MO640/MC668

Guilherme P. Telles

IC-Unicamp

Avisado está

- Estes slides são incompletos.
- Estes slides contêm erros.

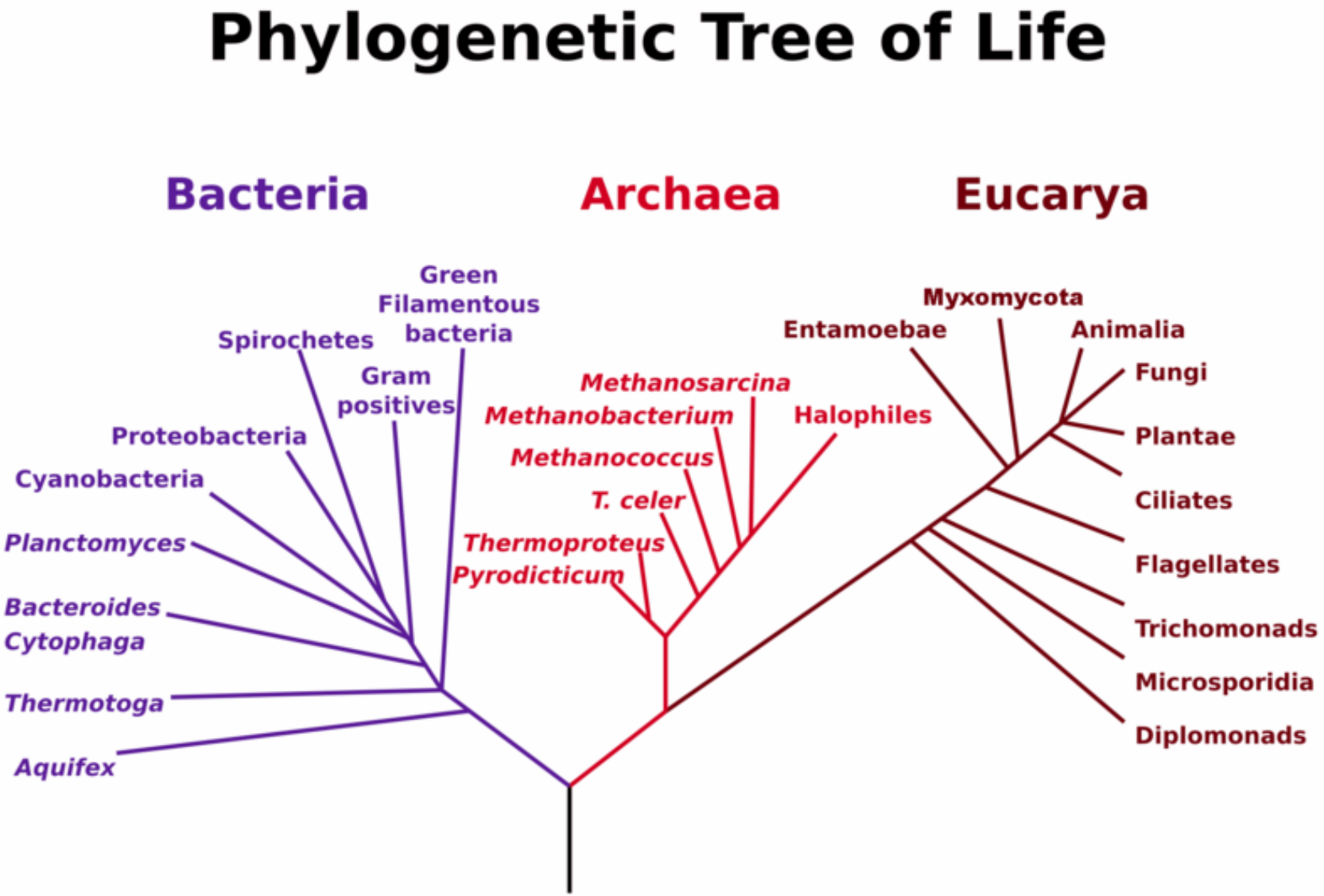
Parte I

Construção de árvores filogenéticas

Árvore filogenética ou filogenia

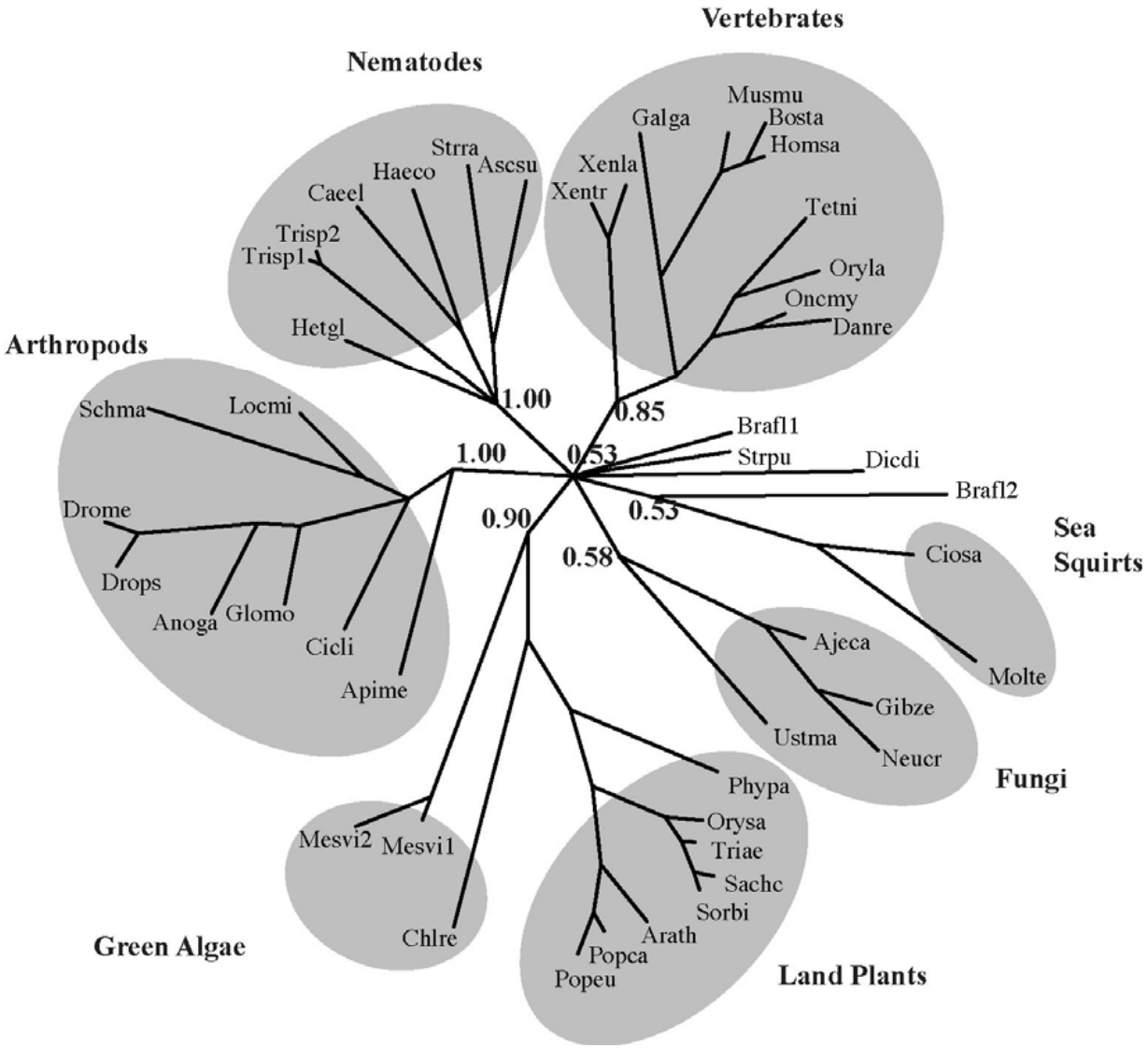
- Uma filogenia é uma tentativa de inferir a história evolucionária entre objetos com relação a ancestrais hipotéticos comuns.
- Os objetos podem ser espécies, gêneros, populações ou outras unidades taxonômicas.
- Uma *árvore filogenética* ou *filogenia* é uma árvore em que as folhas representam objetos que existem atualmente e em que os nós internos representam ancestrais hipotéticos.
- Uma filogenia é sempre uma hipótese:
  - Não temos dados suficientes sobre os ancestrais, que podem ter pouco significado biológico.
  - É difícil ter certeza com relação à ancestralidade entre os organismos.

Exemplo



[ref. desconhecida]

Exemplo



[Hayden e Jorgensen, BMC Biology,5:32, 2007]

Filogenia molecular

- A construção de filogenias é uma sub-área da taxonomia numérica.
- Filogenia molecular: moléculas também evoluem e podem ser usadas como unidades taxonômicas.

Aspectos de interesse

- Topologia: como os nós internos estão conectados entre si e com as folhas.
- Distância: pesos associados às arestas podem representar distância evolucionária entre os nós.
- Raiz: a árvore pode ser enraizada ou não. A raiz implica uma relação de ancestralidade entre os nós.

