Advances in Applied Mathematics **27,** 705–732 (2001)
doi:10.1006/aama.2001.0758, available online at http://www.idealibrary.com on IDEAL®

A Structured Family of Clustering and Tree Construction Methods

David Bryant

LIRMM, Montpellier, France; and Departments of Mathematics and Computer Science, McGill University, Canada
E-mail: bryant@math.mcgill.ca

and

Vincent Berry

LIRMM, Montpellier, France
E-mail: vberry@lirmm.fr

Received January 24, 2001; accepted May 30, 2001

A cluster A is an Apresjan cluster if every pair of objects within A is more similar than either is to any object outside A. The criterion is intuitive, compelling, but often too restrictive for applications in classification. We therefore explore extensions of Apresjan clustering to a family of related hierarchical clustering methods. The extensions are shown to be closely connected with the well-known single and average linkage tree constructions. A dual family of methods for classification by splits is also presented. Splits are partitions of the set of objects into two disjoint blocks and are widely used in domains such as phylogenetics. Both the cluster and split methods give rise to progressively refined tree representations. We exploit dualities and connections between the various methods, giving polynomial time construction algorithms for most of the constructions and NP-hardness results for the rest.

Key Words: clustering; splits; tree construction; algorithms; classification; compatibility; quartets; hierarchies; single linkage tree; average linkage tree.

1. INTRODUCTION

A cluster is a subset of a collection of objects. A split is a separation of the collection into two disjoint parts. We will be investigating methods for classification using clusters and methods usings splits, repeatedly exploiting

0196-8858/01 $35.00

connections between the two approaches. We start with two classification methods that are now almost classical: Apresjan clusters [1] and Buneman splits [10]. We will show how these closely related constructions can be extended and refined to give a highly structured family of clustering and "splitting" classification methods.

The aim of almost every clustering method is to cluster objects together that are more similar to one another than they are to objects outside the cluster. We represent the level of similarity between elements of a set X using a **similarity function** s: $X \times X \to \Re$ which we define to be any symmetric function from pairs of objects to $\Re$. The intuitive notion of clustering is captured in the definition of Apresjan clusters [1], which is the first clustering method we investigate. Following Bandelt and Dress [2], we define Apresjan clusters in terms of similarity functions, rather than dissimilarities, to emphasize the mathematical duality between the clustering and the splitting constructions. All of the clustering methods we describe here can be modified to handle dissimilarity data.

Define the **Apresjan clusters** of a similarity function s to be $\{A : A \subseteq X, \iota_s(A) > 0\}$, where $\iota_s(A)$ is the strong isolation index

$$\iota_s(A) := \min_{a,a',x} \{s(a, a') - s(a, x) : a, a' \in A, x \in X - A\}. \tag{1}$$

The construction was first studied by Parker-Rhodes and Needham [22] who called the clusters K-clumps. Apresjan clusters are also known as L-groups, K-groups, and strong clusters [2, 14, 18].

The Apresjan clusters of a similarity function form a **(strong) hierarchy**, in the sense that if two Apresjan clusters intersect then one contains the other. If we have that $s(a, a) > s(a, b)$ for all $a \neq b$, as is usually the case, then all singleton clusters are Apresjan and we obtain a proper hierarchy. The clusters can then be represented using a rooted tree T with leaves labelled by members of X. Every node v in T is associated with the cluster $C(v)$, the set of all leaves that are descendents of v. For each node v we assign the length $\iota_s(C(v))$ to the edge connecting v and its parent node.

If $s(a, a)$ is constant for all $a \in X$ then the rooted tree will be a dendrogram, with all leaves the same distance from the root. Furthermore, if s satisfies the (sign reversed) ultrametic inequality

$$s(a, b) \geq \min\{s(a, c), s(b, c)\}$$

for all a, b, c then the tree T will be the dendrogram corresponding to s [2].

Buneman introduced a related classifcation method in a paper on the filation history of ancient manuscripts [10]. The input for Buneman's method is a **dissimilarity function** δ: $X \times X \to \Re^+$, and the basic grouping unit is a **split** (bipartition $A|B$ of X) rather than a cluster. The **Buneman splits** of

δ are $\{A|B : \mu_\delta(A|B) > 0\}$, where μ_δ is the **strong separation index**

$$\mu_\delta(A|B) = \frac{1}{2}\min\big\{\big(\delta(a,b)+\delta(a',b')\big) - \big(\delta(a,a')+\delta(b,b')\big)\colon a,a'\in A,\ b,b'\in B\big\}. \quad (2)$$

An X-**tree** is an *unrooted* tree $T = (V, E)$ with labelling $L\colon X \to V$ such that every node of degree less than three is labelled. The Buneman splits can be represented using a valued X-tree. If we remove any edge e in an X-tree we divide the tree into two components and induce a split $S(e)$ of X given by the label sets of the two components. Buneman showed that there is an X-tree T such that the splits $\{S(e) : e \in E\}$ are exactly the Buneman splits for δ. We say that T is the **Buneman tree** for δ. We can construct a valued X-tree by assigning the length $\mu_s(A|B)$ to the edge corresponding to $A|B$. In this way we obtain a mapping from dissimilarity functions to valued X-trees, which can be computed in polynomial time with respect to $|X|$. The separation indices $\mu_\delta(A\,|\,B)$ are continuous, so a small change in δ will induce only a small change in the tree. One can show that if δ is additive, that is,

$$\delta(a,b)+\delta(c,d) \le \max\{\delta(a,c)+\delta(b,d),\ \delta(a,d)+\delta(b,c)\},$$

for all a, b, c, d, then the tree T will be the valued X-tree corresponding to δ.

The methods of Apresjan and Buneman are quite conservative, only returning groupings with a lot of support in the data. With Apresjan clusters, the condition that $\iota_s(A) > 0$ is very strong: for *every* choice of $a, a' \in A$ and $x \in X - A$ we must have $s(a, a') > s(a, x)$. A single "errant" similarity would exclude the cluster from the collection. Similarly, the condition for Buneman splits that $\mu_\delta(A|B) > 0$ is also very strong, with a small pertubation in the input dissimilarity potentially excluding all splits from the Buneman tree. Consequently, when the similarity or dissimilarity data are not sufficiently "tree-like," the methods produce poorly resolved trees with high degree internal nodes and only a few small clusters or splits.

Both methods are continuous, in the sense that a small change in the input will induce only a small change in the final tree. This, as well as the ability to construct the corresponding trees in polynomial time and invariance on tree-like dissimilarities (i.e., respectively ultrametric and additive), makes the two methods attractive for numerous applications such as phylogenetics. The uninformative output tree is still a shortcoming. The lack of resolution in the Buneman tree motivated Moulton and Steel [20] to investigate modifications of the Buneman tree construction. They defined a new separation index μ^r_δ such that $\mu^r_\delta(A|B) \ge \mu_\delta(A|B)$ for all splits $A|B$ and yet the collection of splits $\{A|B : \mu^r_\delta(A|B) > 0\}$ could still be represented by an X-tree. The new separation index leads to a tree construction

method that is continuous, polynomial time, and invariant on additive dissimilarities and gives trees that contain all of the Buneman splits and often additional splits. The tree produced is called the **refined Buneman tree**. A tree T' **refines** a tree T if every split induced by an edge of T is also induced by some edge of T'.

In this paper we extend the work of Moulton and Steel [20] in several directions. Our goal is to derive clustering and tree construction methods that are continuous, polynomial time, and informative. To this end we present a family of methods containing and extending the clustering method of Apresjan and the tree construction method of Buneman. We explore and exploit connections between the methods, formalise the connection between Buneman splits and Apresjan clusters, and extend this duality to the new constructions. We prove that Apresjan clusters are contained in the single linkage tree (Theorem 2.3), and that our extensions to Apresjan clusters are contained in the average linkage tree (Theorem 2.4). We use NP-hardness proofs to indicate the limit of this series of refinements and extensions.

We also extend the Apresjan and Buneman constructions so that they can accept different kinds of input. One can rewrite the definition of the strong isolation index (Eq. (1)) as

$$\iota_s(A) = \min\{w(aa'|x) : a, a' \in A, x \in X - A\}, \tag{3}$$

where $w(aa'|x) = s(a, a') - \max\{s(a, x), s(a', x)\}$. We can now replace w with another function. Different weighting functions w give different clustering methods. We explore the conditions that the weighting function w has to satisfy for the resulting clusters to form a hierarchy, and we introduce equivalent extensions for splits and unrooted tree constructions. The ability to use more general weighting functions greatly increases the flexibility and utility of these methods.

The algorithms have been implemented in C++. Source code will be made available by the authors or online at http://www.math.mcgill.ca/~bryant.

2. A FAMILY OF CLUSTERING METHODS

2.1. *Definitions*

A **rooted X-tree** is a rooted tree $T = (V, E)$, with nodes V and edges E, together with a labelling function $L: X \to V$ such that every node with less than two children is labelled. Put $n = |X|$. Given two nodes u and v we write $u \prec_T v$ if u is a strict descendent of v in T. The **cluster associated to** v is the set of labels on nodes that are descendents of v (including v) and

is denoted $C(v)$. The **clusters of** T are $clus(T) = \{C(v) : v \in V\}$. A cluster A is **trivial** if $|A| = 1$ or $A = X$. We use $\text{lca}(x, y)$ to denote the node that is the least common ancestor of the nodes labelled by x and y.

A set of clusters $\mathscr{C}$ is **compatible** if $\mathscr{C} = clus(T)$ for some rooted X-tree T, which holds if and only if $\mathscr{C}$ forms a hierarchy. It is a classical result that $\mathscr{C}$ is compatible if and only if for each pair $A, B \in \mathscr{C}$ one of $A - B$, $B - A$, $A \cap B$ is empty (e.g., [10]).

