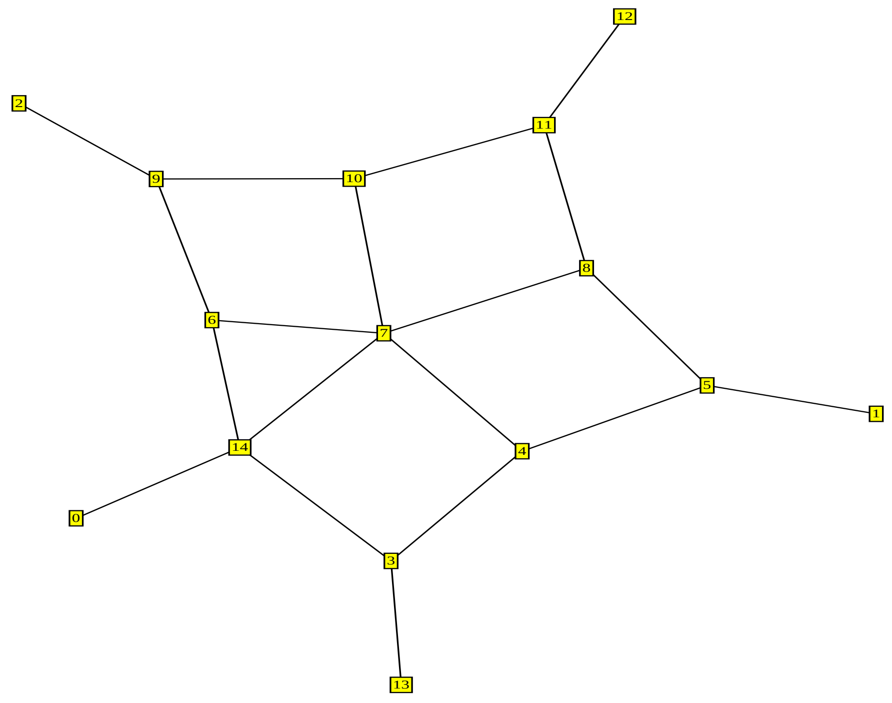
Tight Span used in Phylogenetics

Pio Korinth

April 10, 2007



Abstract

Suppose we have a set of species and that we know the genetic difference between any pair in that set. We want to figure out which species have the same ancestor/ancestors. One way of finding approximative solutions is using a mathematical tool known as *Tight Span*. I will describe what Tight Span is and also implement my own algorithm using Tight Span on computers. I will also describe a way to show how a conjecture given by Andreas W.M. Dress in [1] on pages 342-345 can be deduced from a different conjecture I have formulated as well as describing another method I have developed for construction of phylogenetic trees. This last method does not use Tight Span and has not yet been implemented.

Acknowledgments

I want especially to thank my brother Leo Korinth for helping me with the computer related parts of this project such as installing Polymake, programming in C++, which I was not used to when I began, and discussing how the algorithm should work. He came up with the idea of connecting just the nodes that are in a cycle.

My advisor professor Svante Linusson has helped me understanding Tight Span geometrically, giving me a basic idea how the algorithm could be made and has also had a big patience with me. He has also given me a lot of help and advices how the report should look like.

Alice Lesser, who I have e-mailed, has made a PhD thesis on a similar subject and has helped me a lot when I have asked questions.

Ewgenij Gawrilow who has constructed Polymake has helped me with the installation of Polymake and has also helped me along with Sven Herrmann when I got strange output from my algorithm. Thanks!!!

My mother Ann-Sofie Korinth has been a big support.

The discussions have been fun with Leo and Svante and I hope and think I have learned a lot through them.

Contents

1 Introduction

I have listed some basic definitions back in the appendix so you do not have to be a mathematician to be able to follow the lemmas and theorems. The more important definitions are in the text, hopefully making the reading easier. Throughout the text the metrics involved will be finite. If you are a biologist you can think each point of the metric represents one of the species and the distance between any two species represents the genetic difference between them.

In Section 2 you can read about *realizations*, which is a concept that relates metrics with graphs. This section will be important later on.

The main results in Section 3 are the theorems and its corollaries, which are also stated in [1]. Lemma 4, 1, 16 and 17 are also of great interest. Theorem 1 and 3 show that an extension Y of a metric space X is tight if and only if $y_1y_2{=}\sup_{x_1,x_2\in X}\ (x_1x_2 - x_1y_1 - x_2y_2)$ for all $y_1, y_2 \in Y$. Theorem 1 shows the "if" case and Theorem 3 shows the "only if" case. Theorem 2 shows that there always exists an isometric embedding of Y into T_X, where Y is a tight extension of the metric space X and T_X is the Tight Span of X. We have the following relationships:

$$\begin{tikzcd}
T_X \arrow[r, bend left, "\tau"] & T_Y \arrow[l, bend left, "\psi_2"] \\
X \arrow[u, hook, "x\mapsto h_x"] \arrow[r, hook] & Y \arrow[u, hook', "y\mapsto h_y"'] \arrow[ul, "\psi"]
\end{tikzcd}$$

I will show how a conjecture by Andreas W.M.Dress given in [1] on pages 342-345 can be deduced from a different conjecture we formulate in Section 4. My hope is that this will give you a greater understanding of the relationship between Tight Spans and optimal realizations.

In Section 5 I describe among others an algorithm which uses Tight Span in the construction of "best fitting" trees from arbitrary finite metrics. The algorithm has been implemented in C++ with the help of *Polymake* [4]. I have done this with my brother Leo Korinth. You will also see some pictures how the algorithm works. Pseudo code has also been included for better understanding.

The following Section 6 tests the algorithm on perturbed randomized trees.

The last section describes a method for making "best fitting" trees out of metrics. This method does not use Tight Span and has not yet been implemented although I have implemented a simplified version of this method in Matlab.

2 Realizations

A realization of a metric space X is a connected positive weighted graph G s.t. for any two points x, y in X the distance $d(x, y)$ corresponds to the *shortest path* between the vertices x, y in G. From now on we will denote the distance $d(x, y)$ by xy to save space. The vertices of G consist of $X \cup Y$, where Y is a set of auxiliary vertices with *degree* $\geq$ three since vertices of degree two can be contracted. That is, we can replace the two edges connected to the vertex of degree two with a single edge with weight equal the sum of the two original edges.

An optimal realization is a realization which minimizes the sum of all edge weights in G. We denote the optimal realization of X by T, where $T = (V, E, l)$, V is the vertex set, E is the edge set and l is the length function defined on E. The Tight Span of X is strongly connected with the optimal realization of X. This is the reason why Tight Span is interesting to me.

3 The Tight Span

In this chapter I will describe the basics of the Tight Span. I have done this mainly by following the works of Dress[1], filling in details in some of his proofs. I hope this will make the basics of Tight Span more accessible.

Suppose we have a finite metric space X with distance function d. Then we can define a function space P_X as:

$$P_X := \{f \in \mathbb{R}^X \mid f(x) + f(y) \geq xy \qquad \text{for all } x, y \in X\},$$

where $\mathbb{R}^X$ denotes all functions from X into the real line $\mathbb{R}$. Recall $xy := d(x, y)$. P_X is a convex space, since if $f \in \mathbb{R}^X$ and $g \in \mathbb{R}^X$ then clearly $h = \alpha f + (1 - \alpha)g \in \mathbb{R}^X$ for any scalar α, $0 \leq \alpha \leq 1$. The Tight Span T_X is defined as:

$$T_X := \{f \in \mathbb{R}^X \mid f(x) = \sup_{y \in X}(xy - f(y)) \qquad \text{for all } x \in X\}$$

The definition of T_X can be a bit difficult to grasp. We can think of T_X as the smallest functions in P_X. This will also be proved in Lemma 8.

The distance D between any two functions $f, g \in P_X$ is defined as $D(f, g) := \| f - g \|_\infty := \sup_{x \in X}(|f(x) - g(x)|)$.