Número de árvores

- O número de árvores binárias enraizadas com n folhas distinguíveis e nós internos não-distinguíveis é

$$\prod_{i=0}^{n}(2i-3) = \Omega((2n/3)^{n-1}).$$

- O número de árvores binárias com n folhas distinguíveis e nós internos não-distinguíveis é

$$\prod_{i=0}^{n}(2i-5) = \Omega((2n/3)^{n-2}).$$

Dois tipos de entradas

- Matriz de características.
- Matriz de distâncias.

Matrizes de características

Matriz de características

- Uma *matriz de características* ou *matriz de estados de características* é uma matriz $n \times m$ em que cada um dos n objetos tem m características que podem assumir r estados distintos.

Exemplo

	pelos	vértebras	esqueleto ósseo	2 aberturas atrás dos olhos	4 membros	embrião amniótico
tubarões	n	s	n	n	n	n
actinopterígeos	n	s	s	n	n	n
crocodilianos	s	s	s	s	s	s
anfíbios	n	s	s	n	s	n
pássaros	s	s	s	s	s	s
primatas	s	s	s	n	s	s
roedores	s	s	s	n	s	s

[evolution.berkeley.edu/evolibrary/article/phylogenetics_07]

Suposições sobre as características

- As características são discretas: cada característica tem no máximo r estados que são denotados por inteiros entre 0 e $r-1$.
- As características devem poder ser herdadas independentemente.
- Para um grupo de objetos, todos os estados observados de uma certa característica evoluíram a partir de um "estado ancestral", que é o estado do ancestral comum mais próximo.
- As características atribuídas aos nós ancestrais podem descrever um objeto biológico sem significado.

Dificuldades

- Convergência ou evolução paralela: dois objetos com a mesma característica não são próximos evolutivamente. P. ex. asas em morcegos e em aves.
- Reversões: um estado ancestral modifica-se entre objetos.
- A princípio vamos considerar formulações do problema em que convergência e reversões não são admitidas.

Tipos de características

- Qualitativa: é uma característica em que o relacionamento entre os estados não é conhecido.
- Cladística: é uma característica qualitativa associada com uma árvore em que cada vértice é um estado da característica.
  - Ordenada: é uma característica cladística em que a árvore é enraizada.
  - Não-ordenada: é uma característica cladística em que a árvore não é enraizada.

Filogenia perfeita

- Uma filogenia T é uma *filogenia perfeita* se o conjunto de todos os nós para os quais o estado é s_i com respeito a c_j forma uma subárvore (subgrafo conexo) de T.

Problema da filogenia perfeita

- O problema da filogenia perfeita consiste em construir uma filogenia perfeita a partir de uma matriz de características.
- Se a filogenia é perfeita então uma transição de um estado ancestral para um estado derivado de uma característica pode ser associada com a aresta da árvore que conecta a subárvore que "contém" aquele estado.
- Se um conjunto de objetos admite uma filogenia perfeita dizemos que as características são *compatíveis*.

Dificuldade

- Características cladísticas: P.
- Características qualitativas: NP-completo.
- Características qualitativas mas número fixo de estados: P. Agarwala e Fernandez-Baca, 1994[1], em tempo $O(2^{3r}(nm^3 + m^4))$.
- Características qualitativas mas número fixo de características: P. McMorris, Warnow e Wimer, 1993[2], em tempo $O((rm)^{m+1} + nm^2)$.

[1] R. Agarwala e D. Fernandez-Baca. A polynomial-time algorithm for the perfect phylogeny problem when the number of character states is fixed. SIAM J. Computing, 23, p. 1216, 1994.

[2] F.R. McMorris, T. Warnow e T. Wimer. Triangulating vertex colored graphs. SODA 1993, p. 120, 1993.

Filogenia perfeita para dois estados

- Vamos supor que temos apenas dois estados para as características e que o estado 0 é ancestral e o 1 é derivado.
- Uma filogenia perfeita para uma matriz binária será tal que para cada característica em M existe uma única aresta na árvore que representa a transição do estado 0 para o estado 1.
- O estado de todas as características da raiz será 0.

Filogenia perfeita para dois estados

- Dada uma matriz M de características binárias, para cada coluna j de M sejam
  - O_j o conjunto de objetos cujo estado é 1 para j e
  - $\overline{O_j}$ o conjunto de objetos cujo estado é 0 para j.
- Teorema: Uma matriz de características binárias M admite uma filogenia perfeita se e somente se para cada par de características i e j os conjuntos O_i e O_j são disjuntos ou um deles contém o outro.

Algoritmo direto para verificação

- Basta comparar todos os pares de conjuntos entre si.
- Cada par de conjuntos pode ser comparado em tempo $O(n)$, todos os pares podem ser comparados em tempo $O(nm^2)$.