A **rooted triple** $ab|c$ denotes a grouping of a and b relative to c. The set of all rooted triples of X is denoted $\mathscr{R}(X)$. The set of rooted triples of a cluster $A \subseteq X$ is defined as

$$r(A) = \{uv|z : u, v \in A,\ z \in X - A\}.$$

We say that $ab|c$ is a rooted triple of T if there is a cluster $A \in clus(T)$ such that $ab|c \in r(A)$. The set of rooted triples of T is denoted $r(T)$.

A **similarity function** is a symmetric function from $X \times X$ to $\Re$. An **isolation weighting** is a function w: $\mathscr{R}(X) \longrightarrow \Re$ such that $w(ab|c) + w(ac|b) \leq 0$ for all $a, b, c \in X$. The weight $w(ab|c)$ indicates the degree to which c is excluded the group $\{a, b\}$. Given a similarity function s define

$$\rho_s(ab|c) = s(a, b) - \max\{s(a, c), s(b, c)\}. \tag{4}$$

Then ρ_s is an isolation weighting.

Given a non-empty set R of rooted triples and an isolation weighting w, let $\text{av}(R)$ denote the average weight of the triples in R and let $\text{avmin}(R, k)$ denote the average weight of the $k \leq |R|$ triples with minimum weight in R. Formally,

$$\text{av}(R) = \frac{1}{|R|} \sum_{xy|z \in R} w(xy|z) \tag{5}$$

$$\text{avmin}(R, k) = \min_{R'}\{\text{av}(R') : R' \subseteq R,\ |R'| = k\}. \tag{6}$$

Given disjoint non-empty U, V, Z define

$$\overline{w}(UV|Z) = \text{av}\{uv|z : u \in U, v \in V, z \in Z\}, \tag{7}$$

the average weight over all triples $uv|z$ with $u \in U$, $v \in V$, and $z \in Z$.

2.2. *A Series of Clustering Methods*

We investigate a family of hierarchical clustering methods. In each case, the clusters are selected according to an explicit criterion. The criteria used by the three methods are closely related and are all based on different definitions of an *isolation index* for selecting clusters of X. Here, and throughout the paper, we use $U \uplus V$ to denote the union of disjoint sets U and V.

DEFINITION OF CLUSTERING INDICES. Let w be an isolation weighting and let A be a cluster of X.

(i) The **strong isolation index** of A is defined

$$\iota_w(A) = \min_{uv|z}\{w(uv|z) : uv|z \in r(A)\} = \text{avmin}(r(A), 1)$$

and the **strong clusters** of w are $\{A : \iota_w(A) > 0\}$.

(ii) The **clean cluster index** of A is

$$\iota^c_w(A) = \text{avmin}(r(A), |A| - 1)$$

and the **clean clusters** of w are $\{A : \iota^c_w(A) > 0\}$.

(iii) The **stability index** of A is

$$\iota^s_w(A) = \min_{U,V,Z\neq\emptyset}\{\overline{w}(UV|Z) : A = U \uplus V, Z \subseteq X - A\}$$

and the **stable clusters** of w are $\{A : \iota^s_w(A) > 0\}$.

Define the strong clusters (Apresjan clusters) of a similarity function s to be the strong clusters of $w = \rho_s$, and similarly for clean clusters and stable clusters of a similarity function.

The name "clean clusters" stems from a connection with the quartet cleaning construction of [6] (see Section 3.5). The indices, and the construction complexities, for all of the clustering methods are summarised in Table I. The inclusion relations between the various clustering indices and methods are presented in Theorem 2.1. The following lemma is used in both compatibility and inclusion proofs.

LEMMA 2.1. *If U, V, Z are nonempty subsets of X such that $A = U \uplus V$ and $Z \subseteq X - A$ then $|U||V||Z| \geq |A| - 1$.*

TABLE I
A Summary of the Cluster Indices and Construction Complexities for the Family of Clustering Methods

Construction	Index	Complexity
Strong clusters	$\iota_w(A) = \min_{uv\|z}\{w(uv\|z) : uv\|z \in r(A)\}$	$O(n^2)$ (sim.[a])
		$O(n^3)$ (iso. wt.[b])
Clean clusters	$\iota^c_w(A) = \text{avmin}(r(A), \|A\| - 1)$	$O(n^3)$ (sim.)
		$O(n^4)$ (iso. wt.)
Stable clusters	$\iota^s_w(A) = \min_{U,V,Z}\{\overline{w}(UV\|Z) : A = U \uplus V, Z \subseteq X - A\}$	NP-hard

[a] Complexity when input is a similarity function.
[b] Complexity when input is an isolation weighting.

Proof. Put $p = |V| = |A - U|$. Then

$$|U||V||Z| = (|A| - p)p|Z| \tag{8}$$

$$\geq (|A| - p)p, \tag{9}$$

since $|Z| \geq 1$. By straightforward calculus we have $(|A| - p)p \geq (|A| - 1)$, giving the result. ∎

We now show that the clustering methods form a series in the sense of inclusion relations for the resulting sets of clusters.

THEOREM 2.1. *For any cluster A we have $\iota_w(A) \leq \iota_w^c(A) \leq \iota_w^s(A)$. Hence every strong cluster of w is a clean cluster of w and every clean cluster is a stable cluster.*

Proof. First note that if $k \leq k'$ then $\text{avmin}(r(A), k) \leq \text{avmin}(r(A), k')$. We therefore have

$$\iota_w(A) = \text{avmin}(r(A), 1) \leq \text{avmin}(r(A), |A| - 1) = \iota_w^c(A).$$

Choose nonempty subsets U, V, and Z such that $A = U \uplus V$, $Z \subseteq X - A$ and $\overline{w}(UV|Z) = \iota_w^s(A)$. By Lemma 2.1, $|U||V||Z| \geq |A| - 1$ and so

$$\begin{aligned}\iota_w^c(A) &= \text{avmin}(r(A), |A| - 1)\\ &\leq \text{avmin}(r(A), |U||V||Z|)\\ &\leq \overline{w}(UV|Z)\\ &= \iota_w^s(A).\end{aligned}$$

∎

Having demonstrated the inclusion relationships between the strong clusters, clean clusters, and stable clusters, we now establish compatibility.

THEOREM 2.2. *Let w be an isolation weighting. The set of strong clusters of w, the set of clean clusters of w, and the set of stable clusters of w are all compatible collections of clusters.*

Proof. We first prove that the set of stable clusters is compatible. Let A be a stable cluster of w and let B be a cluster incompatible with A. Put $U = A - B$, $V = A \cap B$, and $Z = B - A$. Since A and B are incompatible, U, V, and Z are all non-empty. Then $\iota_w^s(A) \leq \overline{w}(UV|Z)$, $\iota_w^s(B) \leq \overline{w}(VZ|U)$, and

$$\begin{aligned}\iota_w^s(A) + \iota_w^s(B) &\leq \overline{w}(UV|Z) + \overline{w}(VZ|U)\\ &= \frac{1}{|U||V||Z|} \sum_{u \in U} \sum_{v \in V} \sum_{z \in Z} w(uv|z) + w(vz|u)\\ &\leq 0\end{aligned}$$

since w is an isolation weighting. The cluster A is stable, so $\iota_w(A) > 0$ and $\iota_w(B) < 0$. Hence B is not a stable cluster and the collection of stable clusters is compatible.

The inclusion of the set of strong clusters and the set of clean clusters in the set of stable clusters (Theorem 2.1) proves compatibility for the other constructions. ∎

The relationship between the three clustering methods can be viewed as a progressive relaxation of the selection criterion. The criteria differ over the conditions on the subsets $R \subseteq r(A)$ that can have zero or negative average weight and the cluster A still be selected: the condition of the strong cluster method is the most strict; the condition of the stable cluster method is the most lenient.

The strong cluster construction is the most conservative. The condition that $\iota_w(A) > 0$ implies that no subset $R \subset r(A)$ can have negative average weight.

At the other extreme, a cluster A can be stable and still have a subset $R \subseteq r(A)$ of size $O(n^3)$ with zero or negative average weight. Such subsets R must be of a particular composition related to the conflicting triples between two incompatible clusters. If A and B are incompatible clusters then $r(A)$ and $r(B)$ conflict on the resolution of several triples, called **conflicting triples**. These are exactly the rooted triples with one element in $A - B$, one element in $A \cap B$, and one element in $B - A$. The stable cluster selection criterion prohibits exactly those negative weight subsets $R \subseteq r(A)$ that correspond to the set of conflicting triples between A and some incompatible cluster B. This guarantees that no two stable clusters will be incompatible.

The clean cluster method falls in between the two other methods: every strong cluster is a clean cluster and every clean cluster is a stable cluster. The clean clustering method bounds the size of subsets $R \subset r(A)$ with negative average weight. The bound $(|A| - 1)$ is equal to the smallest possible set of conflicting triples between A and any other cluster B. We cannot increase this bound without losing the connection with stable clusters and, additionally, the guarantee of compatibility. As we shall see, the bound enables the efficient construction of clean clusters (Section 2.4) whereas the construction of stable clusters is an NP-hard problem (Section 2.5).

2.3. *Properties and Construction of Strong and Apresjan Clusters*

The single linkage tree is one of the oldest hierarchical clustering methods in classification. The method starts with a collection of singleton clusters $\mathscr{A}_1 = \{\{x\} : x \in X\}$. At each iteration $k > 1$ the algorithm chooses a pair of maximal (by inclusion) clusters in $\mathscr{A}_{k-1}$ for which

$\max_{a,b} \{s(a, b) : a \in A,\ b \in B\}$ is largest, and puts $\mathcal{A}_k = \mathcal{A}_{k-1} \cup \{A \cup B\}$. Ties are broken randomly. The procedure finishes when $\mathcal{A}_k$ contains X.