Lemma 1. *T_X is a subset of P_X.*

Proof. Suppose $f \in T_X \Rightarrow f(x) = \sup_{y \in X}(xy - f(y)), \forall\ x \in X \Rightarrow f(x) \geq xy - f(y),\ \forall\ x, y \in X \Rightarrow f(x) + f(y) \geq xy,\ \forall\ x, y \in X \Rightarrow f \in P_X$. $\square$

Lemma 2. *The functions h_x, defined for each $x \in X$ as $h_x : X \to \mathbb{R} : z \mapsto xz$ for all $z \in X$, are members of T_X.*

Proof. We have that h_x is in P_X since $h_x(y) + h_x(z) = xy + xz \geq yz\ \forall y, z \in X$, i.e. $h_x(y) \geq \sup_{z \in X}(yz - h_x(z))$. Especially $h_x \in T_X$ since for all $y \in X$, $h_x(y) \geq \sup_{z \in X}(yz - h_x(z)) \geq yx - h_x(x) = xy - xx = xy = h_x(y) \Rightarrow h_x(y) = \sup_{z \in X}(yz - h_x(z))\ \forall y \in X$. $\square$

Lemma 3. *If $f \in T_X$ then $D(h_x, f) = f(x)$.*

Proof. We have two inequalities:

i) $f(z) = \sup_{y\in X}(zy - f(y)) \le \sup_{y\in X}(zx + xy - f(y)) = zx + \sup_{y\in X}(xy - f(y)) = h_x(z) + f(x)$, i.e. $f(z) - h_x(z) \le f(x)$

ii) $h_x(z) = xz \le f(x) + f(z) \Rightarrow h_x(z) - f(z) \le f(x)$

i) and ii) $\Rightarrow |h_x(z) - f(z)| \le f(x)\ \forall z \in X$; but for $z = x$ we get equality since $|h_x(x) - f(x)| = |xx - f(x)| = |-f(x)| = f(x) \Rightarrow D(h_x, f) = \sup_{z\in X}(|h_x(z) - f(z)|) = f(x)$. □

Definition of extension 1. *A space (Y, d_Y) is said to be an* ***extension*** *of a space (X, d_X) if $X \subseteq Y$, d_Y and d_X are distance functions satisfying the rules (except possibly (M2)) of a metric and also satisfying $d_Y|_X = d_X$, i.e. $d_Y(x_1, x_2) = d_X(x_1, x_2)$ for all $x_1, x_2 \in X$.*

Lemma 4. *(T_X, D) is an extension of (X, d) with respect to the isometric embedding $X \ni x \mapsto h_x \in T_X$. Let $S_X = \{h_x \mid x \in X\}$. Then $d(x, y)$ in X equals $D(h_x, h_y)$ in $S_X\ \forall x, y \in X$ where $S_X \subseteq T_X$.*

Proof. It follows from Lemma 2 and Lemma 3 that $h_x : X \to \mathbb{R} : z \mapsto xz$ is an isometry since $D(h_x, h_y) = h_y(x) = yx = xy = d(x, y)\ \forall x, y \in X$. □

Definition of tight extension 2. *A space (Y, d_Y) is said to be a* ***tight extension*** *of a space (X, d_X) if:*

I/ (Y, d_Y) is an extension of (X, d_X)

II/ There does not exist a distance function (satisfying the rules of a metric except possibly (M2)) d on Y such that (Y, d) is an extension of (X, d_X) for which $d(y_1, y_2) \le d_Y(y_1, y_2)$ for all $y_1, y_2 \in Y$ and for which there exist points $y_1, y_2 \in Y$ such that $d(y_1, y_2) < d_Y(y_1, y_2)$.

Loosely speaking, if Y is a space with the smallest possible distance function d_Y on it such that $d_Y|_X = d_X$ then Y is tight. According to this definition there may exist different tight extensions Y_1 and Y_2 of X such that $|Y_1| = |Y_2|$.

The following theorem is interesting mainly because of its corollary.

Theorem 1. *An extension Y of a metric space X is tight if $y_1y_2 = sup_{x_1,x_2\in X}(x_1x_2 - x_1y_1 - x_2y_2)$ for all $y_1, y_2 \in Y$.*

Proof. Suppose our extension Y of X is defined as: $y_1y_2 = \sup_{x_1,x_2\in X}(x_1x_2 - x_1y_1 - x_2y_2)$ for all $y_1, y_2 \in Y$. Let d be any function satisfying (M1), (M3) and (M4) and which in addition satisfies $d(x_1, x_2) = x_1x_2$ for all $x_1, x_2 \in X$ as well as $d(y_1, y_2) \le y_1y_2$ for all $y_1, y_2 \in Y$. This extension Y is tight, since for all $y_1, y_2 \in Y$ we have $y_1y_2 = \sup_{x_1,x_2\in X}(x_1x_2 - x_1y_1 - x_2y_2) \le \sup_{x_1,x_2\in X}(d(x_1, x_2) - d(x_1, y_1) - d(x_2, y_2)) \le d(y_1, y_2)$, since suppose $\sup_{x_1,x_2\in X}(d(x_1, x_2) - d(x_1, y_1) - d(x_2, y_2)) > d(y_1, y_2)$. This implies: $d(y_1, y_2) < d(x_1, x_2) - d(x_1, y_1) - d(x_2, y_2)$ for some $x_1, x_2 \in X \Rightarrow d(y_1, y_2) + d(x_1, y_1) + d(x_2, y_2) < d(x_1, x_2)$ for some

$x_1, x_2 \in X$. The triangle inequality (M4) gives us: $d(y_1, y_2) + d(x_1, y_1) + d(x_2, y_2) \geq d(y_2, x_1) + d(x_2, y_2) \geq d(x_1, x_2)$. Thus we get a contradiction which proves the theorem. □

Lemma 5. *For all functions $f, g \in T_X$, we have $sup_{x\in X}(|f(x) - g(x)|) = sup_{x\in X}(f(x) - g(x))$.*

Proof. $\sup_{x\in X}(f(x)-g(x)) = \sup_{x\in X}(\sup_{y\in X}(xy-f(y)-g(x))) = \sup_{x\in X}(xy_1 - f(y_1) - g(x)) = \sup_{x\in X}(xy_1 - g(x)) - f(y_1) = g(y_1) - f(y_1) \Rightarrow$

i) $\sup_{x\in X}(f(x) - g(x)) = g(y_1) - f(y_1) \leq \sup_{x\in X}(g(x) - f(x))$

ii) $\sup_{x\in X}(g(x) - f(x)) = f(y_2) - g(y_2) \leq \sup_{x\in X}(f(x) - g(x))$

i) together with ii) $\Rightarrow \sup_{x\in X}(f(x) - g(x)) = \sup_{x\in X}(g(x) - f(x))\ \forall f, g \in T_X$. □

Corollary 1. *We have the following corollary of Theorem 1: T_X is a tight extension of X.*

Proof. From Lemma 4 we see that T_x is an extension of X. First we show that $D(f, g) = \sup_{x,y\in X}(xy - f(y) - g(x))\ \forall f, g \in T_X$. From Lemma 5: $D(f, g) = \sup_{x\in X}(f(x) - g(x)) = \sup_{x\in X}(g(x) - f(x)) \Rightarrow \sup_{x\in X}(\sup_{y\in X}(xy - f(y) - g(x))) = \sup_{x\in X}(\sup_{y\in X}(xy - g(y) - f(x))) = \sup_{x\in X}(\sup_{y\in X}(yx - f(x) - g(y)))$. The previous equation implies: $D(f, g) = \sup_{x,y\in X}(xy - f(x) - g(y)) = \sup_{x,y\in X}(D(h_x, h_y) - D(h_x, f) - D(h_y, g))$. The proof follows from Theorem 1. □

Let us introduce the map $p_x : P_X \to P_X$ defined as:

$$p_x(f) := \begin{cases} f(z) & \text{if } z \neq x \\ \sup_{y\in X}(0, zy - f(y)) := \max(0, \sup_{y\in X}(zy - f(y))) & \text{if } z = x \end{cases}$$

The zero in the definition of p_x has to be there to make the proof of case iv) in Lemma 6 possible.