Algoritmo mais eficiente para verificação

- Um algoritmo mais eficiente primeiro ordena as colunas de M pelo número de uns.
- Depois constrói uma matriz auxiliar L para registrar, para cada valor 1, qual a coluna à esquerda mais próxima que contém um 1, ou -1 se não existir.
- Finalmente o algoritmo usa L para verificar se uma coluna que tem interseção com outra está contida totalmente nela.
- O custo é $O(nm)$.

Algoritmo

PERFECT-PHYLOGENY-DECISION(M)

```
1   Sort M by the non-increasing number of 1's in each column
2   L[n, m] = 0
3   for i = 1 to n
4         k = −1
5         for j = 1 to m
6               if M[i, j] == 1
7                     L[i, j] = k
8                     k = j
9   for each column j of L
10        if L[i, j] ≠ L[l, j] for some i,l and L[i, j], L[l, j] ≠ 0
11              return false
12  return true
```

Algoritmo de construção

- O algoritmo de construção adiciona um objeto de cada vez, construindo caminhos que representam os 1s na matriz.

Algoritmo

PERFECT-PHYLOGENY-CONSTRUCTION(M)

```
 1  Sort M by the non-increasing number of 1's in each column
 2  Create root
 3  for each object i
 4       p = root
 5       for j = 1 to m
 6            if M[i, j] == 1
 7                 if exists edge (p, u) labeled j
 8                      p = u
 9                 else
10                      Create node u
11                      Create edge (p, u) labeled j
12                      p = u;
13       Place object i in p
14  for each node u except root
15       Create as many leaves linked to u as there are objects in u
16  return root
```

Complexidade

- A ordenação pode ser feita em tempo $O(nm)$ usando radix sort.
- Cada elemento de M vai ser processado uma vez, a um custo constante.
- O tempo de execução total é $O(nm)$.

Filogenia perfeita para duas características

- Um grafo de interseções para uma coleção de conjuntos $\mathcal{C}$ é um grafo que tem um vértice para cada conjunto em $\mathcal{C}$ e que tem uma aresta entre dois conjuntos se os conjuntos têm interseção não-vazia.
- Seja O_u o conjunto dos objetos que têm um estado de uma caraterística u.
- O grafo de interseção de estados S tem
  - um vértice para cada estado de uma caraterística u e
  - uma aresta entre os vértices u e v de S se e somente se os conjuntos correspondentes possuem interseção não-vazia.

Filogenia perfeita duas características

- Teorema: Uma matriz de estados de características com duas características admite uma filogenia perfeita se e somente se seu grafo de interseção de estados é acíclico.
- O custo dessa verificação é $O(mr^2)$.

Filogenia perfeita para duas características

1. Construa o grafo de interseção de estados S para M.
2. Construa um grafo $G = (V, E)$ com
   - um vértice v para cada aresta (x, y) de S, representando o conjunto $O_v = O_x \cap O_y$ e associado com o conjunto de estados $E_v = \{x\} \cup \{y\}$.
   - uma aresta entre dois vértices u e v se $E_u \cap E_v \neq \emptyset$.
3. Se G não for conexo, adicione uma aresta entre pares de nós em componentes conexos distintos, conectando-os.
4. Construa uma árvore geradora T de G.
5. Para cada nó v adicione os objetos em O_v como folhas de v.

Características ordenadas

- Características ordenadas não-binárias podem ser fatoradas em características binárias.
- Seja c uma característica cladística e seja T_c a árvore para os estados da característica c.
- O *fator binário* de c em relação a v é a característica binária d cujo estado 1 consiste de todos os objetos s cujo estado na característica c é um nó da subárvore de T_c enraizada em v.
- Teorema: Um conjunto de características ordenadas $\mathcal{C}$ é compatível se e somente se o todos os fatores binários das características em $\mathcal{C}$ são compatíveis.
- A partir dos fatores binários podemos verificar se uma matriz de características ordenadas admite filogenia perfeita em tempo $O(nmr)$ e podemos construir a filogenia em tempo polinomial (tree popping).

Características binárias não-ordenadas

- Teorema: Seja $\mathcal{C}$ um conjunto de características binárias não-ordenadas. Seja $\mathcal{C}'$ o conjunto de características binárias ordenadas obtido de $\mathcal{C}$ enraizando cada característica pelo estado que tem mais objetos quebrando empates arbitrariamente. Então $\mathcal{C}$ é compatível se e somente se $\mathcal{C}'$ é compatível.
- Dada uma característica não-ordenada c é sempre possível escolher uma raiz para T_c tal que em todo fator binário da característica ordenada resultante o estado ancestral tenha pelo menos tantos objetos quanto o estado descendente.

