 \cup B_4| \leqslant 6n/k$, it follows that $s(e, e') \geqslant n - 6n/k$. □

We also need that a k-bin contraction of T_{OPT} be a good approximation of T_{OPT}:

Theorem 4.2. *There is a k-bin contraction T_k of T_{OPT} such that $s(T_k, R) \geqslant s(T_{\mathrm{OPT}}, R) - (6/k)mn$ where R is an instance of **FCC** with label set S, $m = |R|$, and $n = |S|$.*

Proof. Let T_k be a k-bin contraction of T_{OPT}. Then for each edge e' of T_{OPT} there is an edge e' of T_k such that $s(e, e') \geqslant n - 6n/k$. It follows that for each $(A, B) \in R$, $s(T_k, (A, B)) \geqslant s(T_{\mathrm{OPT}}, (A, B)) - 6n/k$. As there are m bipartitions in R, $s(T_k, R) \geqslant s(T_{\mathrm{OPT}}, R) - (6/k)mn$. □

4.2. A PTAS for LBA

Let $R = \{(A_i, B_i) \mid 1 \leqslant i \leqslant m\}$, S and K be an instance of the **LBA** problem where K has k leaves. $\widehat{T}$ is an optimal completion of K if $\widehat{T}$ maximizes $s(\widehat{T}, R)$, over all completions of K. To formalize this optimization problem *smooth polynomials* are used. A degree d polynomial $p(x)$ is *t-smooth*, where t is a constant, if the coefficient of each degree i term is in the interval $[-tn^{d-i}, tn^{d-i}]$, for $0 \leqslant i \leqslant d$.

Define the 0–1 label-to-bin assignment with variables $x = (x_{vb})$ as follows:

$x_{vb} = 1$ if label v is assigned to bin b, otherwise $x_{vb} = 0$.

To ensure that each label $v \in S$ is assigned to one bin the following constraints are created for each label v:

$$\sum_b x_{vb} = 1.$$

In order to evaluate $s(T', R)$, where T' is the completion of K specified by the label-to-bin assignment $x = (x_{vb})$, it is necessary to assign bipartitions (A_i, B_i) in R to edges e of K so that $s(e, (A_i, B_i))$ can be evaluated. To this end, define the 0–1 bipartition-to-edge assignment with variables $y = (y_{ie})$ as follows:

$y_{ie} = 1$ if (A_i, B_i) is assigned to edge e of T, otherwise $y_{ie} = 0$.

To ensure that each bipartition is assigned to one edge the following constraints are created for each bipartition (A_i, B_i):

$$\sum_e y_{ie} = 1.$$

Create the following polynomial for each $(A_i, B_i) \in R$:

$$p_i(x, y) = \sum_{e=(A,B)\in K} y_{ie}\left(\sum_{v\in A_i}\sum_{a\in A} x_{va} + \sum_{v\in B_i}\sum_{b\in B} x_{vb}\right).$$

Since $y_{ie} = 1$ for exactly one edge $e = (A, B) \in K$,

$$p_i(x, y) = \sum_{v\in A_i,\ a\in A} x_{va} + \sum_{v\in B_i,\ b\in B} x_{vb} = s\big(e, (A_i, B_i)\big).$$

Finally, define $p(x, y) = \sum_{1\leqslant i\leqslant m} p_i(x, y)$. The optimal value for $p(x, y)$ is then $s(\widehat{T}, R)$.

In summary, our optimization problem is to find a 0–1 label-to-bin assignment x and a 0–1 bipartition-to-edge assignment y so that

$$\begin{aligned}
&p(x, y) \text{ is maximized,}\\
&\sum_b x_{vb} = 1, \quad \text{for each label } v,\\
&\sum_e y_{ie} = 1, \quad \text{for each bipartition } (A_i, B_i),\\
&\sum_v x_{vb} \leqslant 6n/k, \quad \text{for each bin } b.
\end{aligned}$$

Observe that $p(x, y)$ is a degree 2 polynomial and it is 1-smooth. Furthermore, $p(x, y)$ is defined on $O(n + m)$ variables.

