Matheurísticas para o problema da filogenia viva com politomia

Maria Elisa Rodrigues Rabello

GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO DA FACOM-UFMS

Data da Defesa:

Assinatura:_____

Matheurísticas para o problema da filogenia viva com politomia

Maria Elisa Rodrigues Rabello

Orientador: *Edna Ayako Hoshino*

Monografia apresentada como requisito parcial para aprovação na Componente Curricular Não-Disciplinar Trabalho de Conclusão de Curso do curso de Graduação em Ciência da Computação, da Universidade Federal de Mato Grosso do Sul.

UFMS - Campo Grande
Dezembro/2025

Resumo

Uma árvore de filogenia mostra a relação evolutiva entre seres vivos ou objetos, como documentos compartilhados online. Uma árvore filogenética viva com politomia é uma árvore em que seus nós internos podem incluir objetos vivos e um interno pode gerar dois ou mais descendentes. As árvores podem ser obtidas através de uma matriz de distância. O objetivo desse trabalho é encontrar um algoritmo, utilizando a programação linear inteira e matheurísticas, que dada uma matriz de distância construa uma árvore de filogenia viva com politomia em que a distância entre os objetos na árvore seja o mais próxima possível da distância entre eles na matriz. Para a resolução do problema foram propostas e implementadas matheurísticas baseadas no método *relax-and-fix*, usando-se as bibliotecas de otimização SCIP e Cplex, e incorporadas como heurísticas em um algoritmo branch-and-bound. Foram propostos três critérios para partição das variáveis: sequencial (armazena as variáveis nas partes do conjunto de partições na sequência do conjunto ordenado das próprias variáveis), aleatória (armazena as variáveis aleatoriamente nas partes do conjunto de partições) e PAI (armazena as variáveis relacionadas as arestas que chegam nos vértices primeiro, uma parte do conjunto de partições para cada vértice), que prioriza as variáveis relacionadas com as arestas que chegam em um vértice. Os resultados obtidos com as matheurísticas propostas não alcançaram valores de limitantes bons e não acharam soluções viáveis em instâncias com mais de 10 objetos. No entanto, os testes realizados mostram que a matheurística sequencial fornece os melhores limitantes quando comparado com as demais.

Sumário

CAPÍTULO

1

Introdução

Neste capítulo será definido o que é uma árvore filogenética, a diferença de uma árvore tradicional com uma árvore viva e com politomia, justificativa e o objetivo desse trabalho.

1.1 Filogenia e árvore filogenética

A filogenia foi definida pela primeira vez pelo entomólogo alemão Willi Hennig em 1950. No seu livro Filogenia Sistemática, Hennig descreve a filogenia como um método de análise da evolução das espécies que permite a construção de relações de parentesco entre elas (Hennig, 1999). A partir de análises filogenéticas é possível construir um grafo que relaciona as espécies de acordo com seu parentesco. Esse grafo é chamado de árvore filogenética. As folhas dessa árvore são as espécies existentes, os nós internos representam espécies hipotéticas que deram origem a outras até a espécie representada nas folhas, e cada espécie possui apenas dois descendentes (Lenzini e Marianelli, 1997). Na Figura 1.1 publicada por Stiller et al. (2024) representa a relação evolutiva entre as espécies de aves modernas.

1.1.1 Características de uma árvore filogenética

As árvores filogenéticas possuem algumas características que serão apresentadas nesta sessão. Elas pode ser enraizadas ou não, vivas ou não, finalmente bifurcadas ou com politomia e finalmente podem ser baseadas em distância ou em características.