Parcimônia e compatibilidade

- Filogenia perfeita raramente é admitida por matrizes de características:
  - erros experimentais.
  - violação da ausência de reversões e convergência.
- Parcimônia: minimizar a ocorrência de reversões e convergência. NP-completo.
- Compatibilidade: excluir o menor número de características para evitar reversões e convergência. NP-completo.

Matrizes de distâncias

Matriz de distâncias

- Uma *matriz de distâncias* para n objetos é uma matriz $n \times n$ de reais não-negativos.
- A célula $M[i, j]$ armazena a distância entre o objeto i e o objeto j.

Filogenia baseada em distâncias

- Dada uma matriz de distâncias M para n objetos, queremos construir uma árvore não-enraizada T tal que:
  - cada uma das n folhas corresponde a um objeto de M,
  - os nós internos têm grau 3,
  - todas as arestas na árvore têm um custo não-negativo,
  - o custo do caminho entre os objetos i e j é igual a $M[i, j]$.
- Se uma tal árvore pode ser construída, dizemos que M e T são aditivas.

Métrica

- Uma matriz M é métrica se
  - $M[i,j] = 0$ para $i = j$,
  - $M[i,j] > 0$ para $i \neq j$,
  - $M[i,j] = M[j,i]$ para todo para todo i, j e
  - $M[i,j] \leq M[i,k] + M[k,j]$ para todo i, j, k.

Condição dos 4 pontos

- Teorema: Quaisquer 4 vértices em uma árvore admitem uma rotulação x, y, x, t tal que

$$d(x, y) + d(z, t) = d(x, z) + d(y, t) \geq d(x, t) + d(y, z).$$

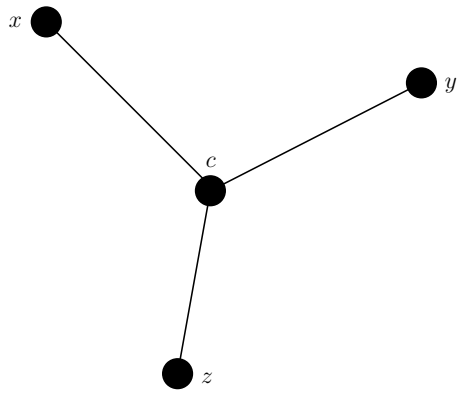
- Essa condição é chamada de *condição dos quatro pontos (C4P)*.
- Fazendo $z = t$ temos a desigualdade triangular como caso particular.

Matriz aditiva

- Teorema: Uma matriz de distâncias M é aditiva se e somente se é métrica e para qualquer subconjunto de quatro objetos existe uma rotulação x, y, z e t que satisfaz à C4P.

Algoritmo para matrizes aditivas

- Se temos apenas dois objetos x e y então uma árvore com uma aresta de peso $M[x, y]$ ligando x e y é aditiva.
- A árvore resultante da adição de um terceiro objeto tem a seguinte topologia:



Algoritmo para matrizes aditivas

- O custo das arestas pode ser calculado da seguinte forma. Seja d_{ij} o custo da aresta (i, j).
  Da árvore,

$$M[x,z] = d_{xc} + d_{cz} \quad (1)$$
$$M[y,z] = d_{yc} + d_{cz} \quad (2)$$
$$d_{yc} = M[x,y] - d_{xc} \quad (3)$$

Subtraindo (2) de (1) e substituindo (3) obtemos:

$$d[x,c] = \frac{M[x,y] + M[x,z] - M[y,z]}{2} \quad (4)$$

Algoritmo para matrizes aditivas

- Podemos proceder de maneira similar e obter

$$d[y,c] = \frac{M[x,y] + M[y,z] - M[x,z]}{2} \tag{5}$$

$$d[z,c] = \frac{M[x,z] + M[y,z] - M[x,y]}{2} \tag{6}$$

Algoritmo para matrizes aditivas

- Vamos adicionar w a uma árvore com três folhas. Vamos supor que o nó interno d vai ser adicionado à árvore. Escolhemos dois nós já na árvore, digamos x e y, e aplicamos as equações (4), (5) e (6) com z substituído por w.
- Se d coincidir com o nó c, isso significa que é a aresta (z, c) que deve ser dividida. Então aplicamos as equações (4), (5) e (6) usando z no lugar de x ou de y.
- Senão d deve dividir a aresta (x, c) ou a aresta (y, c), de acordo com os pesos dados pelas equações.