Here we show that Apresjan clusters are always contained in any single linkage tree for s.

THEOREM 2.3. *Let s be a similarity function on X. Then every Apresjan cluster of s is a cluster in every single linkage tree for s.*

Proof. We use a characterization of single linkage trees presented in [4] and used in [12] to solve a related problem. For each $k \in \Re$ we define the threshold graph $G[k]$ with vertex set X and edge set $E[k] = \{\{a, b\} : s(a, b) \geq k\}$.

Let A be an Apresjan cluster of s and suppose that k is the maximum value such that the subgraph induced by A is connected in $G[k]$. This is well defined since $G[-\infty]$ is complete. We will show that no other element of X is in the same component as A. Given any a in A and $x \in X - A$ if $\{a, x\} \in E[k]$ then $s(a, x) \geq k$. Since A is a strong cluster, there is $\epsilon > 0$ such that $s(a, a') \geq \epsilon + s(a, x)$ for all $a' \in A$. Then A is connected in $G[k + \epsilon]$, contradicting the maximality assumption for k. We conclude then that A is a component of $G[k]$. From [4], every component of $G[k]$ is a cluster in every single linkage tree for s. ∎

The link leads to an efficient way to compute the Apresjan clusters:

COROLLARY 2.1. *We can construct the Apresjan clusters of a similarity function s in $O(n^2)$ time, where $n = |X|$.*

Proof. One can construct a single linkage tree T for s in $O(n^2)$ time [15, 23]. By Theorem 2.3, *clus*(T) contains all the Apresjan clusters, and possibly additional clusters. For each node v and element $x \in C(v)$ we compute

$$m[v, x] = \min_{x'} \{s(x, x') : x' \in C(v)\}$$
$$M[v, x] = \max_{x''} \{s(x, x'') : x'' \notin C(v)\}.$$

Let u be the (unique) child of v such that $x \in C(u)$. If $x' \in C(v)$ then either $x' \in C(u)$ or $\text{lca}(x, x') = v$. Hence

$$m[v, x] = \min \left\{ \min_{x'} \{s(x, x') : x' \in C(u)\}, \min_{x'} \{s(x, x') : \text{lca}(x, x') = v\} \right\}$$
$$= \min \left\{ m[u, x],\ \min_{x'} \{s(x, x') : \text{lca}(x, x') = v\} \right\}. \tag{10}$$

The lca relationships can be precomputed in $O(n^2)$ time. A post-order traversal of T using the above conversion then gives the values $m[v, x]$

in $O(n^2)$ total time. Note that a post-order traversal of T processes the children of a node *before* processing the node.

The computation of the values $M[v, x]$ proceeds from the root to the leaves. Let v be a node, let x be an element of $C(v)$, and let u be the unique child of v for which $x \in C(u)$. If $x'' \notin C(u)$ then either $x'' \in C(u)$ or $\mathrm{lca}(x, x'') = v$. Hence

$$\begin{aligned} M[u, x] &= \max\left\{\max_{x''}\{s(x, x'') : x'' \notin C(v)\}, \max_{x''}\{s(x, x'') : \mathrm{lca}(x, x'') = v\}\right\} \\ &= \max\left\{M[v, x], \max_{x''}\{s(x, x'') : \mathrm{lca}(x, x'') = v\}\right\}. \end{aligned} \tag{11}$$

We can therefore use a pre-order traversal of T to compute all of the $M[v, x]$ values in $O(n^2)$ total time. Note that a pre-order traversal processes the children of a node *after* processing the node.

The values $M[u, x]$ and $m[u, x]$ enable the computation of the isolation index of the clusters of T. Let u be a node of T and let $U = C(u)$. Then

$$\begin{aligned} \iota_s(U) &= \min_{x \in U}\left\{\min_{x'}\{s(x, x') : x' \in U\} - \max_{x''}\{s(x, x'') : x'' \notin U\}\right\} \\ &= \min_{x \in U}\{m[u, x] - M[u, x]\}. \end{aligned}$$

The Apresjan clusters are then exactly those clusters of T with positive isolation index. ∎

The strong cluster construction generalises the method of Apresjan by allowing an arbitrary isolation weighting w. We show that this more general construction can be performed using Apresjan clustering method.

LEMMA 2.2. *Let w be an isolation weighting. For each $a, b \in X$ put*

$$s(a, b) = |\{ab|y : w(ab|y) > 0\}|. \tag{12}$$

Then every strong cluster of w is an Apresjan cluster of s.

Proof. Put $R = \{ab|c : w(ab|c) > 0\}$. Let A be a strong cluster for w. Fix $a, a' \in A$ and $x \in X - A$. For all $y \in X - A$ we have $w(aa'|y) \geq \iota_w(A) > 0$ so

$$s(a, a') = |\{y : aa'|y \in R\}| \geq |X - A|.$$

For all $a'' \in A$, $w(aa''|x) \geq \iota_w(A) > 0$ so $w(ax|a'') < 0$ and $ax|a'' \notin R$. Since also $w(ax|x) \leq 0$ we have that $ax|z \in R$ implies $z \notin A \cup \{x\}$. Thus

$$s(a, x) = |\{z : ax|z \in R\}| \leq |X - A - \{x\}|.$$

We therefore have $s(a, x) \leq |X - A| - 1 < s(a, a')$. By the same argument, $s(a', x) < s(a, a')$.

It follows that $s(a, a') > s(a, x)$ for all $a, a' \in A$ and $x \in X - A$. Hence A is an Apresjan cluster of s. ∎

The converse of Lemma 2.2 is not true. For example, if $X = \{a, b, c, d\}$, $w(ab|d) = w(ac|d) = w(bc|a) = 1$ and all other triples have weight -1 then $\{a, b, c\}$ is a strong cluster of s but not of w.

COROLLARY 2.2. *The strong clusters of an isolation weighting can be constructed in $O(n^3)$ time, where $n = |X|$.*

Proof. First construct $s(a, b) = |\{ab|y : w(ab|y) > 0\}|$ for all a, b, taking $O(n^3)$ time, and compute the collection $\mathscr{C}$ of Apresjan clusters of s ($O(n^2)$ time). The isolation indices $\iota_w(A)$ for the clusters in $\mathscr{C}$ can be computed in $O(n^3)$ time using a modification of the quartet path algorithms of [7]. ∎

2.4. *Properties and Construction of Clean Clusters*

We first examine the special case when $w = \rho_s$ for some similarity function s on X (Eq. (4)).

By Theorem 2.1, every Apresjan cluster of s is a clean cluster of s. The converse is not true. Consider the similarity function s on $X = \{a, b, c, d\}$ given in Fig. 1. There are two nontrivial clean clusters, $\{a, b\}$ and $\{a, b, c\}$, but only $\{a, b\}$ is an Apresjan cluster. Furthermore, the unique single linkage tree for s contains nontrivial clusters $\{a, b\}$ and $\{a, b, d\}$, so the clean clusters of s are not necessarily present in a single linkage tree for s. However, as we now show, the clean cluster construction is closely connected with another classical clustering method: the average linkage tree [24] (also known as the group average linkage tree [14]).

THEOREM 2.4. *Every stable cluster of s is a cluster in every average linkage tree for s. Hence every clean cluster of s is a cluster in every average linkage tree for s.*

Proof. For any two disjoint subsets A, B define $\bar{s}(A, B) = \text{av}\{s(a, b) : a \in A,\ b \in B\}$. The average linkage tree algorithm begins with a collection of singleton clusters $\mathscr{A}_1 = \{\{x\} : x \in X\}$. At each iteration $k > 1$ it chooses the pair of maximal (by inclusion) clusters $A, B \in \mathscr{A}_{k-1}$ for which $\bar{s}(A, B)$

s	a	b	c	d
a	5	4	2	3
b		5	2	1
c			5	1
d				5

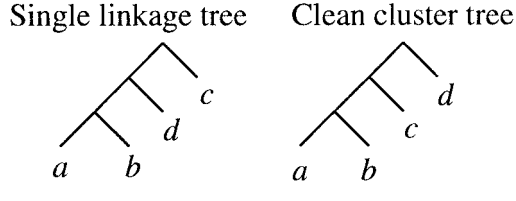

FIG. 1. A similarity function s on four points, its single linkage tree and clean cluster tree.

is largest and puts $\mathscr{A}_k = \mathscr{A}_{k-1} \cup \{A \cup B\}$. Ties can be broken randomly. The procedure terminates when $\mathscr{A}_k$ contains X.

Let $\mathscr{A}$ be the clusters in the average linkage tree for s and let A be a stable cluster that is not a cluster in $\mathscr{A}$. Since $\mathscr{A}$ is a maximal hierarchy, $\mathscr{A} \cup \{A\}$ is incompatible. Let k be the first iteration of the average linkage tree algorithm for which $\mathscr{A}_k$ contains a cluster B that is incompatible with A. There are maximal clusters V and Z in $\mathscr{A}_{k-1}$ such that V and Z are amalgamated to give B. Hence $B = V \uplus Z$ and $\bar{s}(V, A_i) \leq \bar{s}(V, Z)$ for all maximal $A_i \in \mathscr{A}_{k-1}$.