Lemma 6. *If $f \in P_X$ then $p_x(f) \in P_X$.*

Proof. We have four possibilities: i) $y, z \in X, y \neq x, z \neq x$ ii) $y, z \in X, y = x, z \neq x$ iii) $y, z \in X, y \neq x, z = x$ iv) $y, z \in X, y = z = x$. Due to symmetry ii) and iii) are equivalent.

i) $p_x(f)(y) + p_x(f)(z) = f(y) + f(z) \geq yz$

ii) $p_x(f)(x) + p_x(f)(z) = \sup_{y\in X}(0, xy - f(y)) + f(z) \geq xz$, since assume the opposite $\Rightarrow \sup_{y\in X}(0, xy-f(y))+f(z) < xz \Rightarrow \sup_{y\in X}(0, xy-f(y)) < xz - f(z) \Rightarrow$ the assumption is wrong $\Rightarrow p_x(f)(x) + p_x(f)(z) = \sup_{y\in X}(0, xy - f(y)) + f(z) \geq xz$

iii) Due to symmetry the proof of iii) is equivalent of that of ii)

iv) $p_x(f)(x) + p_x(f)(x) = 2\text{sup}_{y \in X}(0, xy - f(y)) \geq 0 = xx$

$\square$

Lemma 7. *For all $f \in P_X$, $p_x(f) \leq f$.*

Proof. We have $p_x(f)(z) = f(z)\ \forall z \in X \setminus \{x\}$ according to the definition of p_x. Assume $p_x(f)(x) > f(x)$, which means $\sup_{y \in X}(0, xy - f(y)) > f(x)$. We have $0 \leq f(x) \Rightarrow \sup_{y \in X}(xy - f(y)) > f(x) \Rightarrow xy_{sup} > f(x) + f(y_{sup}) \Rightarrow f \notin P_X$ which is a contradiction $\Rightarrow p_x(f)(x) \leq f(x)$. $\square$

Let us introduce a poset structure on P_X by pointwise comparisons (see Definition 3). Let us also label each point in X with an index, i.e. $X = \{x_1, x_2, ..., x_n\}$. We have the functions $g_k : P_X \to P_X$ defined for any $g \in P_X$ as: $g_k := p_{x_k} \circ p_{x_{k-1}} \circ ... \circ p_{x_1} \circ g$, $1 \leq k \leq n$.

Lemma 8. *T_X contains all minimal functions in P_X; and every function in T_X is minimal in P_X.*

Proof. First we show that T_X contains all minimal functions in P_X, i.e. for every function $g \in P_X$ we have a function $f \in T_X$ s.t. $f \leq g$. Since $g_k \in P_X$, we get for all $1 \leq k \leq n$, $g_k(x_k) + g_k(x_k) \geq x_k x_k = 0$, i.e. $g_k(x_k) \geq 0$. Then we have $g_n(x_k) = g_k(x_k) = \sup_{y \in X}(0, x_k y - g_{k-1}(y)) = \sup_{y \in X}(x_k y - g_{k-1}(y)) \leq \sup_{y \in X}(x_k y - g_n(y))$ for all $1 \leq k \leq n$. We cannot have $g_n(x_k) < \sup_{y \in X}(x_k y - g_n(y))$ since $g_n \in P_X$. Hence we can conclude: $g_n(x) = \sup_{y \in X}(xy - g_n(y))$. Now we can define our function $f \in T_X$ as $f := g_n$. Clearly the inequality $f \leq g$ is satisfied.

We now show that every function in T_X is minimal in P_X. Assume there exists a function $f \in T_X$, which is not minimal in P_X. This implies there exists a $g \in P_X$ s.t. $g \leq f$. We get $f(x) = \sup_{y \in X}(xy - f(y)) \leq \sup_{y \in X}(xy - g(y)) \leq g(x)$. We get a contradiction, i.e. every function in T_X is minimal in P_X.

Lemma 9. *For all $f \in T_X$, $p_x(f) = f$.*

Proof. Assume $p_x(f) < f$ for some $f \in T_X \Rightarrow f$ is not minimal in $P_X \Rightarrow f \notin T_X \Rightarrow p_x(f) \geq f$, but $p_x(f) \leq f$ from Lemma 7 $\Rightarrow p_x(f) = f\ \forall f \in T_X$. $\square$

Lemma 10. *$D(p_x(f), p_x(g)) \leq D(f, g)$ for all $f, g \in P_X$.*

Proof. Since $p_x(f)(y) - p_x(g)(y) = f(y) - g(y)\ \forall y \neq x$ it suffices to show that $|p_x(f)(x) - p_x(g)(x)| \leq D(f, g)$. We get the inequality $p_x(f)(x) = \sup_{y \in X}(0, xy - f(y)) = \sup_{y \in X}(0, (xy - g(y)) + (g(y) - f(y))) \leq \sup_{y \in X}(0, xy - g(y)) + \sup_{y \in X}(0, g(y) - f(y)) \leq p_x(g)(x) + D(f, g) \Rightarrow$

i) $p_x(f)(x) - p_x(g)(x) \leq D(f, g)$

ii) $p_x(g)(x) - p_x(f)(x) \leq D(f, g)$

i) and ii) imply that $|p_x(f)(x) - p_x(g)(x)| \leq D(f,g)$.

□

Define the set $\Phi(X)$ as the set of all maps $p : P_X \to P_X$ satisfying:

I/ $p(f) \leq f \ \forall f \in P_X$

II/ $D(p(f), p(g)) \leq D(f,g) \ \forall f, g \in P_X$

We define a partial order on $\Phi(X)$ as: $\Phi(X) \ni p_1 \leq p_2 \in \Phi(X)$ if $p_1(f) \leq p_2(f) \ \forall f \in P_X$ and $D(p_1(f), p_1(g)) \leq D(p_2(f), p_2(g)) \ \forall f, g \in P_X$.

Lemma 11. *The set $\Phi(X)$ contains minimal elements with respect to the partial order defined on $\Phi(X)$.*

Proof. We saw in proof of Lemma 8 that the map $p_{x_n} \circ p_{x_{n-1}} \circ ... \circ p_{x_1}, P_X \ni g \mapsto g_n \in T_X$ was minimal with respect to I/. From Lemma 10 we also see (using induction) that II/ is satisfied, which proves the theorem. □

Lemma 12. *For any minimal $p \in \Phi(X)$, $p(P_X) \subseteq T_X$.*

Proof. For all $x \in X$ we have from Lemma 7: $p_x(p(f)) \leq p(f)$ for all $f \in P_X \Rightarrow p_x \circ p = p$, since otherwise p would not be minimal. We see (using induction) that $p_{x_n} \circ p_{x_{n-1}} \circ ... \circ p_{x_1} \circ p = p$. From proof of Lemma 8 with $g = p(f)$ we get $p_{x_n} \circ p_{x_{n-1}} \circ ... \circ p_{x_1} \circ p(f) \in T_X$, i.e. $p(f) \in T_X$. □

Let Y be an extension of X and take an arbitrary $x \in X$ The function $\alpha_x : T_X \to P_Y : f \mapsto f'$ is defined below.

$$\alpha_x(f) := \begin{cases} f'(y) = f(y) & \text{if } y \in X \\ f'(y) = xy + f(x) \text{ for some fixed x} \in X & \text{if } y \in Y \setminus X \end{cases}$$