Arora et al. [2] present a PTAS for solving t-smooth integer polynomial programs:

Theorem 4.3 [2]. *For each $\epsilon > 0$ there is a polynomial time algorithm that produces a* 0–1 *assignment z such that $p(z) \geqslant p(z^*) - \epsilon n^d$ where $p(z)$ is a t-smooth polynomial defined on n variables with degree d and with maximum value $p(z^*)$.*

Following from the above discussion and Theorem 4.3, we have

Theorem 4.4. *For each $\epsilon > 0$, there is a polynomial time algorithm that, for each instance R and K of **LBA**, produces a completion T' of K such that $s(T', R) \geqslant s(\widehat{T}, R) - \epsilon(n+m)^2$ where $\widehat{T}$ is an optimal completion of K.*

4.3. Summary of the approximation algorithm and an interesting caveat

We summarize the PTAS that approximates an optimal solution T_{OPT} to an instance R of **FCC**. For each kernel K on k leaves (for sufficiently large k) an approximately optimal completion of K is found using the PTAS for **LBA**. Let T_{APX} be the completion

that maximizes $s(T_{\mathrm{APX}}, R)$. It follows that $s(T_{\mathrm{APX}}, R) \geqslant s(T', R)$ where T' is the approximately optimal completion of the kernel K of T_k and T_k, is a k-bin contraction of T_{OPT}.

By Theorem 4.4, $s(T', R) \geqslant s(\widehat{T}, R) - \epsilon(n+m)^2$ where $\widehat{T}$ is the optimal completion of K, for any $\epsilon > 0$. We also have that $s(\widehat{T}, R) \geqslant s(T_k, R)$ and $s(T_k, R) \geqslant s(T_{\mathrm{OPT}}, R) - (6/k)mn$ by Theorem 4.2. Thus we have

$$s(T_{\mathrm{APX}}, R) \geqslant s\big(\widehat{T}, R\big) - \epsilon(m+n)^2 \geqslant s(T_{\mathrm{OPT}}, R) - (6/k)mn - \epsilon(m+n)^2.$$

At this point a lower bound on $s(T_{\mathrm{OPT}}, R)$ is established using the following lemma:

Lemma 4.2. *If (A, B) and (C, D) are bipartitions of S where $n = |S|$ then $s((A, B), (C, D)) \geqslant n/2$.*

Proof. Since, $|A \cap C| + |A \cap D| + |B \cap C| + |B \cap D| = n$, either $|A \cap C| + |B \cap D| > n/2$ or $|A \cap D| + |B \cap C| \geqslant n/2$. □

Lemma 4.2 leads directly to the lower bound $s(T_{\mathrm{OPT}}, R) \geqslant mn/2$. Now our caveat, if $m = \Theta(n)$ then the following theorem is immediate:

Theorem 4.5. *For each $\epsilon > 0$, there is a polynomial time algorithm that, for each instance R of **FCC**, produces a tree T_{APX} such that $s(T_{\mathrm{APX}}, R) \geqslant (1-\epsilon)s(T_{\mathrm{OPT}}, R)$.*

The requirement that $m = \Theta(n)$ is natural. Since an evolutionary tree has $n - 3$ nontrivial edges, it is expected that only $O(n)$ estimated bipartitions will actually be informative.

5. Conclusions

This paper introduces the **FCC** criterion for branch selection, justifies the use of this criterion and then provides a PTAS for solving the branch selection problem using the **FCC** criterion. Some final comments are warranted:

- A weighted version of the **FCC** criterion can be obtained by assigning a weight to each bipartition. Weights arise naturally in practice:
  - Hypercleaning produces a list of bipartitions ranked by support. It is then preferable to select bipartitions with the highest weighted **FCC** score.
  - Given a collection of estimates $T_1, T_2, \ldots, T_k$ of the unknown evolutionary tree T, the weight of a bipartition (X, Y) is the percentage of these estimates containing (X, Y). The weighted **FCC** score would prefer high frequency bipartitions.

The PTAS presented here can easily be extended to the weighted version of **FCC**.

- The results presented here are algorithmic in nature. To obtain a practical tool for branch selection, further work is required. First, efficient implementations of algorithms for approximating smooth polynomial integer programs are required in order for the PTAS to be effective. Second, the **FCC** criterion must be demonstrated as being biologically relevant through experimentation or simulation.