Every cluster in $\mathscr{A}_{k-1}$ is compatible with A, so every maximal cluster in $\mathscr{A}_{k-1}$ is either contained in A or disjoint with A. W.l.o.g. suppose that $V \subset A$ and $Z \cap A = \varnothing$. Let $U_1, U_2, \ldots, U_p$ be the maximal clusters of $\mathscr{A}_{k-1} - \{V\}$ that are also contained in A. Thus $\bar{s}(U_l, V) \leq \bar{s}(V, Z)$ for all $l = 1, 2, \ldots, p$. Let $U = A - V$, so that $U = \bigcup_{l=1}^{p} U_l$ and

$$\begin{aligned} |U|\bar{s}(U, V) &= \sum_{l=1}^{p} |U_i|\bar{s}(U_l, V) \\ &\leq \sum_{l=1}^{p} |U_i|\bar{s}(V, Z) \\ &= |U|\bar{s}(V, Z), \end{aligned}$$

giving $\bar{s}(U, V) \leq \bar{s}(V, Z)$.

For all $u \in U$, $v \in V$, $z \in Z$ we have

$$\rho_s(uv|z) = s(u, v) - \max\{s(u, z), s(v, z)\} \leq s(u, v) - s(v, z).$$

Let $w = \rho_s$. Then

$$\begin{aligned} \iota_w^s(A) &\leq \overline{w}(UV|Z) \\ &= \frac{1}{|U||V||W|} \sum_{u \in U} \sum_{v \in V} \sum_{z \in Z} \rho_s(uv|z) \\ &\leq \frac{1}{|U||V||W|} \sum_{u \in U} \sum_{v \in V} \sum_{z \in Z} s(u, v) - s(v, z) \\ &= \bar{s}(U, V) - \bar{s}(V, Z) \\ &\leq 0, \end{aligned}$$

a contradiction. Hence A is contained in the average linkage tree.

By Theorem 2.1 all clean clusters are stable clusters and so are contained in every average linkage tree. ∎

Apart from providing a characterisation of a subset of clusters in the average linkage tree, Theorem 2.4 leads to an $O(n^3)$ time algorithm for constructing clean clusters of a similarity function (cf. Algorithms 1 and 2).

COROLLARY 2.3. *The clean clusters of a similarity function s can be constructed in $O(n^3)$ time, where $n = |X|$.*

Proof. First compute an average linkage tree T for s using the $O(n^2)$ time algorithm of [21]. By Theorem 2.4, every clean cluster of s is a cluster of T.

For each pair of nodes u, v in T such that $u \prec_T v$, define

$$R[u, v] = \{xy|z : \text{lca}(x, y) = u,\ \text{lca}(x, z) = \text{lca}(y, z) = v\} \tag{13}$$

and

$$R_{\preceq}[u, v] = \bigcup_{u' \preceq_T u} R[u', v]. \tag{14}$$

Every rooted triple in $r(T)$ is in exactly one set $R[u, v]$, and these sets can be constructed in $O(n^3)$ time.

Let $R[u, v, n]$ denote the set of n minimum weight rooted triples of $R[u, v]$. It takes $O(|R[u, v]|)$ time to extract $R[u, v, n]$ from $R[u, v]$ using the linear time selection algorithm of [8]. It therefore takes $O(n^3)$ time to extract all of the sets $R[u, v, n]$.

Let $R_{\preceq}[u, v, n]$ denote the minimum n triples of $R_{\preceq}[u, v]$. We construct these sets using dynamic programming. Let $u_1, u_2, \ldots, u_m$ be the children of u. Then $R_{\preceq}[u, v, n]$ is equal to the set of n minimum weight triples in

$$R[u, v, n] \cup R_{\preceq}[u_1, v, n] \cup R_{\preceq}[u_2, v, n] \cup \cdots \cup R_{\preceq}[u_{m(u)}, v, n]. \tag{15}$$

For each v, it takes $O(n^2)$ to compute all of these sets $R[u, v, n]$ using a post-order traversal. There are $O(n)$ choices for v so extracting all the sets $R_{\preceq}[u, v, n]$ takes $O(n^3)$ time.

Let $U = C(u)$. Then $r(U) = \bigcup_{v \succ_T u} R_{\preceq}[u, v]$ and so the minimum $|U| - 1$ triples in $r(U)$ are the minimum $|U| - 1$ triples in $\bigcup_{v \succ_T u} R_{\preceq}[u, v, n]$. Since there are at most n^2 triples in $|\bigcup_{v \succ_T u} R_{\preceq}[u, v, n]|$ we can compute

$$\iota_w^c(U) = \text{avmin}(r(U), |U| - 1)$$

in $O(n^2)$ time. It therefore takes $O(n^3)$ time to compute the $O(n)$ isolation indices. ∎

We now consider the generalisation from clean clusters of ρ_s to clean clusters of an arbitrary isolation weighting w. There is apparently no analogue of Lemma 2.2 for C-clusters. We have constructed an example for five elements where a clean cluster of w is not a clean cluster for the similarity function $s(a, b) = |\{ab|y : w(ab|y) > 0\}|$ of Eq. (12). We were also unable to find a variation on s for which a version of Lemma 2.2 might hold.

We follow an alternative approach. Given an isolation weighting w and $r \in X$, put

$$CH(w, r) = \{A \subseteq X : w(ar|x) > 0 \text{ for all } a \in A,\ x \in X - A\}. \tag{16}$$

LEMMA 2.3. *The set $CH(w, r)$ forms a chain; that is, for each $A, B \in CH(w, r)$ either $A \subseteq B$ or $B \subseteq A$.*

Proof. From the definition of an isolation weighting we deduce $w(xr|r) \leq 0$ for all $x \in X$ and so if $A \in CH(w, r)$ and $A \neq \emptyset$ then $r \in A$. Given any two clusters $A, B \in CH(w, r)$, if there is a, b such that $a \in A - B$ and $b \in B - A$ then we obtain $w(ar|b) > 0$ and $w(br|a) > 0$, a contradiction. ∎

LEMMA 2.4. *We can construct a chain containing $CH(w, r)$ in $O(n^2)$ time.*

Proof. Let G be the semi-complete digraph with vertex set X and edge set

$$E = \{(u, v) : w(ur|v) \leq 0\}.$$

Use the $O(n^2)$ algorithm of [16] to determine a (directed) Hamiltonian path $a_1, a_2, \ldots, a_n$ for G. For each $A \in CH(w, r)$ there is no edge (u, v) in E such that $u \in A$ and $v \notin A$. Hence there must be i such that $A = \{a_i, a_{i+1}, \ldots, a_n\}$. The collection of sets $\{a_i, a_{i+1}, \ldots, a_n\}$ for $i = 1, \ldots, n$ is a chain containing all of the clusters in $CH(w, r)$. ∎

Our method for constructing the clean clusters of an isolation weighting is iterative. We first restrict the isolation weighting to three elements and then incorporate one element at a time into the analysis. The basis for the iterative algorithm, and the connection between chains and clean clusters, is provided by:

THEOREM 2.5. *Let w be an isolation weighting for X. If A is a clean cluster of w and $r \in X$ then either $A \in CH(w, r)$ or $A - \{r\}$ is a clean cluster of w restricted to $X - \{r\}$, or both.*

Proof. Suppose that A is not a clean cluster of w restricted to $X - \{r\}$. Then there is R such that

$$R \subseteq \{aa'|x : a, a' \in A - \{r\},\ x \in X - \{r\}\},$$

$|R| = |A - \{r\}| - 1$, and $\sum_{aa'|x \in R} w(aa'|x) \leq 0$. If A is also not a cluster in $CH(w, r)$ then there is $a \in A$ and $x \in X - A$ such that $w(ar|x) < 0$. Putting $R' = R \cup \{ar|x\}$ we obtain a subset of $r(A)$ containing at least $|A| - 1$ elements such that $\sum_{aa'|x \in R'} w(aa'|x) \leq 0$. Hence A is not a clean cluster of w. ∎

Theorem 2.5 leads directly to a polynomial time algorithm for constructing clean clusters of an isolation weighting. Pseudocode is presented in the Appendix (Algorithm 3).

COROLLARY 2.4. *The clean clusters of an isolation weighting can be computed in $O(n^4)$ time.*

Proof. We use an iterative algorithm. Order X arbitrarily as $X = \{x_1, x_2, \ldots, x_n\}$, putting $X_k = \{x_1, x_2, \ldots, x_k\}$ for each $k : 1 \leq k \leq n$. Let $w_{|X_k}$ denote the restriction of w to X_k and let $\mathscr{C}_k$ denote the clean clusters for $w_{|X_k}$. We compute $\mathscr{C}_2$ directly.

By Theorem 2.5,

$$\mathscr{C}_{k+1} \subseteq CH(w_{|X_{k+1}}, x_{k+1}) \cup \{A \subseteq X_{k+1} : A - \{x_{k+1}\} \in \mathscr{C}_k\}.$$

We construct $CH(w_{|X_{k+1}}, x_{k+1})$ using Lemma 2.4. The clean cluster indices of the clusters $A \in CH(w_{|X_{k+1}}, x_{k+1})$ can be computed in $O(n^3)$ time using the same technique as in the proof of Corollary 2.3. The clean cluster indices of the clusters in $\{A : A - \{x_{k+1}\} \in \mathscr{C}_k\}$ can be computed by maintaining a list $\hat{r}[A, k]$ of n minimum weight triples for each cluster A. After each element addition,

$$\hat{r}[A, k+1] \subseteq \hat{r}[A, k] \cup \{aa'|x_{k+1} : a, a' \in A\} \tag{17}$$

$$\hat{r}[A \cup \{x_{k+1}\}, k+1] \subseteq \hat{r}[A, k] \cup \{ax_{k+1}|y : a \in A, y \in X_k - A\}. \tag{18}$$

Hence we can construct $\hat{r}[A, k+1]$ and $\hat{r}[A \cup \{x_{k+1}\}, k+1]$ in $O(n^2)$ time per cluster A. Each iteration step from $\mathscr{C}_k$ to $\mathscr{C}_{k+1}$ can be completed in $O(n^3)$ time, giving $O(n^4)$ time for the entire construction. ∎

2.5. *Construction of Stable Clusters*

The stable cluster method is the last, and most refined, method in the series. Let s be a similarity function. By Theorem 2.4 we know that any average linkage tree T for s contains all of the stable clusters of s. All that remains is to examine the $O(n)$ clusters in $clus(T)$ and discard those clusters that are not stable. This proves to be a significantly more difficult problem.