Uma árvore filogenética é uma árvore enraizada quando há um ancestral

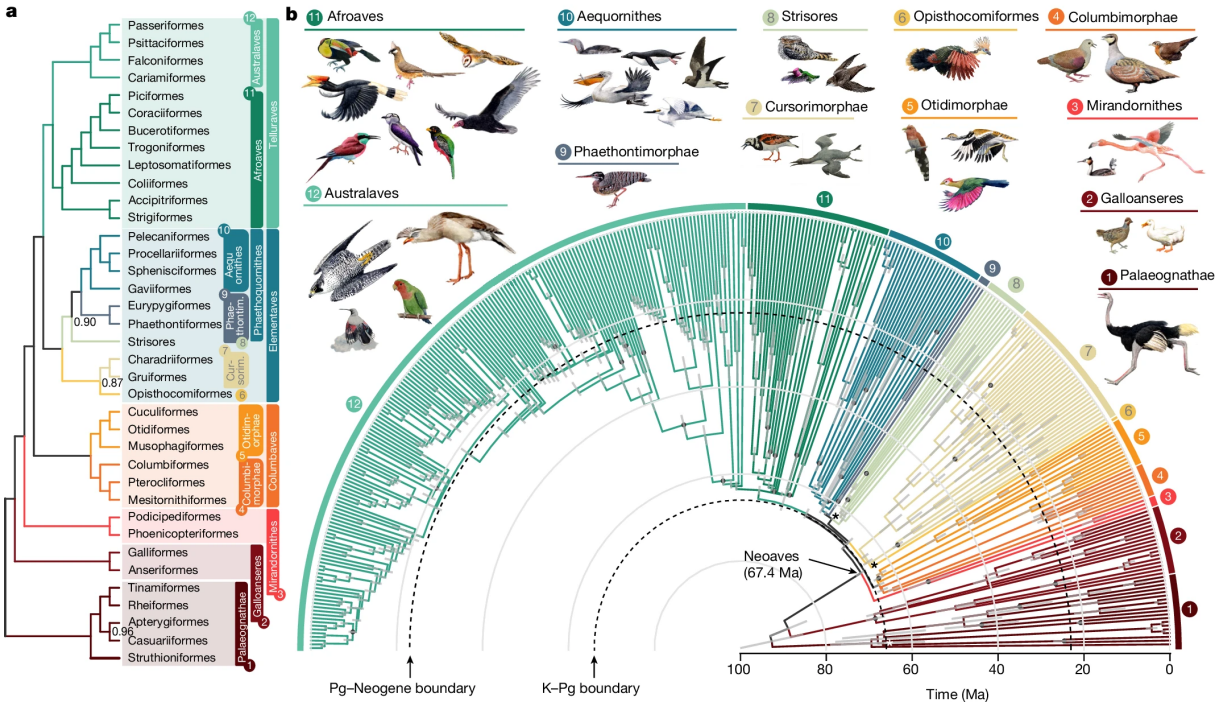


Figura 1.1: Árvore da vida das aves.