Acknowledgments

The authors thank the referees for suggested improvements to the paper.

References

[1] R. Agarwala, D. Fernández-Baca, A polynomial–time algorithm for the perfect phylogeny problem when the number of character states is fixed, SIAM J. Comput. 23 (6) (1994) 1216–1224.
[2] S. Arora, C. Lund, R. Motwani, M. Sudan, M. Szegedy, Proof verification and hardness of approximation problems, in: Proceedings of the Thirty-Third IEEE Symposium on the Foundations of Computer Science, 1992, pp. 14–23.
[3] J.-P. Barthélemy, A. Guénoche, Trees and Proximity Representations, Wiley, New York, 1991.
[4] V. Berry, O. Gascuel, Inferring evolutionary trees with strong combinatorial evidence, in: Proceedings of the Third Annual International Computing and Combinatorics Conference, 1997, pp. 111–123.
[5] D. Bryant, V. Berry, T. Jiang, P. Kearney, M. Li, T. Wareham, H. Zhang, A practical algorithm for recovering the best supported edges of an evolutionary tree, in: Proceedings of the llth Annual ACM–SIAM Symposium on Discrete Algorithms, 2000, pp. 287–296.
[6] W.H.E. Day, Inferring phytogenies from dissimilarity matrices, Bull. Math. Biol. 49 (4) (1987) 461–467.
[7] P. Erdös, M. Steel, L. Székely, T. Warnow, Constructing big trees from short sequences, in: Proceedings of the 24th International Colloquium on Automata, Languages, and Programming, 1997.
[8] G.F. Estabrook, F.R. McMorris, When are two qualitative taxonomic characters compatible? J. Math. Biol. 4 (1977) 195–200.
[9] J. Felsenstein, Evolutionary trees from DNA sequences: A maximum likelihood approach, J. Mol. Evol. 17 (1981) 368–376.
[10] W.M. Fitch, Toward defining the course of evolution: Minimal change for a specific tree topology, Syst. Zool. 20 (1971) 406–441.
[11] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman, New York, 1979.
[12] D. Gusfield, Efficient algorithms for inferring evolutionary trees, Networks 21 (1991) 19–28.
[13] M. Fellows, H. Bodlaender, T. Warnow, Two strikes against perfect phylogeny, in: Proceedings of the 19th International Colloquium on Automata, Languages, and Programming, in: Lecture Notes in Comput. Sci., 1992, pp. 273–283.
[14] D. Hillis, J. Huelsenbeck, D. Swofford, Hobgoblin of phylogenetics? Nature 369 (1994) 363–364.
[15] T. Jiang, P.E. Kearney, M. Li, Orchestrating quartets: Approximation and data correction, in: Proceedings of the 39th IEEE Symposium on Foundations of Computer Science, 1998, pp. 416–425.
[16] L.B. Maidak, J.R. Cole, C.T. Parker, G.M. Garrity Jr., N. Larsen, B. Li, T.G. Lilburn, M.J. McCaughey, G.J. Olsen, R. Overbeek, S. Pramanik, T.M. Schmidt, J.M. Tiedje, C.R. Woese, A new version of the RDP (Ribosomal Database Project), Nucleic Acids Research 27 (1999) 171–173.
[17] C.A. Meacham, G.F. Estabrook, Compatibility methods in systematics, Ann. Rev. Ecol. Syst. 16 (1985) 431–446.
[18] N. Saitou, M. Nei, The neighbor-joining method: A new method for reconstructing phylogenetic trees, Mol. Biol. Evol. 4 (1987) 406–425.
[19] B. Snel, P. Bork, M.A. Huynen, Genome phylogeny based on gene content, Nat. Genet. 21 (1999) 108–110.
[20] K. Strimmer, A. von Haeseler, Quartet puzzling: A quartet maximum-likelihood method for reconstructing tree topologies, Mol. Biol. Evol. 13 (7) (1996) 964–969.
[21] D.L. Swofford, G.J. Olsen, P.J. Waddell, D.M. Hillis, Phylogenetic inference, in: D.M. Hillis, C. Moritz, B.K. Mable (Eds.), Molecular Systematics, 2nd edition, Sinauer Associates, Sunderland, MA, 1996, pp. 407–514.