THEOREM 2.6. *It is an NP-hard problem to construct the set of stable clusters of a similarity function. Hence it is also NP-hard to construct the stable clusters of an arbitrary isolation weighting.*

Proof. We provide a transformation from:

SPARSEST CUT

Instance: Vertex set V, cost $c(u, v)$ for every unordered pair $\{u, v\} \in V \times V$. Rational k.

Question: Is there $U \subsetneq V$ such that

$$\sum_{u \in U, v \in V - U} c(u, v) < k|U||V - U|?$$

SPARSEST CUT was shown to be NP-hard in [19].

Let V, c, and k make up an arbitrary instance of SPARSEST CUT. Put $X = V \cup \{z\}$, where z is a new element. Define a similarity function s on X by $s(u, v) = c(u, v)$ for all $u, v \in V$ and $s(z, v) = k$ for all $v \in V$. For all $u, v \in V$ we have $\rho_s(uv|z) = c(u, v) - k$. We claim that $\iota_s^s(V) > 0$ if and only if there is no $U \subset V$ such that $\sum_{u \in U, v \in V-U} c(u, v) \leq k|U||V - U|$.

All clusters incompatible with V are of the form $U \cup \{z\}$ for some $U \subset V$. For such a cluster, we have

$$\begin{aligned}\overline{w}(U(V-U)|\{z\}) &= \frac{1}{|U||V-U|} \sum_{u \in U, v \in V-U} \rho_s(uv|z) \\ &= -k + \sum_{u \in U, v \in V-U} \frac{c(u, v)}{|U||V-U|}.\end{aligned}$$

Hence $\iota_s^s(V) > 0$ if and only if there is no U such that

$$\sum_{u \in U, v \in V-U} c(u, v) \leq k|U||V - U|.$$

The result follows. ∎

3. A FAMILY OF SPLIT METHODS

In this section we present a series of methods for classification using splits. The series is analogous to the clustering constructions investigated in the previous section. Indeed, the duality between the cluster and split cases fits nicely into the framework of [11]: the splits are to clusters what projective space is to affine space.

Our presentation of the split and unrooted tree constructions follows the same lines as the presentation of the clustering methods.

3.1. *Definitions*

Unrooted X-trees and splits were defined in Section 1. Put $n = |X|$. Let *splits*(T) denote the set of splits of a tree T. A set of splits $\mathcal{S}$ of X is **compatible** if there is a tree T such that $\mathcal{S} \subseteq$ *splits*(T). Buneman showed that $\mathcal{S}$ is compatible if and only if for each pair $A|B$, $C|D$ of splits in $\mathcal{S}$ at least one of $A \cap C$, $A \cap D$, $B \cap C$, $B \cap D$ is empty [10].

A **quartet** $ab|cd$ defines a separation of a and b from c and d. The set of all possible quartets on X is denoted $\mathcal{Q}(X)$. The set of quartets of a split $A|B$ is defined by

$$q(A|B) = \{aa'|bb' : a, a' \in A,\ b, b' \in B\}.$$

We say that $ab|cd$ is a quartet of T if there is a split $A|B \in$ *splits*(T) such that $ab|cd \in q(A|B)$. The set of quartets of T is denoted $q(T)$.

A **dissimilarity function** is a symmetric function from $X \times X$ to $\Re^+$. A **separation weighting** is a function ω: $\mathcal{Q}(X) \to \Re$ such that $\omega(ab|cd) + \omega(ac|bd) \leq 0$ for all $a, b, c, d \in X$. The weight $\omega(ab|cd)$ indicates the degree to which a and b are separated from c and d. Given a dissimilarity function δ, define

$$\beta_\delta(ab|cd) = \frac{1}{2}(\min\{\delta(a, c) + \delta(b, d), \delta(a, d) + \delta(b, c)\} - \delta(a, b) - \delta(c, d)). \tag{19}$$

Then β_δ is a separation weighting.

Given a set Q of quartets, a separation weighting ω, and a positive integer $k \leq |Q|$, define

$$\text{av}(Q) = \frac{1}{|Q|} \sum_{ab|cd \in Q} \omega(ab|cd) \tag{20}$$

$$\text{avmin}(Q, k) = \min_{Q'}\{av(Q') : Q' \subseteq Q,\ |Q'| = k\}. \tag{21}$$

Given non-empty disjoint subsets U, V, Y, Z define

$$\bar{\omega}(UV|YZ) = \text{av}\{uv|yz : u \in U, v \in V, y \in Y, z \in Z\}, \tag{22}$$

the average weight of all quartets $uv|yz$ with $u \in U$, $v \in V$, $y \in Y$, and $z \in Z$.

3.2. *A Family of Tree Construction Methods*

In Section 2.2 we defined a collection of isolation indices and showed that, for each index, the clusters with positive index formed a hierarchy. In this section we do the same for splits: we define a collection of separation indices and then show that the resulting collections of splits are compatible.

DEFINITION OF SPLIT INDICES. Let ω be a separation index and let $A|B$ be a split of X.

(i) The **strong separation index** of $A|B$ is defined

$$\mu_\omega(A|B) = \min_{uv|yz}\{\omega(uv|yz) : uv|yz \in q(A|B)\}$$

and the **Buneman splits** for ω are $\{A|B : \mu_\omega(A|B) > 0\}$.

(ii) The **refined Buneman index** of $A|B$ is defined

$$\mu^r_\omega(A|B) = \text{avmin}(q(A|B), n - 3)$$

and the **refined Buneman splits** for ω are $\{A|B : \mu^r_\omega(A|B) > 0\}$.

(iii) The **clean split index** of $A|B$ is defined

$$\mu^c_\omega(A|B) = \text{avmin}(q(A|B), (|A|-1)(|B|-1))$$

and the **clean splits** for ω are $\{A|B : \mu^c_\omega(A|B) > 0\}$.

(iv) The **split stability index** of $A|B$ is defined

$$\mu^s_\omega(A|B) = \min_{U,V,Y,Z\neq\emptyset}\{\bar{\omega}(UV|YZ) : A = U \uplus V, B = Y \uplus Z\}$$

and the **stable splits** of ω are $\{A|B : \mu^s_\omega(A|B) > 0\}$.

The Buneman splits of a dissimilarity function δ are the Buneman splits of $\omega = \beta_\delta$, similarly for refined Buneman splits, clean splits, and stable splits.

The refinement relations between the different methods are presented in Theorem 3.1. Theorem 3.2 establishes that each construction gives compatible splits. Both theorems use the following bounds on the size of the conflict set for incompatible splits. The indices, and the construction complexities, for all of the split methods are summarised in Table II.

LEMMA 3.1. *If U, V, Y, Z are non-empty subsets of X such that $A = U \uplus V$ and $B = Y \uplus Z$ then*

$$|U||V||Y||Z| \geq (|A|-1)(|B|-1) \geq n-3.$$

Proof. Put $p = |V|$ and $q = |Z|$. Then

$$|U||V||Y||Z| = (|A|-p)p(|B|-q)q. \tag{23}$$

Using straightforward calculus we can show that $(|A|-p)p \geq (|A|-1)$ and $(|B|-q)q \geq (|B|-1)$, giving the first inequality. For the second, put $m = (|B|-1)$ so that $|A|-1 = n-2-m$ and $(|A|-1)(|B|-1) = (n-2-m)m$. The minimum $n-3$ is obtained when $m = 1$. ∎

TABLE II
A Summary of the Split Indices and Construction Complexities for the Family of Split Methods

Construction	Index	Complexity
Buneman splits	$\mu_\omega(A\|B) = \min_{uv\|yz}\{\omega(uv\|yz) : uv\|yz \in q(A\|B)\}$	$O(n^3)$ (dis.[a])
		$O(n^4)$ (sep. wt.[b])
Ref. Buneman splits	$\text{avmin}(q(A\|B), n-3)$	$O(n^5)$
Clean splits	$\mu^c_\omega(A\|B) = \text{avmin}(q(A\|B), (\|A\|-1)(\|B\|-1))$	$O(n^5)$
Stable splits	$\mu^s_\omega(A\|B) = \min_{U,V,Y,Z}\{\bar{\omega}(UV\|YZ) : A = U \uplus V, B = Y \uplus Z\}$	NP-hard

[a] Complexity when input is a dissimilarity function.
[b] Complexity when input is a separation weighting.

THEOREM 3.1. *Let ω be a separation weighting on X. For any split $A|B$ we have*

$$\mu_\omega(A|B) \leq \mu^r_\omega(A|B) \leq \mu^c_\omega(A|B) \leq \mu^s_\omega(A|B).$$

Hence the set of Buneman splits is contained within the set of refined Buneman splits which is contained within the set of clean splits which is contained within the set of stable splits.