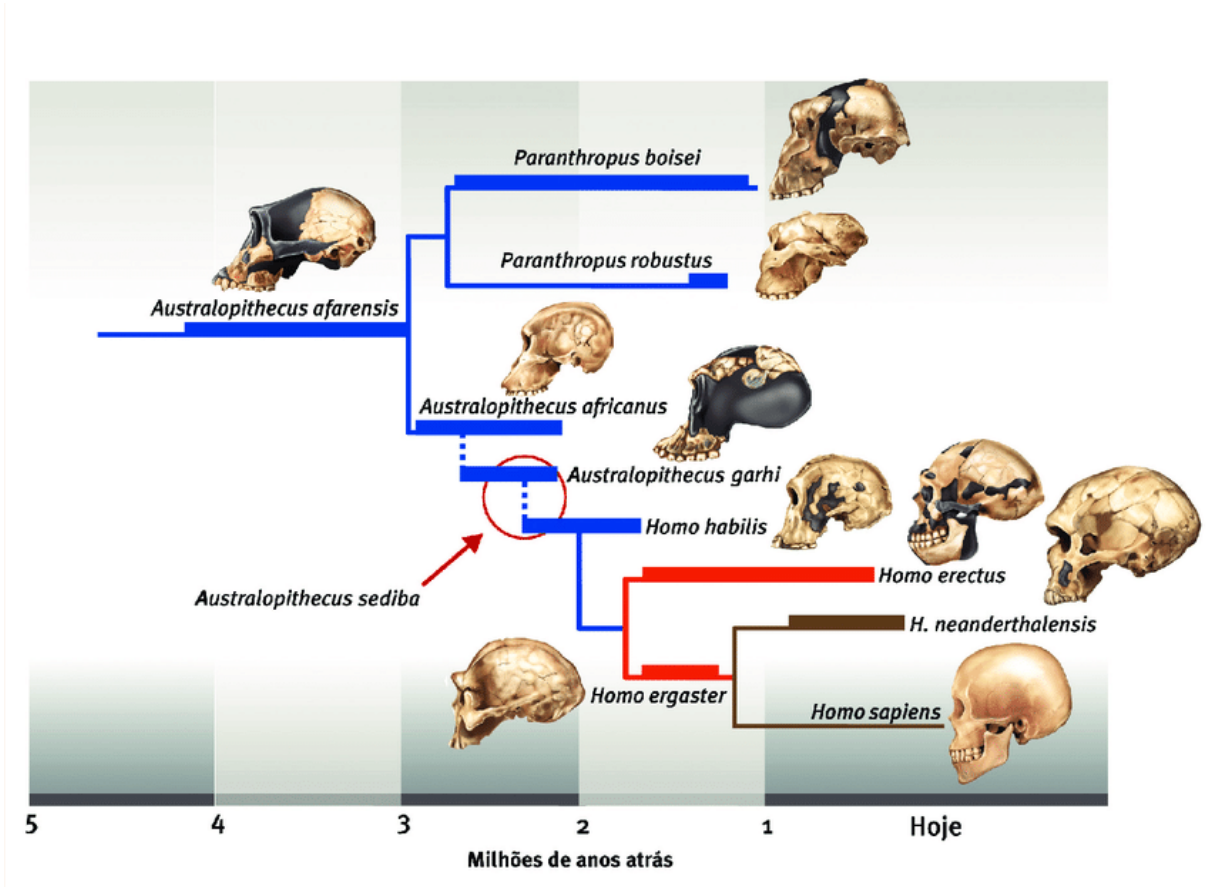


Figura 1.2: Árvore evolutiva das espécies conhecidas de hominídeos.

comum a todas as espécies em estudo. São árvores que apresentam uma história de ancestralidade e parentesco entre as espécies nas folhas da árvore.

A Figura 1.2 é uma árvore enraizada, em que a espécie *Australopithecus afarensis* é a raiz, ou seja, é o ancestral comum a todas as demais espécies. Já as árvores sem raiz, não demonstram uma relação de ancestralidade, mas de distância evolutiva, como representado pela Figura 1.3.

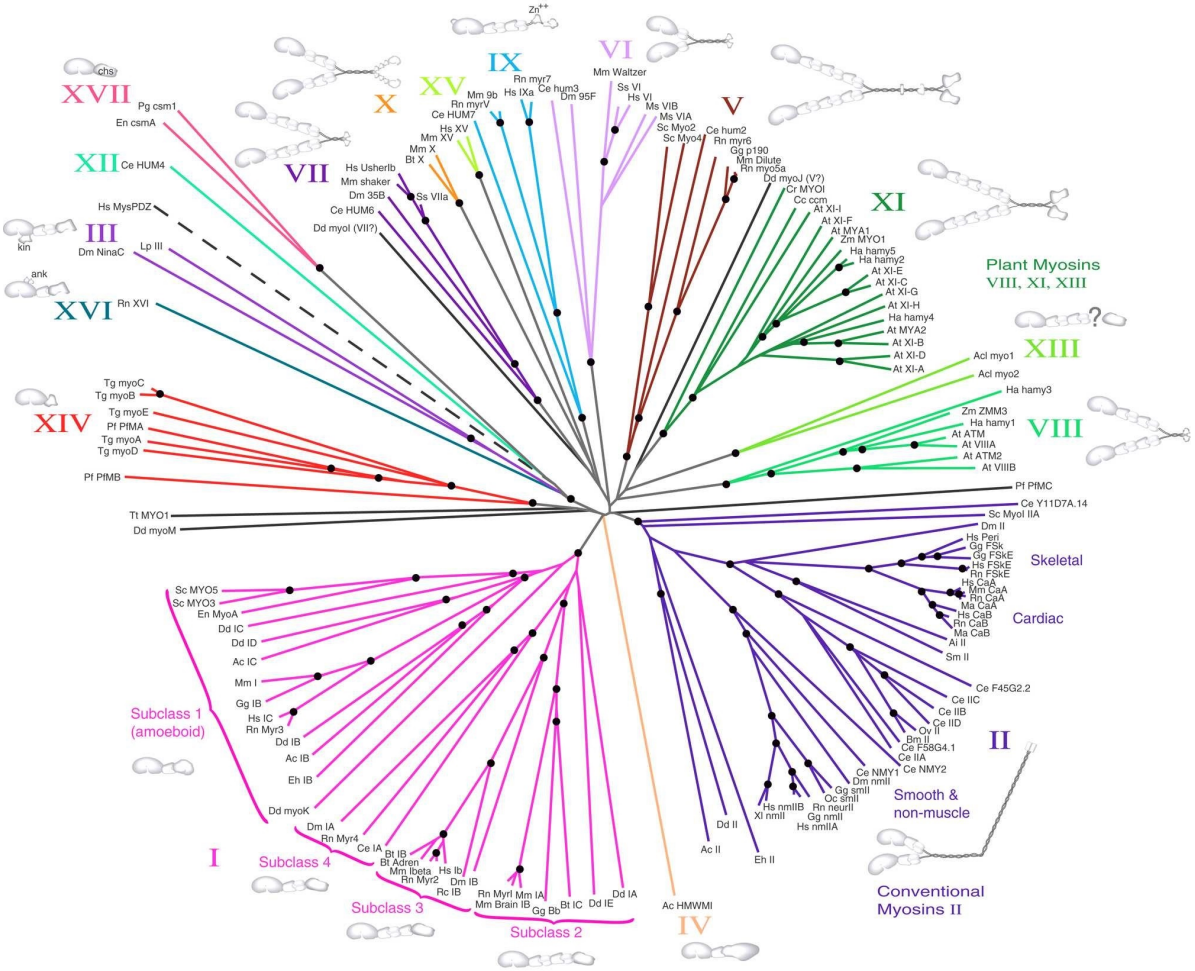


Figura 1.3: Árvore sem raiz da Miosina, proteína responsável pela contração muscular (Hodge e Cope, 2000).