We will now show f' is in P_Y as claimed above. For any $x_1, x_2 \in X$ we have $f'(x_1) + f'(x_2) = f(x_1) + f(x_2) \geq x_1 x_2$. We have three other possibilities:

i) $x_1 \in Y \setminus X$, $x_2 \in X$. When this happens $f'(x_1) + f'(x_2) = xx_1 + f(x) + f(x_2) \geq xx_1 + xx_2 \geq x_1 x_2$

ii) $x_1 \in X$, $x_2 \in Y \setminus X$ The proof of this follows from i) due to symmetry

iii) $x_1, x_2 \in Y \setminus X$. When this happens $f'(x_1) + f'(x_2) = xx_1 + xx_2 + 2f(x) \geq x_1 x_2 + 2f(x) \geq x_1 x_2$

Lemma 13. *Let Y be any extension of X, $f \in T_X$ and $f' = \alpha_x(f)$, then $p(f')|_X \in P_X$ for any minimal $p \in \Phi(Y)$.*

Proof. We showed above: $f' \in P_Y$. We have from Lemma 12: $p(f') \in T_Y$. This implies $p(f') \in P_Y \Rightarrow p(f')(x) + p(f')(y) \geq xy \ \forall x, y \in Y \Rightarrow p(f')(x) + p(f')(y) \geq xy \ \forall x, y \in X \Rightarrow p(f')|_X \in P_X$. □

Define the function $\tau : T_X \to T_Y$ as the composition of p with α_x, i.e. $\tau = p \circ \alpha_x$, where Y is an extension of X, p is minimal in $\Phi(Y)$ and $x \in X$.

Lemma 14. *For any minimal $p \in \Phi(Y)$ and any function $f \in T_X$ s.t. $f' = \alpha_x(f)$, we have $p(f')|_X = f$, i.e. $\tau(f)|_X = f$.*

Proof. $p(f')|_X \in P_X$ and $p(f') \leq f' \Rightarrow p(f')|_X \leq f'|_X = f \in T_X \Rightarrow \{$ from Lemma 8 $\} \Rightarrow p(f')|_X = f$. □

Lemma 15. *$D(f', g') = D(f, g)$ for any two functions $f, g \in T_X$, where $f' = \alpha_x(f)$ and $g' = \alpha_x(g)$.*

Proof. $D(f', g') = sup_{x \in Y}(|f'(x) - g'(x)|) = max(D(f', g')|_X, D(f', g')|_{Y \setminus X}) = D(f, g)$, since $D(f', g')|_X = D(f, g)$ and $D(f', g')|_{Y \setminus X} = sup_{y \in Y \setminus X}(|xy + f(x) - (xy + g(x))|) = |f(x) - g(x)| \leq D(f, g)$. □

Lemma 16. *If Y is an extension of X, then τ is an isometry from T_X into T_Y, i.e. $\tau : T_X \hookrightarrow T_Y$.*

Proof. $D(f, g) = \{$ from Lemma 14 $\} = D(\tau(f)|_X, \tau(g)|_X) \leq D(\tau(f), \tau(g)) = D(p(f'), p(g')) \leq D(f', g') = \{$ from Lemma 15 $\} = D(f, g) \Rightarrow D(f, g) = D(\tau(f), \tau(g)) \Rightarrow \tau : T_X \hookrightarrow T_Y$ is an isometry. □

Lemma 17. *If Z is a tight extension of Y and Y is a tight extension of X, then Z is a tight extension of X.*

Proof. Assume Z is not a tight extension of $X \Rightarrow$ there exists a distance function d and points $z_1, z_2 \in Z$, s.t. $d(z_1, z_2) < z_1 z_2$. We have five possibilities to consider:

i) $z_1, z_2 \in Z \setminus Y$

ii) $z_1, z_2 \in Y \setminus X$

iii) $z_1 \in Z \setminus Y, z_2 \in Y \setminus X$

iv) $z_1 \in Y \setminus X, z_2 \in X$

v) $z_1 \in Z \setminus Y, z_2 \in X$

The options i), iii) and v) are not possible since this would imply that Z is not a tight extension of Y. The options ii) and iv) are not possible since this would imply that Y is not a tight extension of X. □

Definition of contraction map 3. *A map $\phi : Y \to E$ from a tight extension Y of a metric space X to an extension E of X satisfying $d_E(\phi(y_1), \phi(y_2)) \leq d_Y(y_1, y_2)$ for all $y_1, y_2 \in Y$ and $\phi(x) = x$ for all $x \in X$ is called a contraction map.*

Lemma 18. *The contraction map $\phi : Y \to E$ from a tight extension Y of a metric space X to an extension E of X is an isometry.*

Proof. Define the distance function $d'_Y : Y \times Y \to \mathbb{R}$ as $d'_Y(y_1, y_2) := d_E(\phi(y_1), \phi(y_2))$. Assume ϕ is not an isometry, i.e. (Y, d_Y) is not isometric to $(E, d_E) \Rightarrow d'_Y(y_1, y_2) = d_E(\phi(y_1), \phi(y_2)) \neq d_Y(y_1, y_2)$ for some $y_1, y_2 \in Y \Rightarrow d'_Y(y_1, y_2) = d_E(\phi(y_1), \phi(y_2)) < d_Y(y_1, y_2)$ for some $y_1, y_2 \in Y$, since ϕ was a contraction map. We have

i) $d'_Y(x_1, x_2) = d_E(\phi(x_1), \phi(x_2)) = d_E(x_1, x_2) = d_X(x_1, x_2)\ \forall x_1, x_2 \in X$

ii) $d'_Y(y_1, y_2) = d_E(\phi(y_1), \phi(y_2)) \leq d_Y(y_1, y_2)\ \forall y_1, y_2 \in Y$

iii) $d'_Y(y_1, y_2) < d_Y(y_1, y_2)$ for some $y_1, y_2 \in Y$

i) says that (Y, d'_Y) is an extension of (X, d_X). ii) together with iii) implies that (Y, d_Y) is not a tight extension of X. This implies that the assumption is wrong, i.e. ϕ is an isometry. □

Let us define a map $\psi_1 : T_Y \to \tau(T_X) \subseteq T_Y, f \mapsto \tau(p(f|_X))$, where p is minimal in $\Phi(X)$ and where Y is a tight extension of X.

Lemma 19. *The map ψ_1 is a contraction map.*

Proof. T_Y is a tight extension of Y (by Corollary 1) and Y is a tight extension of X, hence (by Lemma 17) T_Y is a tight extension of X. We get that $\tau(T_X)$ is an extension of X since T_X is an extension of X and τ is an injective function and isometry between T_X and $\tau(T_X)$, i.e. T_X and $\tau(T_X)$ are isomorphic. We thus have that $x \mapsto \tau(h_x)\ \forall x \in X$ is an isometric embedding of X.

Take any two functions $f, g \in T_Y$. $D(\psi_1(f), \psi_1(g)) = D(\tau(p(f|_X)), \tau(p(g|_X))) = D(p(f|_X), p(g|_X)) \leq D(f|_X, g|_X) \leq D(f, g)$.

The only thing left to show is that $\psi_1(\tau(h_x)) = \tau(h_x)\ \forall x \in X$. This is easily shown using the facts $\tau(f)|_X = f$ and $p(f) = f\ \forall f \in T_X$, which implies $\psi_1(\tau(h_x)) = \tau(p(\tau(h_x)|_X)) = \tau(p(h_x)) = \tau(h_x)\ \forall x \in X$. □

Let us define a map $\psi_2 : T_Y \to T_X, f \mapsto p(f|_X)$, where p is minimal in $\Phi(X)$ and where Y is a tight extension of X.