Algoritmo para matrizes aditivas

- Generalizando a idéia anterior, suponha que temos uma árvore aditiva com k objetos. Para adicionar o objeto $k+1$:
  1. Selecione dois objetos x e y quaisquer na árvore e calcule a posição do novo nó interno c.
  2. Se após esse cálculo a posição de c não coincide com a posição de nenhum outro nó interno na árvore, adicionamos $k+1$ e c.
  3. Senão a posição de c coincide com a de algum nó u na árvore. Então escolhemos um objeto r que pertence à (outra) subárvore pendurada em u e verificamos o caso para três objetos. Repetimos esse processo até que a posição correta da divisão seja encontrada.

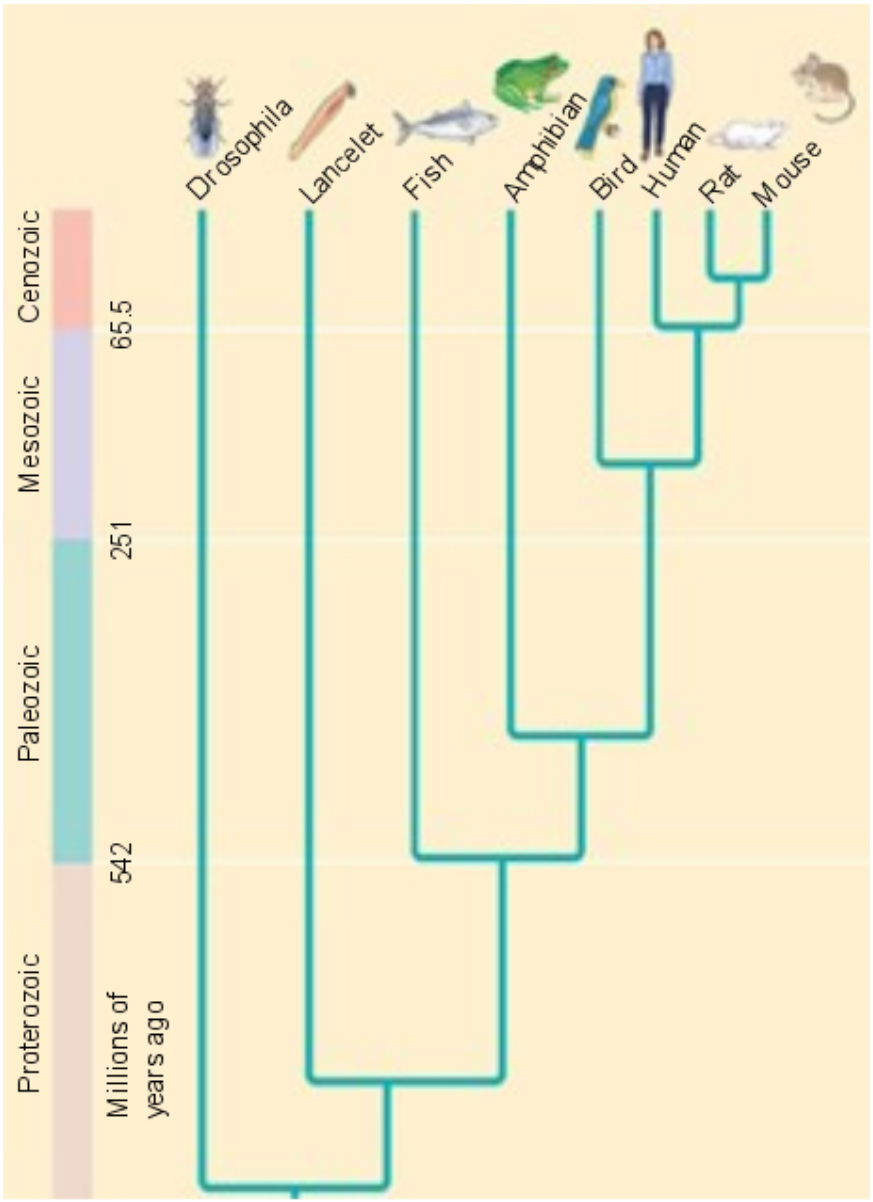
Matrizes não-aditivas

- É comum que matrizes de distâncias não sejam aditivas.
  - erros experimentais,
  - mutações sucessivas, reversões, convergências.
- O problema de aproximar a aditividade por várias medidas é NP-completo.

Árvores ultramétricas

- Uma matriz métrica $n \times n$ M é ultramétrica se e somente se $M[i,j] \leq \max\{M[i,k], M[j,k]\}$ para todo i, j, k.
- Um árvore aditiva é *ultramétrica* se puder ser enraizada de forma que todos os caminhos da raiz a uma folha tenham o mesmo tamanho.
- Uma matriz M é ultramétrica se e somente se existe T tal que $d_T(i,j) = M[i,j]$.
- Uma árvore ultramétrica está relacionada com a noção de evolução a uma taxa constante a partir de um ancestral comum.