Um tipo especifico de árvore filogenética é a árvore que possui em seus nós internos espécies vivas, ou seja, espécies que coexistem com seus descendentes. São chamadas de árvores filogenéticas vivas e descrevem a relação de espécies que evoluem rapidamente, como é o caso de vírus e bactérias. Também é possível usar essas árvores para analisar a evolução de objetos não biológicos, como documentos que circulam na internet e em áreas como a linguística e a paleontologia (Telles et al., 2013). A Figura 1.2 é um exemplo de árvore viva da paleontologia que representa a história evolutiva dos hominídeos. Repare que em ramos internos da árvore há espécies conhecidas que fazem parte do grupo de espécies em estudo (*Australopithecus sediba* e *Homo ergaster*).

Outra característica que uma árvore filogenética pode apresentar é a politomia. As árvores com politomia permitem que as espécies internas possuam

dois ou mais descendentes (Davidson e Sullivant, 2014). A politomia em árvores filogenéticas é usada para representar o evento evolutivo de uma espécie dando origem a mais de duas espécies (Figura 1.4), ou quando as informações disponíveis para gerar a árvore não fornece dados suficientes para afirmar que as espécies descendem de ancestrais diferentes (Figura 1.5) (University of California Museum of Paleontology).

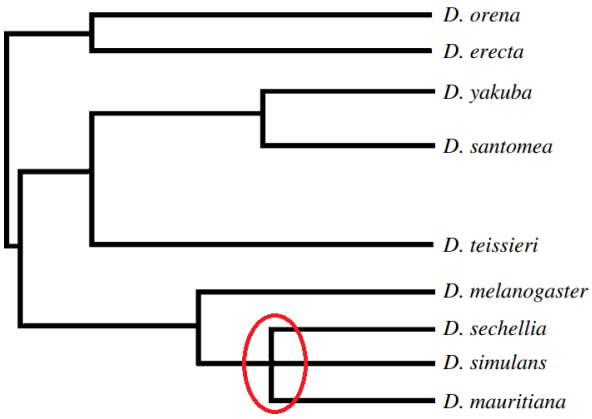


Figura 1.4: Árvore filogenética de moscas do gênero *Drosophila* em que três espécies possuem o mesmo ancestral (Coyne et al., 2004). O círculo vermelho destaca o ramo com politomia das árvores. Destaque inserido pela autora.

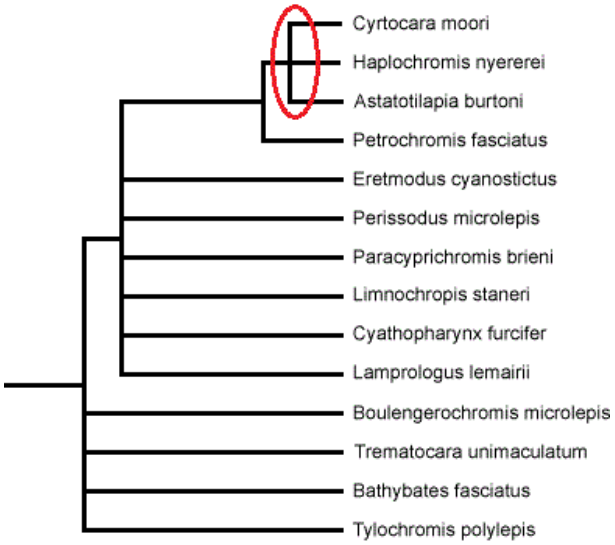


Figura 1.5: Árvore filogenética de Ciclídeos do Lago Malawi, espécies de peixes que evoluíram muito rápido e não há material suficiente para identificar os pontos de origem de algumas espécies (University of California Museum of Paleontology). O círculo vermelho destaca o ramo com politomia das árvores. Destaque inserido pela autora.