Proof. The inequalities $\mu_\omega(A|B) \leq \mu^r_\omega(A|B) \leq \mu^c_\omega(A|B)$ follow from the observation that $k \leq k'$ implies $\text{avmin}(q(A|B), k) \leq \text{avmin}(q(A|B), k')$. Choose non-empty and disjoint subsets U, V, Y, Z such that $A = U \uplus V$, $B = Y \uplus Z$, and $\mu^s_w(A|B) = \bar{\omega}(UV|YZ)$. Then

$$\mu^c_\omega(A|B) = \text{avmin}(q(A|B), (|A| - 1)(|B| - 1)) \tag{24}$$

$$\leq \text{avmin}(q(A|B), |U||V||Y||Z|) \tag{25}$$

$$\leq \bar{\omega}(UV|YZ) \tag{26}$$

$$= \mu^s_w(A|B). \tag{27}$$

∎

THEOREM 3.2. *Let ω be a separation weighting. The sets of Buneman splits, refined Buneman splits, clean splits, and stable splits of ω are all compatible collections of splits.*

Proof. We prove that the set of stable splits is compatible. The other compatibility results follow from Theorem 3.1.

Let $A|B$ be a stable split and let $C|D$ be a split incompatible with $A|B$. Put $U = A \cap C$, $V = A \cap D$, $Y = B \cap C$, $Z = B \cap D$. Since $A|B$ and $C|D$ are incompatible, U, V, Y, and Z are non-empty. Then

$$\begin{aligned}\mu^s_\omega(A|B) + \mu^s_\omega(C|D) &\leq \bar{\omega}(UV|YZ) + \bar{\omega}(UY|VZ) \\ &= \frac{1}{|U||V||Y||Z|} \sum_{u \in U} \sum_{vV} \sum_{y \in Y} \sum_{z \in Z} (\omega(uv|yz) + \omega(uy|vz)) \\ &\leq 0\end{aligned}$$

and so $\mu^s_\omega(A|B) > 0$ implies $\mu^s_\omega(C|D) < 0$. It follows that the set of stable splits is compatible. ∎

The relationship between the tree construction methods parallels that for the clustering methods. The Buneman split method is quite conservative. For $A|B$ to be a Buneman split of ω we must have $\omega(aa'|bb') > 0$ for all $aa'|bb' \in q(A|B)$. The refined Buneman index allows $n - 3$ quartets in $q(A|B)$ to have negative average weight. The clean split method incorporates the size of A and B into the calculation of the index. The bound of

$(|A| - 1)(|B| - 1)$ stems from Lemma 3.1. If we increase this bound then clean splits will no longer necessarily be stable splits and, additionally, we lose the guarantee of compatibility for the splits selected.

The stable split method allows the largest number of negative weight quartets. A split $A|B$ can have $O(n^4)$ negative weight quartets in $q(A|B)$ and yet still be stable. There will always, however, be more positive weight quartets in $q(A|B)$ than negative ones. The stability index of a split can be viewed as the measure of support for a split $A|B$ on exactly those quartets with which $A|B$ is in conflict with an incompatible split $C|D$.

3.3. *Properties and Construction of Buneman Splits*

We now formalise the connection between the Buneman split method and Apresjan clusters. Let δ be a dissimilarity function on X. The *Farris transform* of δ with respect to $x \in X$ is the similarity function s_x on $X - \{x\}$ with

$$s_x(a, b) = \frac{1}{2}(\delta(a, x) + \delta(b, x) - \delta(a, b)) \tag{28}$$

for all $a, b \in X - \{x\}$ [13]. The inverse of the Farris transform is given by

$$\delta(a, b) = s_x(a, a) + s_x(b, b) - 2s_x(a, b)$$

for all $a, b \in X - \{x\}$, and $\delta(a, x) = s_x(a, a)$ as well as $\delta(x, x) = 0$.

The Farris transform links Buneman splits and Apresjan clusters. This connection leads to the suprising algorithmic result that Buneman splits of a dissimilarity function can be constructed in $O(n^3)$ time, even though the definition appears to imply that $O(n^4)$ quartets need to be considered.

THEOREM 3.3. (i) *Let δ be a dissimilarity function on X and let s_x denote the Farris transform of δ with respect to $x \in X$. For any split $A|B$ of X,*

$$\mu_\delta(A|B) = \min_b\{\iota_{s_b}(A) : b \in B\} = \min_a\{\iota_{s_a}(B) : a \in A\}. \tag{29}$$

Thus $A|B$ is a Buneman split for δ if and only if A is an Apresjan cluster of s_b for all $b \in B$, and this holds if and only if B is an Apresjan cluster of s_a for all $a \in A$. (ii) *The Buneman splits of a dissimilarity function δ can be computed in $O(n^3)$ time.*

Proof. Equation (29) follows immediately from the observation that $\beta_\delta(ab|cx) = \rho_{s_x}(ab|c)$.

We now prove (ii). For each $x \in X$ we compute the Farris transform s_x of δ with respect to x ($O(n^2)$ time for each x) and then construct the Apresjan clusters of s_x in $O(n^2)$ time (Corollary 2.1). Let S_x be the set of splits $A|(X - A)$ such that A is a strong cluster of s_x. The Buneman splits are exactly those splits appearing in all of the sets of splits S_x, and the corresponding separation indices are given by Eq. (29). Hence the entire computation takes $O(n^3)$ time. ∎

The Farris transform can be extended to a general separation weighting ω. For each $x \in X$, define an isolation weighting w_x on rooted triples of $X - \{x\}$ by $w_x(ab|c) = \omega(ab|cx)$ for all $a, b, c \in X - \{x\}$.

PROPOSITION 3.1. *For all splits $A|B$,*

$$\mu_\omega(A|B) = \min_b\{\iota_{w_b}(A) : b \in B\} = \min_a\{\iota_{\omega_a}(B) : a \in A\}.$$

Hence $A|B$ is a Buneman split for ω if and only if A is a strong cluster of w_b for all $b \in B$, if and only if B is a strong cluster of w_a for all $a \in A$.

Proof. Follows directly from the definition $w_x(ab|c) = \omega(ab|cx)$. ∎

Proposition 3.1 leads directly to a new $O(n^4)$ time algorithm for constructing the Buneman splits of an separation weighting. The first $O(n^4)$ time algorithm for this problem was given by [7].

3.4. *Properties and Construction of Refined Buneman Splits*

Theorem 3.3 and Lemma 3.1 link the Buneman method for constructing unrooted trees with the strong cluster method for constructing rooted trees. There appears to be no analogous relationship between the refined Buneman tree and a clustering method. One can easily construct a dissimilarity matrix δ on five points for which there exists a refined Buneman split $A|B$ such that A is not a stable cluster of s_a for some $a \in A$. Hence A would also not be a C-cluster or strong cluster for s_a.

We can, however, exploit another kind of connection between refined Buneman splits and the clustering methods. Theorem 3.4, similar in structure to Theorem 2.5, was used by [9] to show that the refined Buneman tree can be constructed in polynomial time. For an arbitrary $r \in X$, let w_r denote the isolation weighting on $X - \{r\}$ given by $w_r(ab|c) = \omega(ab|cr)$ (cf. Section 3.3). The basis of the iterative algorithm is

THEOREM 3.4 [9]. *Let ω be an separation weighting for X. If $A|B$ is a refined Buneman split of ω and $r \in B$ then either A is a strong cluster of w_r or $A|(B - \{r\})$ is a refined Buneman split of ω restricted to $X - \{r\}$.*

The algorithm of [9] runs in $O(n^6)$ time. Here we improve this complexity to $O(n^5)$ time, first showing how refined Buneman indices can be quickly computed on a tree. Pseudocode for the computation of refined Buneman indices can be found in the Appendix (Algorithm 4).

LEMMA 3.2. *Let T be an X-tree. We can compute $\mu^r_\omega(A|B)$ for each split $A|B \in splits(T)$ in $O(n^4)$ time.*

Proof. We adapt the algorithm presented in the proof of Corollary 2.3 to quartets. We first root T at an arbitrary node. For each pair of vertices u and v in T let $Q[u, v]$ be the set of quartets $ab|cd$ such that the path from a to c intersects the path from b to d exactly along the path from u to v. Then every quartet $ab|cd$ induced by the previously unrooted tree corresponding to T appears in exactly two sets $Q[u, v]$ (noting $Q[u, v] = Q[v, u]$), and we can construct these sets in $O(n^4)$ time.

Let $Q[u, v, n-3]$ denote the $n-3$ minimum weight quartets in $Q[u, v]$. Using the algorithm of [8] we can compute $Q[u, v, n-3]$ from $Q[u, v]$ in $O(|Q[u, v]|)$ time. Hence we can compute all of the sets $Q[u, v, n-3]$ in $O(n^4)$ time.

For each pair (u, v) such that $v \not\preceq u$ we define

$$Q_{\preceq}[u, v] = \bigcup_{u' \preceq_T u} Q[u', v]$$

and let $Q_{\preceq}[u, v, n-3]$ denote the set of $n-3$ minimum weight quartets in $Q_{\preceq}[u, v]$. We can compute the sets $Q_{\preceq}[u, v, n-3]$ using one post-order traversal for each choice of v: Let $u_1, u_2, \ldots, u_m$ be the children of u. Then $Q_{\preceq}[u, v, n-3]$ equals the set of $n-3$ minimum weight quartets in

$$Q[u, v, n-3] \cup Q_{\preceq}[u_1, v, n-3] \cup \cdots \cup Q_{\preceq}[u_m, v, n-3]. \tag{30}$$

For each v it takes $O(n^2)$ time to compute all of these sets $Q_{\preceq}[u, v, n-3]$ using a post-order traversal. There are $O(n)$ choices for v so extracting all of the sets $Q_{\preceq}[u, v, n-3]$ takes $O(n^3)$ time.