Lemma 20. *T_Y is isometric to T_X, where Y is a tight extension of X.*

Proof. From Lemma 18 and 19 we get that T_Y is isometric to $\tau(T_X)$. This implies that T_Y is isometric to T_X, since T_X is isomorphic to $\tau(T_X)$. One isometry between T_Y and T_X is thus the map ψ_2. □

Lemma 21. *The function $\psi_2 : T_Y \to T_X$ is surjective for every tight extension Y of X.*

Proof. We have the function $\tau : T_X \to T_Y$ defined earlier satisfying $\tau(f)|_X = f$ for all $f \in T_X$. Take any $f \in T_X \Rightarrow f = p(f) = p(\tau(f)|_X) = \psi_2(\tau(f)) \Rightarrow \psi_2$ is surjective since f was an arbitrary function in T_X and $\tau(f) \in T_Y$. □

Lemma 22. *The function $\psi_2 : T_Y \to T_X$ is an isomorphism for every tight extension Y of X.*

Proof. ψ_2 is an isomorphism between T_Y and T_X since it is both surjective (see Lemma 21) and an isometry (see proof of Lemma 20). □

Theorem 2. *The function $\psi : Y \to T_X := y \mapsto h_y|_X$ is an isometric embedding of Y into T_X, where Y is a tight extension of X.*

Proof. From Lemma 21 we have: $\psi_2(\tau(f)) = f$, i.e. $\tau : T_X \to T_Y$ is the inverse isomorphism. For any $f \in T_Y$ we have that $\psi_2(f) = \tau(\psi_2(f))|_X = f|_X$, i.e. the retraction p is unnecessary in the the definition of ψ_2. We then have that the restriction map $f \mapsto f|_X$ induces an isomorphism between T_Y and T_X, where Y is tight of X.

Since $y \mapsto h_y$ is an isometric embedding of Y into T_Y and T_Y and T_X are isomorphic via $\psi_2 : h_y \mapsto h_y|_X$, this implies that $\psi : Y \to T_X := y \mapsto h_y|_X$ is an isometric embedding of the tight extension Y of X into T_X. □

The map below over the spaces and their functions is a good way of summarizing the basic theory of Tight Span.

$$\begin{tikzcd}
T_X \arrow[r, bend left, "\tau"] & T_Y \arrow[l, bend left, "\psi_2"] \\
X \arrow[u, hook, "x\mapsto h_x"] \arrow[r, hook] & Y \arrow[u, hook', "y\mapsto h_y"'] \arrow[ul, "\psi"]
\end{tikzcd}$$

Theorem 3. *An extension Y of a metric space X is tight only if $y_1y_2 = sup_{x_1,x_2\in X}(x_1x_2 - x_1y_1 - x_2y_2)$ for all $y_1, y_2 \in Y$.*

Proof. Assume we have an extension Y of X which is tight. $y_1y_2 = \{$ from Theorem 2 $\} = D(\psi(y_1), \psi(y_2)) = \{$ from proof of Corollary 1 with $f = \psi(y_1) \in T_X$ and $g = \psi(y_2) \in T_X$ $\} = \sup_{x_1,x_2\in X}(x_1x_2 - \psi(y_1)(x_1) - \psi(y_2)(x_2)) = \sup_{x_1,x_2\in X}(x_1x_2 - h_{y_1}|_X(x_1) - h_{y_2}|_X(x_2)) = \sup_{x_1,x_2\in X}(x_1x_2 - y_1x_1 - y_2x_2)$ for all $y_1, y_2 \in Y$. □

4 Why Tight Spans are interesting

One interesting property of Tight Span is that if the under laying metric X can be realized by a tree, then the Tight Span T_X will give us just that tree. This is proved in [1] on pages 359-364.

Now I am going to state a conjecture given by Andreas W.M.Dress [1] on pages 342-345. The conjecture is:

Conjecture 1. *Let $T = (V, E, l)$ be an optimal realization of the finite metric space $X \subseteq V$. Let $\psi : V \to T_X$ be defined by $\psi(v) = p(h_v|_X)$ for all $v \in V$. Then the maps $\psi : V \to T_X$ are injective.*

I will give you an idea of how Conjecture 1 may be proved. First I prove a Lemma below (23), and then I state another Lemma (2). A proof of the second Lemma (2) will prove Conjecture 1.

Lemma 23. *If (X, d) is a finite metric space, then every optimal realization $T = (V, E, l)$ of (X, d) is a metric space (V, l) s.t. $l|_X = d$, i.e. (V, l) is an extension of (X, d).*

Proof. We can define a distance function l as: $l(x, y) := shortest\_path(x, y)$ for all $x, y \in V$ since T was a realization of X. We have that (V, l) is a metric, since $l(x, y) \leq l(x, z) + l(z, y)$. Assume this is not the case then we have $shortest\_path(x, y) > shortest\_path(x, z) + shortest\_path(z, y)$. This is not possible since it would imply that there exists a shorter path from x to y than $shortest\_path(x, y)$. The rest of the requirements of a metric ($(M1)$ to $(M3)$) are obvious. □

Conjecture 2. *If $T = (V, E, l)$ is an optimal realization of (X, d), then (V, l) is a tight extension of (X, d).*

Assume (V, l) is not a tight extension of (X, d). If this is the case we have an extension (V, l') of (X, d) s.t. $l'(x, y) \leq l(x, y)$ for all $x, y \in V$ and at least two points v_1 and v_2 in V for which $l'(v_1, v_2) < l(v_1, v_2)$.

If we can prove that this implies that at least one distance $l'(x_e, y_e)$ in the path between v_1 and v_2 in (V, l') is shorter than the corresponding edge in (V, E, l), then the proof of Conjecture 1 will follow from the arguments below.

Each edge is used in (V, E, l) since otherwise $T = (V, E, l)$ would not be optimal. This implies that there exists points s, t in X s.t. we use the edge (x_e, y_e) when calculating $shortest\_path(s, t) = d(s, t)$ in (V, E, l). If we walk the same path from s to t in (V, l') we get a shorter distance since $l'(x, y) \leq l(x, y)$ for all $x, y \in V$ and $l'(x_e, y_e) < l(x_e, y_e)$, i.e. $l'(s, t) < d(s, t)$ since otherwise we would violate the triangle inequality; thus (V, l') is not an extension. We get a contradiction to the assumption which implies that (V, l) is a tight extension of (X, d).

Theorem 4. *Conjecture 2 imply Conjecture 1.*

From Theorem 2 we saw that if V was a tight extension of X, then $\psi : V \to T_X$ defined as $V \ni v \mapsto h_v|_X \in T_X$ was an isometric embedding of V into T_X and thus injective. Since $f|_X = p(f|_x)$ for all f in T_V and in particular $h_v|_X = p(h_v|_x)$ for all v in V, the truth of Conjecture 1 follows from Conjecture 2.

5 Algorithm using Tight Span

I have constructed an algorithm in C++, which uses Tight Span from Polymake to construct trees from given metrics. It can be of great interest to see how good the algorithm is, so I have also made an algorithm which makes metrics from random trees and then perturbs them so they will no longer be trees. This algorithm can be used among others for statistical purposes, to evaluate the power of the tree making algorithm, and also to compare my algorithm (called ts2tree) with other tree making algorithms. In the subsections down under you can read in detail what the different programs (create random tree, modify tree and treeify) do, and see how treeify both in picture and pseudo code works.

5.1 Create random tree

Here we can construct random binary trees with optional number of leaves (species). Let say that we have chosen to create a tree with n leaves. The algorithm *create random tree* (or for short *crt*) consists of two separated recursive functions. The first function is called *create tree*, where you create the random topology of the tree. The randomness of *create tree* is such that it creates recursively a left tree with k number of leaves and a right tree with $n-k$ number of leaves. The integer k is an integer uniformly distributed between 1 and $n-1$. The second function is called *build graph*. This function calls *create tree* (takes *create tree* as an argument) and calculate the random (uniformly distributed between 1 and max_edge_length, which I have set to three) length of each edge in the tree and finally return the matrix of this tree metric.