Por fim, uma das maneiras de construir árvores filogenéticas através de dois tipos de matriz: de características ou de distância. As colunas da matriz de características representam certas características e cada linha representa

as espécies de interesse. Os valores no interior da matriz podem ser binário (espécie possui ou não possui a característica) ou reais (quantidade da característica presente na espécie). Já matriz de distância apresenta a que distância evolutiva uma espécie está de outra.

	Vértebras?	Esqueleto ósseo?	Quatro membros?	Ovo amniótico?	Cabelo?	Duas fenestras pós-orbitais?
Tubarões e parentes	SIM	não	não	não	não	não
Peixes com nadadeiras raiadas	SIM	SIM	não	não	não	não
Anfíbios	SIM	SIM	SIM	não	não	não
Primatas	SIM	SIM	SIM	SIM	SIM	não
Roedores e coelhos	SIM	SIM	SIM	SIM	SIM	não
Crocodilos e parentes	SIM	SIM	SIM	SIM	não	SIM
Dinossauros e pássaros	SIM	SIM	SIM	SIM	não	SIM

Figura 1.6: Exemplo de uma matriz de característica (tradução livre)(University of California Museum of Paleontology).

	A	B	C	D	E
A	0				
B	65	0			
C	88	56	0		
D	132	78	59	0	
E	45	34	90	108	0

Figura 1.7: Exemplo hipotético de uma matriz de distância.

O problema de transformar essas matrizes em árvores filogenéticas é chamado de problema da filogenia baseada em características (para a matriz de características) e baseada em distância (para a matriz de distâncias). Quando trata da filogenia viva então é chamado de problema da filogenia viva. E quando a politomia pode estar presente, então é o problema da filogenia com politomia. Este trabalho irá abordar o problema da filogenia viva com politomia baseada em distância.

1.2 Motivação

O problema da filogenia, que consiste em, dada uma entrada com espécies achar uma árvore filogenética correspondente, é um problema NP-difícil (Lenzini e Marianelli, 1997). Dada a importância do estudo evolutivo das espécies e demais aplicações (como estudos de objetos que circulam online e a linguística) ter um algoritmo eficiente capaz de achar uma árvore filogenética que represente o conjunto de espécies a serem estudadas traria uma grande contribuição a pesquisa.

O problema da árvore filogenética viva é recente. Foi destrico formalmente pela primeira vez em 2013 por Telles et al. (2013) e provado ser NP-difícil

por Araújo et al. em 2017. Ainda não existe um algorítimo eficiente para resolvê-lo e poucas heurísticas foram desenvolvidas até o momento. O mesmo é válido para o problema da filogenia viva com politomia. Como uma árvore filogenética com politomia (viva ou não) é isomórfica a árvore sem politomia caso as duas contenham o mesmo conjunto de espécies, podendo-se converter uma árvore com politomia em uma árvore sem politomia e vice-versa com o mesmo número de transformações (Robinson e Foulds, 1981), o problema da filogenia viva com e sem politomia possuem a mesma dificuldade de resolução. Embora a politomia no problema da filogenia não seja nova, na filogenia viva não há muitos trabalhos que a apliquem.

1.3 Objetivos

O objetivo deste trabalho é encontrar uma matheurística, utilizando programação linear inteira, que, dada uma matriz de distância entre objetos, gere uma árvore filogenética em que as distâncias entre os nós dessa árvore sejam o mais próximas possíveis das distâncias da matriz de entrada para o problema da filogenia viva com politomia.

A partir deste ponto este trabalho irá tratar apenas do problema da filogenia viva com politomia, a menos que de outra forma mencionado.

CAPÍTULO

2

Trabalhos Relacionados e Descrição do Problema

Este capítulo mostrará os trabalhos que serviram de inspiração e referência para este e descreverá o problema da filogenia viva com politomia formalmente.

2.1 Trabalhos relacionados

Esse trabalho busca inspiração nos trabalhos de Araújo et al. (2017), Papamichail et al. (2017) e Hoshino et al. (2018).

Araújo et al. (2017) prova que o problema da filogenia viva (sem politomia) é NP-difícil e propôs um algoritmo baseado na heurística Neighbor-Joining, chamado de Live-Neighbor-Joining, para resolver o problema utilizando como entrada matriz de distância.

Papamichail et al. (2017) propôs um algoritmo para resolver o problema da filogenia viva com politomia utilizando como entrada uma matriz de características. Utilizando dois algoritmos de *branch-and-bound* (*Mixed Tree Enumeration Algorithm* (MTEA) e *Cubic tree enumeration and edge contraction algorithm* (CTEECA)) é possivél encontrar árvores de máxima parcimônia (árvores filogenéticas com o menor número possível de mudanças evolutivas), sendo essas árvores filogenéticas vivas e com politomia.