Given a vertex u, let U be the set of leaves that are descendents of u and put $V = X - U$. Then $q(U|V) = \bigcup_{v: v \not\preceq_T u} Q_{\preceq}[u, v]$ and so the $n-3$ minimum weight quartets in $q(U|V)$ are the $n-3$ minimum weight quartets in

$$Q_u = \bigcup_{v \not\preceq_T u} Q_{\preceq}[u, v, n-3].$$

The refined Buneman index $\mu^r_w(U|V)$ can therefore be computed in $O(n^2)$ time. The entire computation takes $O(n^4)$ time. ∎

COROLLARY 3.1. *The refined Buneman splits of a separation weighting ω can be constructed in $O(n^5)$ time.*

Proof. We use an iterative algorithm, whose pseudocode is given in the Appendix (Algorithm 5). Order X arbitrarily as $X = \{x_1, x_2, \ldots, x_n\}$, putting $X_k = \{x_1, x_2, \ldots, x_k\}$ for each $k : 1 \le k \le n$. Let $\omega_{|X_k}$ denote the restriction of ω to X_k and let $\mathscr{R}_k$ denote the refined Buneman splits for $\omega_{|X_k}$. We compute $\mathscr{R}_2$ directly.

Put $r = x_{k+1}$. Let $\mathscr{C}_{k+1}$ denote the set of strong clusters of the isolation weighting w_r on X_k given by $w_r(ab|c) = \omega(ab|cr)$. By Theorem 3.4,

$$\mathscr{R}_{k+1} \subseteq \{(A \cup \{r\})|B : A|B \in \mathscr{R}_k\} \cup \{A|(X_{k+1} - A) : A \in \mathscr{C}_{k+1}\}.$$

All refined Buneman indices of splits in $\{(A \cup \{r\})|B : A|B \in \mathscr{R}_k\}$ can be computed in $O(n^4)$ time in total by maintaining a list $q_{\min}[A|B, k]$ of $n-3$ minimum quartets for each split $A|B \in \mathscr{R}_k$. After each element addition

$$q_{\min}[A \cup \{x_{k+1}\}|B, k+1] \subseteq q_{\min}[A|B, k] \cup \{ax_{k+1}|bb' : a \in A, b, b' \in B\} \qquad (31)$$

and so $q_{\min}[A \cup \{x_{k+1}\}|B, k+1]$ can be obtained in $O(n^3)$ time, similarly for $q_{\min}[A|B \cup \{x_{k+1}\}, k+1]$. The refined Buneman indices of splits in $\{A|(X_{k+1} - A) : A \in \mathscr{C}_{k+1}\}$ can be computed in $O(n^4)$ time using Lemma 3.2. Hence we can compute $\mathscr{R}_{k+1}$ from $\mathscr{R}_k$ in $O(n^4)$ time and the entire construction takes $O(n^5)$ time. ∎

3.5. *Properties and Construction of Clean Splits*

There is apparently no analogue of Theorem 3.3 for clean splits. We have constructed an example on eight elements where δ has a clean split $A|B$ such that A is not in the average linkage tree for s_b for *any* $b \in B$.

In Section 2.4 we showed how clean clusters can be constructed iteratively using a connection with rooted chains (Theorem 2.5). Here we show that an similar connection applies to the clean split case. Let w_r denote the isolation weighting on $X - \{r\}$ given by $w_r(ab|c) = \omega(ab|cr)$ (cf. Section 3.3).

THEOREM 3.5. *Let ω be a separation weighting for X. If $A|B$ is a clean split of ω and $r \in B$ then either A is a clean cluster of w_r or $A|(B - \{r\})$ is a clean split of ω restricted to $X - \{r\}$.*

Proof. Suppose that $A|(B - r)$ is not a clean split of ω restricted to $X - \{r\}$. Then there is $Q \subseteq q(A|B - \{r\})$ such that $|Q| = (|A| - 1)(|B - \{r\}| - 1)$ and $\sum_{aa'|bb' \in Q} \omega(aa'|bb') \leq 0$. If A is not a clean cluster of w_r then there is R such that

$$R \subseteq \{aa'|x : a, a' \in A,\ x \in X - \{r\}\},$$

$|R| = |A| - 1$, and $\sum_{aa'|x \in R} w_r(aa'|x) \leq 0$. Put $Q_R = \{aa'|xr : aa'|x \in R\}$, so

$$\sum_{aa'|xr \in Q_R} \omega(aa'|xr) \leq 0.$$

Put $Q' = Q \cup Q_R$. Then Q' is a subset of $q(A|B)$ containing $(|A| - 1)(|B| - 2) + (|A| - 1) = (|A| - 1)(|B| - 1)$ quartets such that $\sum_{ab|cd \in Q'} \omega(ab|cd) \leq 0$. Hence $A|B$ is not a clean split of ω. ∎

Theorem 3.5 provides the reduction step of a polynomial time iterative algorithm for constructing the set of clean splits.

THEOREM 3.6. *The clean splits of a separation weighting ω can be constructed in $O(n^5)$ time.*

Proof. First note that we can compute $\mu^c_\omega(A|B)$ for each split $A|B \in splits(T)$ in $O(n^4)$ time by using a variant of Lemma 3.2. The $O(n^5)$ time algorithm is almost identical to that for refined Buneman splits described in Corollary 3.1 (cf. Algorithms 4 and 5). ∎

The quartet cleaning method of [6] is a special case of the unrooted C-tree construction. Let Q be a set of quartets such that for each $a, b, c, d \in X$ at most one of $ab|cd$, $ac|bd$, $ad|bc$ is in Q. Define a separation weighting ω by putting $\omega(ab|cd) = 1$ if $ab|cd \in Q$ and $\omega(ab|cd) = -1$ if $ab|cd \notin Q$. Then $A|B$ is a clean split of ω if and only if $|q(A|B) - Q| < (|A| - 1)(|B| - 1)/2$, the condition for $A|B$ to be a "clean" split with respect to the terminology of [6].

3.6. *Construction of Stable Splits*

Like the stable clusters, the stable splits are the last, and most refined, in the series of constructions. As with stable clusters, the stable splits are difficult to construct:

THEOREM 3.7. *It is an NP-hard problem to construct the set of stable splits of a dissimilarity function. Hence it is also NP-hard to construct the stable clusters of an arbitrary separation weighting.*

Proof. Let V, c, and k make up an arbitrary instance of SPARSEST CUT. We construct a dissimilarity function δ such that $\beta_\delta(uv|xy) = c(u, v) - k$ for all $u, v \in V$. The result then follows by the same arguments used to prove Theorem 2.6. ∎

4. DISCUSSION

We have presented two parallel series of tree based classification methods, one giving clusters and rooted trees, the other giving splits and unrooted trees. The progression began with two classical constructions, Apresjan clusters and Buneman splits, and finished with two general (but NP-hard) constructions, stable clusters, and stable splits. We explored links between the different constructions and with well-known classification methods.

We see two major directions for future work. The first is to investigate the performance of these methods in an applied field of classification such

as phylogenetics. Three variants of the methods described here [6, 7, 17] have already been employed by computational biologists. The appeal of combinatorial approaches in phylogenetics is that quartet weights can be determined using complex phylogenetic criteria and models that are computationally infeasible for larger sets of taxa. Quartet weights are then used to construct a tree in a reasonable amount of time. The clean split construction is particularly well suited for incorporation in a combinatorial strategy for phylogenetic reconstruction. The method, like others presented here, is continuous, is clearly defined, has a polynomial time algorithm, and can be used to analyse both dissimilarity data and quartet weights.

A second direction of future work is further generalisation. Bandelt and Dress [3] established a duality between weak hierarchies and weakly compatible splits [3]. We have already used this to provide a more efficient algorithm for split decomposition [5]. A series of general clustering methods, extending work on weak hierarchies, was explored in [11]. The question arises of whether these two methods can be refined like the strong clusters and Buneman tree, and whether the connections with single linkage/average linkage can be generalised to agglomerative methods for non-hierarchical clustering.

APPENDIX: ALGORITHMS

In the following algorithms we let $\mathscr{L}(u)$ denote the labels of the leaves that are descendents of the node u.

ALGORITHM 1 (*Compute* $\iota_w^c(U)$ *for all* $U \in clus(T)$).

Compute and tabulate $\mathrm{lca}(x, x')$ for all leaves x, x' in T.
Iterate through all rooted triples $xy|z \in r(T)$ and use lca tables to construct $R[u, v]$ for all $u, v \in V(T)$ such that $u \prec_T v$.
For each pair $u, v \in V(T)$ such that $u \prec_T v$ use the algorithm of [8] to determine the set $R[u, v, n]$ of n minimum weight triples in $R[u, v]$.
For all vertices $v \in V(T)$ do
  For all vertices u in a post-order traversal of T do
    Use Eq. (15) and [8] to construct $R_{\preceq}[u, v, n]$.
For all vertices $u \in V(T)$ compute
  $R_u \leftarrow \bigcup_{v \succ_T u} R_{\preceq}[u, v, n]$
  $\iota_w^c(\mathscr{L}(u)) \leftarrow \mathrm{av\ min}(\{w(xy|z) : xy|z \in R_u\}, |\mathscr{L}(u) - 1|)$, using [8].
Output cluster indices $\iota_w^c(\mathscr{L}(u))$ for all $u \in V(T)$.

ALGORITHM 2 (*Rooted C-tree for a similarity function s on X*).

Compute an average linkage tree T for s.
Use Algorithm 1 with $w = \beta_s$ to compute $\iota_w^c(U)$ for all $U \in clus(T)$.
Output $RC(\beta_s) = \{U \in clus(T) : \iota_w(U) > 0\}$.