5.2 Modify tree

From *crt* we get a tree metric matrix M. In this algorithm we want to mimic both the errors coming from the evolving nature and those coming from the way we define the distances between any pair of species. These distances can be defined for example as the Hamming distance between some comparable DNA-sequences in the species. In –modify-tree we can perturb the trees in two different ways, either –normal or –uniform and we can also precise the amount of deviation from the tree metric we want with <deviation>. If we choose –uniform and some deviation δ, the program returns for each distance $d(x,y)$ in M the number $d(x,y) + d(x,y) * \delta * (1 - 2 * \text{random}())$, where random() gives a uniformly distributed number between zero and one, to M. The perhaps more natural way of perturbing the tree comes from –normal. In this method I have thought it reasonable to assume that every unit distance is a stochastic variable normally distributed around the unit distance with standard deviation equal to that of <deviation>. To randomize the lengths I have transformed the uniformed distribution to normal distribution through a process called Box-Muller transform. There is a possibility that the perturbed metric is not a metric in which case we could get strange out puts from –treeify, since the Tight Span needs a metric. The program can handle this and will ask you to decrease the deviation and hence increase the probability to get a perturbed tree, which is a metric.

5.3 Treeify

This is the main program. It uses the well known Dijkstra's method (see [6]) to compute the shortest path between a source point (start point) and its target points (finish points) in undirected positive weighted connected graphs. It also

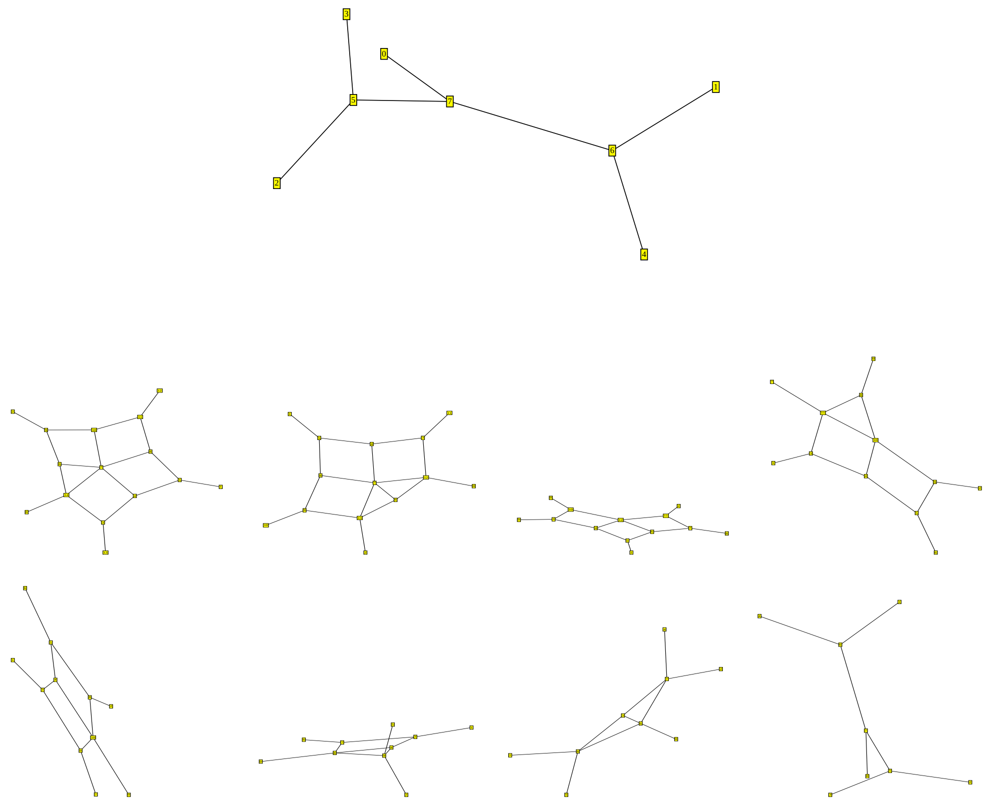


Figure 1: Upper picture shows original tree structure. The eight small pictures, from left to right, represent the progress of the algorithm.

uses the Tight Span computed from the program *Polymake*[4]. The main idea is following.

1 Compute the weighted Tight Span graph G from *Polymake*, i.e. its vertex set V and its edge set E with corresponding weights $w(E)$. We also need the subset $V_{interior}$ consisting of all vertices in V which is not the original points in the metric, i.e. a vertex from $V_{interior}$ is not one of the species. In the same way we define $E_{interior}$ as $E|_{V_{interior} \times V_{interior}}$ and its weight function $w_{interior}$ as $w|_{E_{interior}}$. Each vertex v in $V_{interior}$ has an integer weight $W(v)$ which will be

used in step four (see 4). When the algorithm starts $W(v) = 1$ for all vertices v in $V_{interior}$. Let $V_{species} = V_{exterior} := V \setminus V_{interior}$.

2 Test all the possibilities to identify vertices from $V_{interior}$ with their neighbours in $V_{interior}$ if and only if they are in a cycle.

3 Evaluate how good it is to identify one vertex with another vertex (in 2). We pick out the two vertices which minimizes the error if joined. In ts2tree you have the option to choose which error you want to minimize. You have three options. The first one is called –minimum-sum, which minimizes $\sum_{x,y \in V_{species}} |d(x,y) - shortest(x,y)|$, where $shortest(x,y)$ is the shortest path between the species x and y in G. The second option is –minimum-square, which minimizes $\sum_{x,y \in V_{species}} (d(x,y) - shortest(x,y))^2$. The third and last option is called –supremum, which minimizes $\sup_{x,y \in X}(|d(x,y) - shortest(x,y)|)$. This is probably the most interesting choice, since the Tight Span is constructed by the sup function.

4 From now on we have chosen one of the three options and have contracted the pair (x_{min}, y_{min}) to a single point z. The new point z will get the coordinates of the old coordinates as $z_{coord} := (W(x_{min}) * x_{min.coord} + W(y_{min}) * y_{min.coord})/(W(x_{min}) + W(y_{min}))$, the weight $W(z) := W(x_{min}) + W(y_{min})$ and the edge set of z will be defined as the union of the edge sets of x_{min} and y_{min}. The weights are needed to make the algorithm not as dependent of the order we join the vertices and also in a sense say that each vertex of $V_{interior}$ will play as big role as any one else in the end. We can now define the distances between z and its neighbours and can thus remove x_{min}, y_{min} and there respective edge sets from the graph and add the new point z along with its edges to the graph (see picture 1).

5 When we have done step 4, we have got a graph with one vertex less. We iterate 2 ,3 ,4 as long we have cycles in the graph. When we have no cycles left, we have a tree.

6 There might be some branches in the tree that will not reach out to species. If this is the case you can simplify this tree to a tree where all branches reach out to species, i.e. all unnecessary edges will be deleted. This is done with the option [–simplify]. The simplified tree will get new edge lengths as the method weights in the extra nodes in the simplified tree. In most cases this will yield a greater error, though the topology of the simplified tree is more interesting. You can also just ignore the extra branches from the original tree if you are more interested of the lengths of the tree than you are of the topology.

The pseudo code below of Treeify can be interesting to read.

```
TREEIFY(g, errortype())
while g $\neq$ cyclic
    error $\leftarrow \infty$
    for each edge(i, j) $\in$ ($V_{interior} \times V_{interior}$)
        g' $\leftarrow$ identify(g, i, j)
        if errortype(g, g') < error
            error $\leftarrow$ errortype(g, g')
            $g_{temp}$ $\leftarrow$ g'
    TREEIFY($g_{temp}$, errortype())
```

6 Testing ts2tree

I have made two random trees to test how good the algorithm ts2tree is. The first tree consists of six species and the second tree consists of eight species. For both trees I have added two uniform deviations ($u(\delta)$) and two normal deviations ($n(\delta)$). For each deviation I have run all three algorithms (minimum sum, minimum square and supremum). To check whether or not the algorithms give the original tree from the perturbed one I have also simplified all trees using the option [–simplify] described in Section 5.3 and then checked if the trees have the same structure. The metrics of the original trees and the results of the computer calculations are displayed in the tables on the following two pages.