Hoshino et al. (2018) propôs três modelos de propagação linear inteira com algoritmos *branch-and-bound* para resolver o problema da filogenia viva (sem politomia) de forma exata, e também uma heurística para melhorar o desempenho dos algoritmos. Este é o trabalho em que esta monografia se baseia

primariamente, adaptando o terceiro modelo proposto pelas autoras para o problema da filogenia viva com politomia., já que esse modelo foi o teve melhor desempenho dos três propostos.

2.2 Descrição formal do problema

2.2.1 Premissas

Para fins de generalização a partir deste ponto espécies serão chamadas de objetos. Este trabalho irá abordar o problema da filogenia viva com politomia baseado em distância. Ou seja, a matriz de entrada será uma matriz de distância, em seus nós internos poderá aparecer objetos vivos (que estão presentes na matriz de distância e além disso seus nós internos também poderão dar origem a dois ou mais objetos. A árvore gerada será uma árvore filogenética viva com politomia sem raiz.

2.2.2 O problema da filogenia baseada em distância

O problema da filogenia viva baseada em distância é uma generalização do problema clássico da filogenia baseada em distância. Para descrevê-lo, vamos começar com a descrição do problema clássico da filogenia nesta subseção e descrever as alterações necessárias nas subseções seguintes para adaptá-lo ao problema da filogenia viva e viva com politomia.

O problema da filogenia baseada em distância busca encontrar uma árvore filogenética sem raiz a partir de uma matriz de distância evolutiva entre objetos já conhecida. Uma árvore filogenética T é uma árvore que representa a história evolutiva de um conjunto S de objetos. A árvore T é uma árvore (V,A) onde V é o conjunto de vértices que representa os objetos do conjunto S e os objetos hipotéticos, que obrigatoriamente terá grau 3 e serão vértices internos da árvore) e A é o conjunto de arestas, que dado dois vértices x e y ligados por uma aresta $(x,y) \in A$ representa que x é mais próximo evolutivamente de y, sendo obrigatoriamente x ou y um vértice interno ou dois vértices internos, já que a filogenia clássica não permite que vértices interno pertençam ao conjunto S, e portanto dois objetos de S não podem ser ligados por uma aresta.

A distância entre objetos i e j do conjunto S é um número real não negativo que representa a distância genética entre os dois objetos. A distância entre os n objetos do conjunto S é dada por uma matriz de distância M, triangular e com dimensões n x n, onde M_{ij} é a distância entre os objetos i e j. Através dessa matriz a árvore T ponderada será construída de forma que suas folhas representam os objetos em S e que somado os pesos das arestas que conectam dois objetos i e j o resultado será o valor M_{ij} (Setubal e Meidanis, 1997).

Quando a matriz M é aditiva (para quaisquer 4 objetos i, j, k e l presente na matriz as equações $M_{ij} + M_{kl} = M_{ik} + M_{jl} \geq M_{il} + M_{jk}$) existe um algoritmo que resolve o problema da filogenia baseada em distância em tempo polinomial (Telles et al., 2013). No entanto, a maioria das matrizes de distâncias não são aditivas e portanto a solução é uma minimização da diferença das distâncias entre as folhas das árvores e os valores M_{ij} da matriz, tornando o problema da filogenia baseada em distância NP-difícil (Setubal e Meidanis, 1997; Lenzini e Marianelli, 1997).

2.2.3 O problema da filogenia viva

Descrita por Telles et al. (2013), a árvore filogenética do problema da filogenia viva pode possuir em seus nós internos objetos do conjunto S de entrada, ou seja, objetos vivos podem ser ancestrais de outros objetos vivos. Agora, uma aresta (x, y) que liga dois vértices x e y pode conectar dois vértices x e y internos, x ou y pertencendo ao conjunto S e os dois vértices sendo objetos do conjunto S. Telles et al. (2013) descreve formalmente a árvore filogenética viva baseada em distância conforme segue:

Seja M^n uma matriz quadrada de ordem n, e S o conjunto de n objetos representados na matriz M, em que $M^n_{ij} \in \mathbb{R}$ é a distância entre os objetos i e j. E seja T^n uma árvore ponderada, sem raiz. T^n é uma árvore viva para M^n se T^n:

- Cada folha em T^n representa um objeto de S,
- cada objeto em S aparece apenas uma vez em T^n
- $d^n_{ij} = M^n_{ij}$, $1 \leq i, j \leq n$, onde d^n_{xy} é a distância entre x e y em T^n.

Um vértice interno será vivo se representa um objeto em S.