ALGORITHM 3 (*Rooted C-tree for an isolation weighting w*).

Order X arbitrarily as $x_1, x_2, \ldots, x_n$
$\mathscr{C}_2 \leftarrow \{\{x_1\}, \{x_2\}\}$
$\hat{r}[\{x_1\}, 2] \leftarrow \{x_1x_1|x_2\}$; $\hat{r}[\{x_2\}, 2] \leftarrow \{x_2x_2|x_1\}$.
For k from 2 to $n-1$ do
  Construct $CH(w_{|X_{k+1}}, x_{k+1})$ using the proof of Lemma 2.4.
  Use Algorithm 1 to compute $\iota^c_w(U)$ for all $U \in CH(w_{|X_{k+1}}, x_{k+1})$.
  $\mathscr{C}_{k+1} \leftarrow \{U \in CH(w_{|X_{k+1}}, x_{k+1}) : \iota^c_w(U) > 0\}$
  For all $A \in \mathscr{C}_k$ do
    Compute $\hat{r}[A, k+1]$ from $\hat{r}[A, k] \cup \{aa'|x_{k+1} : a, a' \in A\}$.
    Compute $\iota^c_w(A)$ using $\hat{r}[A, k+1]$ and [8].
    If $\iota^c_w(A) > 0$ then add A to $\mathscr{C}_{k+1}$.
    Compute $\hat{r}[A \cup \{x_{k+1}\}, k+1]$ from $\hat{r}[A, k] \cup \{ax_{k+1}|y : a \in A, y \notin A\}$.
    Compute $\iota^c_w(A \cup \{x_{k+1}\})$ using $\hat{r}[A \cup \{x_{k+1}\}, k+1]$ and [8].
    If $\iota^c_w(A \cup \{x_{k+1}\}) > 0$ then add $A \cup \{x_{k+1}\}$ to $\mathscr{C}_{k+1}$.
Output $\mathscr{C}_n$.

ALGORITHM 4 (*Compute $\mu^r_w(U|X-U)$ for all splits $U|(X-U)$ of T*).

Root T at an arbitrary vertex.
Compute and tabulate $\mathrm{lca}(x, x')$ for all leaves x, x' in T.
Iterate through all $a, b, c, d \in X$ and use lca tables to
 construct $Q[u, v]$ for all $u, v \in V(T)$.
For each pair $u, v \in V(T)$ use the algorithm of [8] to
 extract $Q[u, v, n-3] \subseteq Q[u, v]$.
For all vertices $v \in V(T)$ do
  For all vertices u for which $v \not\preceq u$ in a post-order traversal of T do
    Use Eq. (30) and [8] to construct $Q_{\preceq}[u, v, n-3]$.
For all vertices $u \in V(T)$ put $U = \mathscr{L}(u)$ and compute
  $Q_u \leftarrow \bigcup_{v \not\preceq_T u} Q_{\preceq}[u, v, n-3]$
  $\mu^r_w(U|X-U) \leftarrow \mathrm{avmin}(\{w(ab|cd) : ab|cd \in Q_u\}, n-3)$.
Output split indices $\mu^r_w(U|X-U)$ for all $U|(X-U) \in$ *splits* (T).

ALGORITHM 5 (*Refined Buneman tree for w*).

Order X arbitrarily as $x_1, x_2, \ldots, x_n$
$\mathscr{R}_3 \leftarrow \{x_i|X_3 - x_i : i = 1, 2, 3\}$
For $i = 1, 2, 3$ do
  $q_{\min}[x_i|X_3 - x_i] = q(x_i|X_3 - x_i)$
For k from 3 to $n-1$ do
  Let T be the strong cluster tree for $w_{|X_{k+1}}$.
  Use Algorithm 4 to compute $\mu^r_w(U|X_{k+1} - U)$ for all $U \in$ *clus*(T).
  $\mathscr{R}_{k+1} \leftarrow \{U|X_{k+1} - U : U \in$ *clus*(T) and $\mu^r_w(U|X_{k+1} - U) > 0\}$.

For all $U|V \in \mathcal{R}_k$ do
  Construct $q_{\min}[U \cup \{x_{k+1}\}|V, k+1]$ using Eq. (31) and [8].
  Compute $\mu_w^r(U \cup \{x_{k+1}\}|V)$ using $q_{\min}[U \cup \{x_{k+1}\}|V, k+1]$ and [8].
  If $\mu_w^r(U \cup \{x_{k+1}\}|V) > 0$ then add $U \cup \{x_{k+1}\}|V$ to $\mathcal{R}_{k+1}$.
  Construct $q_{\min}[U|V \cup \{x_{k+1}\}, k+1]$ using Eq. (31) and [8].
  Compute $\mu_w^r(U|V \cup \{x_{k+1}\})$ using $q_{\min}[U|V \cup \{x_{k+1}\}, k+1]$ and [8].
  If $\mu_w^r(U|V \cup \{x_{k+1}\}) > 0$ then add $U|V \cup \{x_{k+1}\}$ to $\mathcal{R}_{k+1}$.
Output $\mathcal{R}_n$.

ACKNOWLEDGMENTS

We thank Olivier Gascuel, Vincent Moulton, and Mike Steel for reading through versions of this manuscript.

REFERENCES

1. J. D. Apresjan, An algorithm for constructing clusters from a distance matrix, *Mashinnyi perevod: prikladnaja lingvistika* **9** (1966), 3–18.
2. H.-J. Bandelt and A. W. M. Dress, Weak hierarchies associated with similarity measures—an additive clustering technique, *Bull. Math. Biol.* **51** (1989), 133–166.
3. H.-J. Bandelt and A. W. M. Dress, A canonical decomposition theory for metrics on a finite set, *Adv. Math.* **92** (1992), 47–105.
4. J.-P. Barthélémy and A. Guénoche, "Trees and Proximity Representations," Wiley, Chichester, UK, 1991.
5. V. Berry and D. Bryant, Faster reliable phylogenetic analysis, *in* "Proc. 3rd International Conference on Comp. Mol. Biol. (RECOMB)," Vol. 3, pp. 59–68, 1999.
6. V. Berry, T. Jiang, P. Kearney, and M. Li, Quartet cleaning: Improved algorithms and simulations, *Lecture Notes Comput. Sci.* **1643** (1999), 313–324.
7. V. Berry and O. Gascuel, Inferring evolutionary trees with strong combinatorial evidence, *Theoretical Comput. Sci.* **240** (2000), 271–298.
8. M. Blum, V. Pratt, R. E. Tarjan, R. W. Floyd, and R. L. Rivest, Time bounds for selection, *J. Comput. System Sci.* **7** (1973), 448–461.
9. D. Bryant and V. Moulton, A polynomial time algorithm for constructing the refined Buneman tree, *Appl. Math. Lett.* **12** (1999), 51–56.
10. P. Buneman, The recovery of trees from measures of dissimilarity, *in* "Mathematics in the Archaeological and Historical Sciences" (F. R. Hodson, D. G. Kendall, and P. Tautu, Eds.), pp. 387–395, Edinburgh Univ. Press, Edinburgh, 1971.
11. A. Dress, Towards a theory of holistic clustering, *in* "Mathematical Hierarchies and Biology" (B. Mirkin, F. R. McMorris, F. Roberts, and A. Rzhetsky, Eds.), DIMACS Series in Discete Math. and Theoretical Comp. Science, pp. 271–290, Am. Math. Soc., Providence, 1997.
12. M. Farach, S. Kannan, and T. Warnow, A robust model for finding optimal evolutionary trees, *Algorithmica* **13** (1995), 155–179.
13. J. S. Farris, A. G. Kluge, and M. J. Eckart, A numerical approach to phylogenetic systematics, *Systematic Zool.* **19** (1970), 172–189.
14. A. D. Gordon, Hierarchial classification, *in* "Clustering and Classification" (P. Arabie, L. J. Hubert, and G. DeSoete, Eds.), pp. 65–122, World Scientific, London, 1996.

15. J. C. Gower and G. J. S. Ross, Minimum spanning tree and single linkage cluster analysis, *Appl. Statist.* **18** (1969), 54–64.
16. P. Hell and M. Rosenfeld, The complexity of finding generalized paths in tournaments, *J. Algorithms* **4** (1982), 303–309.
17. D. Huson, S. Nettles, and T. Warnow, Disk-covering, a fast converging method for phylogenetic tree reconstruction, *J. Comp. Biol.* **6** (1999), 369–386.
18. N. Jardine, Towards a general theory of clustering, *Biometrics* **25** (1969), 609–610.
19. D. W. Matula and F. Shahrokhi, Sparsest cuts and bottlenecks in graphs, *Discrete Appl. Math.* **27** (1990), 113–123.
20. V. Moulton and M. Steel, Retractions of finite distance functions onto tree metrics, *Discrete Appl. Math.* **91** (1999), 215–233.
21. F. Murtagh, Complexities of hierarchic clustering algorithms: state of the art, *CSQ* **1** (1984), 101–113.
22. A. F. Parker-Rhodes and R. M. Needham, A reduction method for non-arithmetic data, and its application to thesauric translation, *in* "Information Processing, Proceedings of the International Conference on Information Processing, Paris, 1960," UNESCO, pp. 321–325.
23. F. J. Rohlf, Algorithm 76: Hierarchical clustering using the minimum spanning tree, *Comput. J.* **16** (1973), 93–95.
24. R. R. Sokal and C. D. Michener, A statistical method for evaluating systematic relationships, *Univ. Kansas Sci. Bull.* **38** (1958), 1409–1438.