The columns in the calculation tables show the results of the three different ways of identifying points in the Tight Span. The rows show the three different types of deviation, where u stands for uniform deviation and n stands for normal deviation, along with the deviations of the simplified trees. They also show if the final result is topological isomorphic with the original tree (before the perturbation).

An upper bound of the running time of the algorithm is (k^5), where k is the number of vertices obtained from Polymake's Tight Span function. This can be shown using the worst case scenario, i.e. we get a complete graph from the Tight Span. If this happens, we have to check $O(k^2)$ possible ways of identifying points. For every identification we use the Dijkstra algorithm to calculate the shortest paths, which is $O(k^2)$. It clearly takes $O(k^4)$ number of operations to get a graph with one vertex less (following the arguments above). We can only get a graph with one vertex less $O(k)$ times before we get our tree. So the total running time of the algorithm is $O(k^5)$.

$$\textbf{Tree 1's metric} = \begin{pmatrix} 0 & 9 & 9 & 5 & 5 & 6 \\ 9 & 0 & 4 & 6 & 10 & 13 \\ 9 & 4 & 0 & 6 & 10 & 13 \\ 5 & 6 & 6 & 0 & 6 & 9 \\ 5 & 10 & 10 & 6 & 0 & 9 \\ 6 & 13 & 13 & 9 & 9 & 0 \end{pmatrix}$$

Tree 1	min.sum	min.sqr	supremum
mean sum u(0.2)	0.5190	0.4042	0.5099
simplified	0.5111	0.4042	0.5111
mean sqr u(0.2)	0.1087	0.0842	0.1106
simplified	0.1131	0.0842	0.1131
supremum u(0.2)	0.9877	1.0227	1.0079
simplified	1.0521	1.0227	1.0521
top. iso.	false	true	false
mean sum u(0.1)	0.2062	0.2062	0.1988
simplified	0.2062	0.2062	0.1988
mean sqr u(0.1)	0.0439	0.0439	0.0469
simplified	0.0439	0.0439	0.0469
supremum u(0.1)	0.4226	0.4226	0.4622
simplified	0.4226	0.4226	0.4622
top. iso.	true	true	true
mean sum n(0.15)	0.8885	0.5712	0.7181
simplified	0.8821	0.5712	0.7110
mean sqr n(0.15)	0.1987	0.1225	0.1538
simplified	0.1974	0.1225	0.1532
supremum n(0.15)	2.3488	1.2307	1.5539
simplified	2.365	1.2307	1.5539
top. iso.	false	true	false
mean sum n(0.075)	0.1721	0.1721	0.1883
simplified	0.1936	0.1936	0.1936
mean sqr n(0.075)	0.0372	0.0372	0.0389
simplified	0.0298	0.0298	0.0298
supremum n(0.075)	0.4305	0.4305	0.3754
simplified	0.4191	0.4191	0.4191
top. iso.	true	true	true

$$\textbf{Tree 2's metric} = \begin{pmatrix} 0 & 13 & 14 & 12 & 11 & 12 & 12 & 11 \\ 13 & 0 & 7 & 5 & 8 & 13 & 13 & 12 \\ 14 & 7 & 0 & 4 & 9 & 14 & 14 & 13 \\ 12 & 5 & 4 & 0 & 7 & 12 & 12 & 11 \\ 11 & 8 & 9 & 7 & 0 & 11 & 11 & 10 \\ 12 & 13 & 14 & 12 & 11 & 0 & 4 & 7 \\ 12 & 13 & 14 & 12 & 11 & 4 & 0 & 7 \\ 11 & 12 & 13 & 11 & 10 & 7 & 7 & 0 \end{pmatrix}$$

Tree 2	min.sum	min.sqr	supremum
mean sum u(0.15)	1.1804	1.2097	1.1909
simplified	1.1970	1.2097	1.1909
mean sqr u(0.15)	0.1900	0.1943	0.1869
simplified	0.1932	0.1943	0.1869
supremum u(0.15)	2.4926	2.4755	2.4599
simplified	2.5127	2.4755	2.4599
top. iso.	false	false	false
mean sum u(0.075)	0.3724	0.3724	0.3960
simplified	0.3886	0.3886	0.3960
mean sqr u(0.075)	0.0587	0.0587	0.0645
simplified	0.0615	0.0615	0.0645
supremum u(0.075)	0.9716	0.9716	1.0557
simplified	0.9575	0.9575	1.0557
top. iso.	true	true	true
mean sum n(0.1)	0.7093	0.7055	1.0992
simplified	0.8101	0.8101	1.1409
mean sqr n(0.1)	0.1199	0.1190	0.1718
simplified	0.1322	0.1322	0.1809
supremum n(0.1)	1.6806	1.6717	2.2302
simplified	1.9935	1.9935	2.2999
top. iso.	false	false	false
mean sum n(0.05)	0.4707	0.5710	0.5710
simplified	0.4941	0.5710	0.5710
mean sqr n(0.05)	0.0769	0.0901	0.0901
simplified	0.0791	0.0901	0.0901
supremum n(0.05)	1.2431	1.4765	1.4765
simplified	1.2268	1.4765	1.4765
top. iso.	true	false	false

As you can see, the simplified trees often give a slightly bigger error than the unsimplified ones. I have added all errors from the unsimplified trees. From this I have concluded that minimum square is on average best (with error 14.6642 and five out of eight tests give correct tree structure), minimum sum comes second (with error 15.9304 and four out of eight tests give correct tree structure) and supremum comes last (with error 16.3573 and three out of eight tests give correct tree structure). The interesting thing is that even if the tree structure is not completely right it is always really close. The only error I have encountered happens when the algorithm contracts short edges.

What is the weakness and strength of ts2tree? The strength of ts2tree is mainly the accuracy of getting the right tree. The larger deviations of both trees sometimes give a matrix which is not a metric, so it is really good of the algorithm to get so many trees topologically correct. The weakness is mainly the speed. Suppose the Tight Span returns $O(n^4)$ vertices (if n is the number of species). The running time of Treeify will become $O(n^{20})$, i.e. if we double the number of species, then it will take approximately one million times longer to solve the problem.

7 Another approach

In genetics it is interesting to find best fitting trees (trees that realize the metric (X, d) as good as possible). Here I want to find a d' which minimizes $\sum_{x,y \in X} \mid d'(x,y) - d(x,y) \mid = \gamma$, where d' is a tree metric defined on X. To make trees you have to connect some points in X. To choose points to connect we want to know how much error this causes. We want to minimize thees errors. First we get some useful notation and definitions.