E seguindo o comportamento da filogenia, quando a matriz de distância é aditiva para o problema da filogenia viva existe algoritmo que resolve o problema em tempo polinomial, mas quando a matriz não é aditiva então o problema da filogenia viva é NP-difícil provado por Araújo et al. (2017).

2.2.4 O problema da filogenia viva com politomia baseada em distância

Robinson e Foulds (1981) demonstra que as árvores filogenéticas com politomia são isomorfas às árvores filogenéticas sem politomias se o conjunto de objetos presentes nas duas árvores são os mesmos e preservarem a posição dos vértices que representam os objetos do conjunto S. Os autores inclusive

consideram a presença de nós vivos nas árvores. É demonstrado que é possível chegar a árvore sem politomia a partir da árvore com politomia e vice-versa com o mesmo número α de passos preservando a posição dos vértices do conjunto S garantido o isomorfismo. E sendo árvores isomórficas construir as duas é igualmente difícil.

Uma árvore filogenética com politomia é uma árvore multifurcada. Isso significa dizer que seus nós internos podem possuir mais de três vizinhos, diferente da árvore bifurcada tradicional em problemas de filogenia e filogenia viva.

Combinada com a filogenia viva, agora o problema pode gerar uma árvore em que os nós internos não só podem ser objetos vivos, como também podem ser relacionados com mais de 3 objetos.

CAPÍTULO

3

Metodologia

Neste capítulo serão descritos os métodos empregados no desenvolvimento da abordagem proposta para problema da filogenia viva com politomia. Primeiro serão conceituadas a programação linear e linear inteira. Em seguida, o algoritmo de *branch and bound* e *relax & fix* serão descritos.

3.1 Programação linear e programação linear inteira

A programação linear é um método de otimização que através de um modelo matemático tem como objetivo maximizar ou minimizar uma função objetivo dado um conjunto de variáveis de decisão e um conjunto de restrições. Um programa linear é considerado resolvido quando uma solução é encontrada ou quando é provado que não existe uma solução para o programa (Graver, 1975).

O modelo matemático a seguir é um exemplo hipotético com as variáveis de decisão x e y:

$$\begin{aligned} \min \quad & 8x + y & \\ s.a. \quad & x + y \leq 40 & (01) \\ & 2x + y \leq 60 & (02) \\ & x, y \in \mathbb{R}^{+} & (03) \end{aligned}$$

Neste exemplo hipotético o objetivo é minimizar o resultado de $8x + y$. Mas deve obedecer as restrições 01 que determina que a soma de $x + y$ deve ser menor que 40, a restrição 2 que determina que $2x + y$ não deve ser maior que

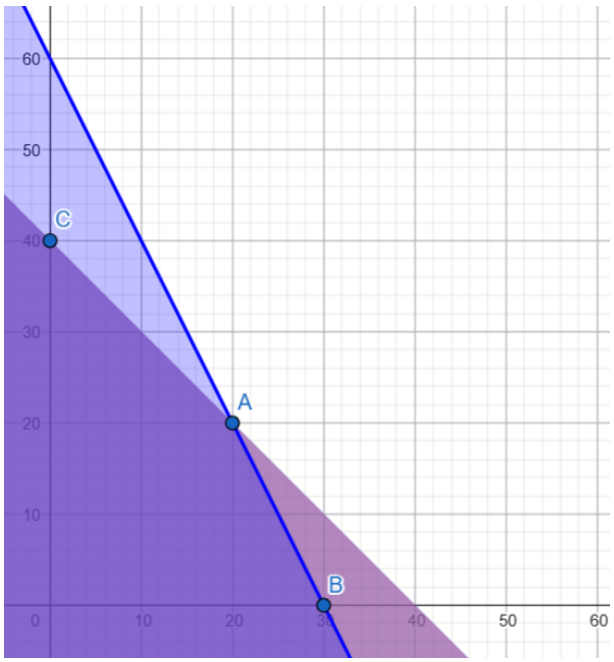


Figura 3.1: Representação gráfica do problema hipotético de exemplo. Os ponto A, B, C e (0,0) marcam os limites da área do espaço da solução do problema.

60 e a restrição 3 fornece o domínio dessas variáveis, sendo ele o dos números reais positivos.

É possível observar na Figura 3.1 a relação gráfica das restrições. O espaço de solução desse problema é a região que o espaço de solução de cada restrição se sobrepõem, limitadas pelos eixos x e y que delimitam o domínio de cada variável.

A programação linear inteira é uma especificação da programação linear, onde o domínio das variáveis de solução será ou o conjunto dos números inteiros ou será binário.

3.2 Branch and bound

Branch and bound, um dos algoritmos mais utilizados em problemas de otimização, é um método que tem como objetivo encontrar a solução ótima sem precisar explorar todo o espaço de solução viável.