Exemplo



[Campbel and Reece, 2005]

Árvores ultramétricas

- Na prática, as árvores quase nunca são ultramétricas.
- Mas se a incerteza sobre as distâncias puder ser quantificada por intervalos que definem um limite inferior e um limite superior para distância real entre dois objetos, podemos resolver o problema em em tempo polinomial.

Árvore ultramétrica sanduíche

- Temos então duas matrizes de distâncias, M_l e M_h com os limites inferiores e superiores para distâncias reais.

M_l

	B	C	D	E
A	3	2	4	3
B		4	1	1
C			3	3
D				1

M_h

	B	C	D	E
A	7	3	6	5
B		10	4	8
C			8	5
D				7

Árvore ultramétrica sanduíche

- O problema é reconstruir uma árvore ultramétrica tal que para todo par de objetos i, j

$$M_l[i,j] \leq d_T(i,j) \leq M_h[i,j].$$

Árvore ultramétrica sanduíche

- Dada uma matriz de distâncias $n \times n$ M podemos associar a ela um grafo completo com pesos com n vértices, onde $w(i, j) = M[i, j]$.
- Sejam G_l e G_h os grafos que correspondem a M_l e M_h.
- Seja T uma árvore geradora mínima para o grafo G_h.

Árvore ultramétrica sanduíche

- Dados dois nós u e v em T, a aresta de maior peso no caminho de u a v é chamada de *link* e é denotada $(u, v)_{max}$.
- Dada uma aresta e de T, ela é o link de pelo menos um par de vértices (os extremos dela). Pode ser de mais de um par.
- O peso-de-corte de uma aresta e de T é

$$cw(e) = \max\{M_l[a, b] \mid e = (a, b)_{max}\}.$$

- Então $cw(e)$ é a maior entrada em M_l entre os pares que têm e como link.

Uma tentativa de intuição

1. Seja (a, b) o par de objetos mais distantes. Seja $d(a, b)$ a distância entre eles.
2. $d(a, b)$ deve ser maior que a maior distância em M_l ($M_l^{\max}$).
3. Mas $d(a, b)$ não precisa ser maior que $M_l^{\max}$.
4. Então deveríamos ter $d_T(a, b) = M_l^{\max}$ e a e b deveriam estar em subárvores diferentes da raiz.

Algoritmo

1. Construa uma árvore geradora mínima T para G_h e para cada aresta $e \in T$ compute $cw(e)$.
2. Ordene as arestas em ordem decrescente de cw, $e_1, e_2, \ldots, e_{n-1}$.
3. Construa uma árvore ultramétrica da seguinte forma. Crie um novo nó r raiz de uma árvore U. Construa as árvores U_1 e U_2 recursivamente para as duas componentes resultantes da remoção de e_1. Faça $H(U) = cw(e_1)/2$ e conecte r com T_i por uma aresta de peso $H(U) - H(U_i)$. Se U_i for unitária, $H(U_i) = 0$.

Correção

- A correção do algoritmo está baseada em parte nos seguintes resultados.
- Uma matriz aditiva M é ultramétrica se e somente se no grafo completo com pesos G correspondente a M a aresta de maior peso de qualquer ciclo não é única.
- Existe uma árvore ultramétrica que se encaixa em duas matrizes M_l e M_h se e somente se $M_l[a, b] \leq w((a, b)_{max})$ para todos os pares de objetos a e b.

Implementação eficiente

- A árvore geradora mínima pode ser computada usando o algoritmo de Kruskal ou o de Prim.
- Para calcular cw podemos usar estruturas para conjuntos disjuntos para construir uma árvore auxiliar R que permite encontrar o link entre dois objetos rapidamente usando consultas de *lca*.
- Consultas de ancestral comum mais baixo podem ser feitas em tempo constante usando uma estrutura de dados que pode ser construída em tempo linear.
- Depois a filogenia é construída iterativamente, resultando em um algoritmo $O(n^2)$.