7.1 Notation and definitions

Let $|X|$ denote the number of points in X and let (X, d) be a finite metric space. Define a tree structure M which has properties:

1) $\mathrm{V}_q^{l(q)} \in M$ is associated with a subset $V_q \subset X$

2) $l(q) = |V_q|$, i.e. $l(q)$ stands for the number of leaves in $\mathrm{V}_q^{l(q)}$

3) $\mathrm{V}_q^{l(q)}$ has pointers (edges) to two subtrees $\mathrm{V}_q^{l(q)^1} \in M$ and $\mathrm{V}_q^{l(q)^2} \in M$ s.t. $\mathrm{V}_q^1 \cup \mathrm{V}_q^2 = V_q$, $\mathrm{V}_q^1 \cap \mathrm{V}_q^2 = \{\phi\}$

4) $\mathrm{V}_q^{l(q)}$ contains the submetric $d(x,y)|_{V_q}$ defined on V_q

5) Between any two $\mathrm{V}_q^{l(q)^1}$ and $\mathrm{V}_q^{l(q)^2}$ there is defined a distance $D(\mathrm{V}_q^{l(q)^1}, \mathrm{V}_q^{l(q)^2}) := \frac{(\sum_{x \in \mathrm{V}_q^1, y \in \mathrm{V}_q^2} d(x,y)) - (D(\mathrm{V}_q^{l(q)^1}) + D(\mathrm{V}_q^{l(q)^2}))}{l(q)^1 l(q)^2}$

6) $\mathrm{V}_q^{l(q)}$ has a number $D(\mathrm{V}_q^{l(q)})$, which is defined as the sum of the lengths to the root $\mathrm{V}_q^{l(q)}$, i.e. $D(\mathrm{V}_q^{l(q)}) := D(\mathrm{V}_q^{l(q)^1}) + D(\mathrm{V}_q^{l(q)^2}) + D(\mathrm{V}_q^{l(q)^1}, \mathrm{V}_q^{l(q)^2})$

7) If $l(q) = 1$ then $D(\mathrm{V}_q^{l(q)=1}) := 0$

8) The edge $(\mathrm{V}_q^{l(q)^1}, \mathrm{V}_q^{l(q)})$ has a length defined as: $D(\mathrm{V}_q^{l(q)^1}, \mathrm{V}_q^{l(q)}) := \frac{D(\mathrm{V}_q^{l(q)^1}, \mathrm{V}_q^{l(q)^2})}{2} - \alpha(\mathrm{V}_q^{l(q)^1}, \mathrm{V}_q^{l(q)^2})$

   The edge $(\mathrm{V}_q^{l(q)^2}, \mathrm{V}_q^{l(q)})$ has a length defined as: $D(\mathrm{V}_q^{l(q)^2}, \mathrm{V}_q^{l(q)}) := \frac{D(\mathrm{V}_q^{l(q)^1}, \mathrm{V}_q^{l(q)^2})}{2} + \alpha(\mathrm{V}_q^{l(q)^1}, \mathrm{V}_q^{l(q)^2})$

9) $\alpha(\mathrm{V}_q^{l(q)^1}, \mathrm{V}_q^{l(q)^2}) := \frac{\sum_{a \in X \setminus V_q} (D(\mathrm{V}_q^{l(q)^2}, a) - D(\mathrm{V}_q^{l(q)^1}, a))}{2(|X| - (l(q)^1 + l(q)^2))}$

10) $D(\mathrm{V}_q^{l(q)}, a) := \frac{l(q)^1 D(\mathrm{V}_q^{l(q)^1}, a) + l(q)^2 D(\mathrm{V}_q^{l(q)^2}, a) - (l(q)^1 D(\mathrm{V}_q^{l(q)^1}, \mathrm{V}_q^{l(q)}) + l(q)^2 D(\mathrm{V}_q^{l(q)^2}, \mathrm{V}_q^{l(q)}))}{l(q)^1 + l(q)^2}$

11) If $l(q) = 1$ then $D(\mathrm{V}_q^{l(q)=1}, a) := d(x, a)$ where x is the only point in V_q

12) The distance $G(x, y)$ between any two points $x, y \in V_q$ is defined to be the sum of all edge lengths in the path between x and y. $G(x, x)$ is defined to be zero for all points $x \in V_q$

13) The distance $G(x, a)$ between any points $x \in V_q, a \in X \setminus V_q$ is defined to be the sum of all edge lengths in the path between x and $\mathrm{V}_q^{l(q)}$ added to the distance $D(\mathrm{V}_q^{l(q)}, a)$

14) The tree $\mathrm{V}_q^{l(q)}$ has an error $ERROR(\mathrm{V}_q^{l(q)}) := \frac{\sum_{x \in V_q} \sum_{y \in X} |d(x,y) - G(x,y)|}{\frac{l(q)(l(q)-1)}{2} + l(q)(|X| - l(q))}$

7.2 Step 1

X is partitioned in trees, i.e. $X = \{\mathrm{V}_1^{l(1)}, \mathrm{V}_2^{l(2)}, ..., \mathrm{V}_n^{l(n)}\}$. When the algorithm starts $l(1) = l(2) = ... = l(n) = 1$, i.e. $V_1, V_2, ..., V_n$ are single points.

7.3 Step 2

For all $i, 1 \leq i \leq n$ join V_i with V_j for all $j, 1 \leq j \leq n$ for which $i \neq j$. When we join V_i with V_j we test the possibilities of connecting $V_i's$ root with all edges of V_j and choose the one connection which minimizes the increase of error.

7.4 Step 3

We iterate step 2 until we just have one tree left.

A Definitions

Definition of metric spaces 1. *A metric space* (X, d) *is a set of points in* X *with a real valued function* d *defined on* $X \times X$ *where the following four properties must be satisfied for all* $x, y, z \in X$
$(M1) \quad d(x, y) \geq 0$
$(M2) \quad d(x, y) = 0 \Leftrightarrow x = y$
$(M3) \quad d(x, y) = d(y, x)$
$(M4) \quad d(x, z) + d(z, y) \geq d(x, y)$
From now on xy *will denote the distance* $d(x, y)$ *in* X*.*

Definition of isometry 2. *An isometry is a relation (function) between two metric spaces. If* (X, d) *and* (X', d') *are metric spaces and there exists a function* $f : X \to X'$ *s.t.* $d(x, y) = d'(f(x), f(y))$ *for all* $x, y \in X$*, then we say that* f *is an isometry. If there exists an isometry* f*,* $f : X \to X'$*, then we say* (X, d) *is isometric to* (X', d')*.*

Definition of pointwise partial order 3. *A function* f *is defined to be smaller than a function* g *iff* $f(x) \leq g(x)$ *for all* $x \in X$ *and* $f(y) < g(y)$ *for some* $y \in X$*.*

Definition of partially ordered sets 4. *If* M *is a set, then a partial order on* M *is a relation* $(M, \leq)$*, which satisfies:*
Reflexivity *if* $x \in M \Rightarrow x \leq x$
Antisymmetry *if* $x, y \in M, x \leq y$ *and* $y \leq x \Rightarrow x = y$
Transitivity *if* $x, y, z \in M, x \leq y$ *and* $y \leq z \Rightarrow x \leq z$

Definition of minimal element 5. *if* $(M, \leq)$ *is a partially ordered set. Then* $m \in M$ *is a minimal element, if* $x \leq m \Rightarrow x = m$ *for each* $x \in M$*.*

References

[1] Andreas W. M Dress, *Trees, Tight Extensions of Metric Spaces, and the Cohomological Dimension of Certain Groups: A Note on Combinatorial Properties of Metric Spaces*, Advances in Mathematics, Vol 53, No. 4, Sep. 1984

[2] Alice Lesser, *Hereditarily Optimal Realizations*, Publications in Mathematics, Uppsala University, No. 45, Dec. 2004

[3] Henrik Baarnhielm, *Tychonoffs sats*, http://henrik.baarnhielm.net

[4] Copyright © 1997 - 2006 · Ewgenij Gawrilow and Michael Joswig · TU Berlin and TU Darmstadt, http://www.math.tu-berlin.de/polymake/

[5] A. Dress, K. T. Huber and V. Moulton, *An Explicit Computation of the Injective Hull of Certain Finite Metric Spaces in Terms of their Associated Buneman Complex*, Advances in Mathematics, University of Bielefeld and Mid Sweden University, No. 168, Sep. 2001

[6] Thomas H. Cormen, Charles E. Leiserson, Ronald L.Rivest and Clifford Stein, *Introduction to Algorithms*, The MIT Press, Second Edition, 2001

[7] http://en.wikipedia.org/wiki/Normal_distribution

[8] Gunnar Blom, *Sannolikhetsteori och statistikteori med tillämpningar*, Studentlitteratur, Fourth Edition, 1989