Construção de R

BUILD_R(T)

```
 1  for each object i
 2      MAKESET(i)
 3      Create one-node tree for i
 4  Sort edges of T in nondecreasing order of weights
 5  for each edge e = (a, b) ∈ T in this order
 6      A = FINDSET(a)
 7      B = FINDSET(b)
 8      if A ≠ B
 9          r_a = tree that contains a
10          r_b = tree that contains b
11          create tree R
12          R.edge = e
13          R.leftChild = r_a
14          R.rightChild = r_b
15          UNION(A, B)
16  return R
```

Computação de cw

$\mathrm{CW}(M_l, T, R)$

1 Pre-process R for lowest common ancestor (lca) queries
2 **for** each edge $e \in T$
3 $\quad CW[e] = 0$
4 **for** each pair of objects a and b
5 $\quad e = lca(R, a, b)$
6 $\quad$ **if** $M^l[a, b] > CW[e]$
7 $\quad\quad CW[e] = M^l[a, b]$
8 **return** CW

Construção de U

BUILD_U(M_l, T, R)

```
 1  for each object i
 2      MAKESET(i)
 3      Create one-node tree for i
 4      height[i] = 0
 5  Sort edges of T in nondecreasing order of cut-weights
 6  for each edge e = (a, b) ∈ T in this order
 7      A = FINDSET(a)
 8      B = FINDSET(b)
 9      if A ≠ B
10          u_a = tree that contains a
11          u_b = tree that contains b
12          create tree U
13          U.leftChild = u_a
14          U.rightChild = u_b
15          height[U] = CW[e]/2
16          w(u_a, U) = height[U] − height[u_a]
17          w(u_b, U) = height[U] − height[u_b]
18          UNION(A, B)
19  return U
```

Árvore aditiva sanduíche

- É NP-completo.

Compatibilidade de filogenias

Árvores consistentes

- Seja O um conjunto de objetos e seja $O' \subseteq O$.
- Sejam T e T' as filogenias para O e O' respectivamente.
- Seja $T|O'$ a árvore obtida pela supressão de nós de grau 2 da subárvore mínima de T que conecta os nós em O'.
- Então T é consistente com T' se T' puder ser obtida de $T|S'$ por contrações de arestas.

Problema da compatibilidade

- Um conjunto $\mathcal{T}$ de filogenias em subconjuntos de O é compatível se existe uma filogenia que é consistente como toda árvore em $\mathcal{T}$.
- O problema da compatibilidade de árvores é determinar se uma coleção $\mathcal{T}$ de filogenias em subconjuntos de O é compatível.

Dificuldade

- Um quarteto é uma filogenia para 4 objetos onde todos os nós internos têm grau 3.
- O problema da compatibilidade não-enraizada é NP-completo mesmo quando a entrada consiste de quartetos[3].
- Mas quando as árvores têm pelo menos um objeto em comum o problema é polinomial.
- O problema se torna um caso de compatibilidade enraizada.

[3] M.A. Steel. The complexity of reconstructing trees from qualitative characters and subtrees. J. of Classification. v. 9, p. 91, 1992.

Compatibilidade enraizada

- Seja T uma filogenia para um conjunto de objetos O.
- Seja $T|O'$ a árvore obtida pela supressão de nós de grau 2 da subárvore mínima de T que conecta os nós em O' sendo que a raiz da subárvore que conecta O' só é suprimida se tiver apenas um filho.
- Para um conjunto $O' \subseteq O$, $\mathcal{T}|O'$ denota o conjunto $\{T|O' : T \in \mathcal{T}\}$.
- Seja $G_{\mathcal{T},O}$ o grafo com um vétice para cada objeto e uma aresta (u, v) se e somente se existe uma árvore $T \in \mathcal{T}$ tal que u e v estão no mesmo componente conexo de $T \setminus \{T.raiz\}$.

Filogenia enraizada

- A idéia do algoritmo é que se existir uma árvore T^* que é consistente com cada árvore em $\mathcal{T}$ então o conjunto representado por cada componente conexo de $G_{\mathcal{T},O}$ está contido em uma subárvore distinta da raiz de T^*.
- Isso fornece o primeiro nível de T e continuamos recursivamente nos componentes.

Algoritmo[4]

ROOTED-TREE-COMPATIBILITY$(\mathcal{T}, O)$

1 let T be a tree with a single vertex v
2 **if** $G(\mathcal{T}, O)$ has more than one connected component
3 **for** each connected component X of $G(\mathcal{T}, O)$
4 $T_X =$ ROOTED-TREE-COMPATIBILITY$(\mathcal{T}|X, X)$
5 make T_X a subtree of v in T
6 **return** T

- As árvores em $\mathcal{T}$ são compatíveis com a árvore T retornada pelo algoritmo se e somente se para todo $v \in O$ existir um nó rotulado v em T.

[4] M.A. Steel. The complexity of reconstructing trees from qualitative characters and subtrees. J. of Classification. v. 9, p. 91, 1992.

O caso não enraizado

- O algoritmo pode ser usado para resolver o caso não-enraizado se considerarmos todos os enraizamentos possíveis para as árvores em $\mathcal{T}$.
- Mas isso levaria tempo exponencial.
- Mas quando todas as árvores têm algum objeto em comum v podemos enraizar toda árvore T em um vizinho de v em T e usar o algoritmo anterior.