O algoritmo explora o espaço de soluções viável S de um problema de otimização P, dividindo P em subproblemas menores, formando uma árvore de enumeração em que na raiz se encontra P. Ele irá dividir P criando restrições adicionais, normalmente fixando um valor (que esteja dentro do domínio da variável) a uma ou mais variáveis, ramificando o nó anterior. Para cada subproblema, será calculado os limitantes dual e primal.

O limitante dual é o melhor valor possível que pode ser alcançado dentro daquele subproblema, que é uma versão relaxada de seu problema pai - o

espaço de solução viável da relaxação, contém o espaço de solução viável do problema pai. Dessa forma, considerando-se um problema de minimização, o limitante dual de um nó é um limitante inferior, ou seja, nenhuma solução no espaço de soluções enraizada naquele nó terá um valor menor que esse limitante.

O limitante primal global é o melhor valor viável encontrado dentro de todos os espaços já explorados. De novo, considerando um problema de minimização, o limitante primal é um limitante superior, ou seja, ele representa o valor da melhor solução válida. Esse limitante é global e caso um nó encontre uma solução viável melhor que o primal atual, o primal é atualizado.

Utilizando-se dos limitantes dual e primal, o algoritmo poda a árvore de subproblemas quando os limitantes dos subproblemas são piores que os limitantes já obtidos, pois o espaço de solução da ramificação do subproblema não irá conter uma solução melhor que a já conhecida.

Utilizando o Figura 3.2 como referência e que o problema P_0 é um problema de minimização e que o limitante primal conhecido para ele é N, o algoritmo irá podar um nó em 3 ocasiões. Caso na criação da restrição para dividir o problema a variável assuma um valor fora de seu domínio aquele ramo é podado por não possuir uma solução em seu espaço de solução que seja viável, por exemplo, vamos supor que no problema original P_0 a variável x_1 não possa assumir um valor menor ou igual a 0, então o ramo do nó P_1 é podado. Outra ocasião em que o nó é podado acontece quando o limitante dual do nó é maior que o limitante primal global, por exemplo, caso o valor obtido para o limitando dual no nó P_4 seja $N+1$, este nó será podado, pois nenhum valor do espaço de solução será menor que $N+1$ já que neste caso o limitante dual é o limitante inferior e o limitante primal conhecido vale N. O último caso de poda ocorre quando quando os limitantes dual e primal de um nó são iguais, indicando que o valor da melhor solução naquele nó já foi encontrado. Por exemplo, caso o limitante dual e primal de P_6 forem iguais, ele será podado.

A árvore é gerada dinamicamente, começando pela raiz que possui o problema original P_0, os limitantes são definidos e então subproblemas são gerados a partir da adição de restrições ao problema original. Caso o valor do limitante não caia em nenhum caso de poda, aquele nó é ramificado. O algoritmo continua até que não exista mais subproblemas viáveis ou que a diferença entre os limitantes dual e primal globais seja 0 (ou próxima de um valor de tolerância permitido), garantindo que uma solução ótima, ou o mais próxima possível tenha sido encontrada.

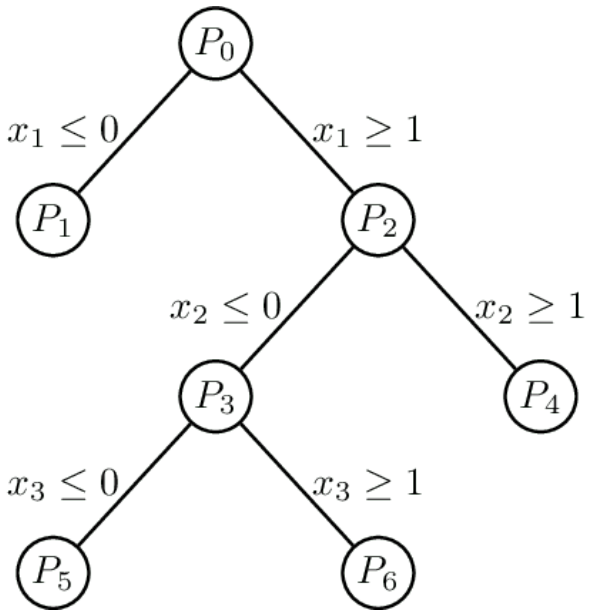


Figura 3.2: Exemplo hipotético de uma árvore gerada pelo algoritmo *branch-and-bound* (Riebesell e Bringuier, 2020).

3.3 Relax & fix

A heurística *relax & fix* consiste em particionar o conjunto de variáveis inteiras de um problema em conjuntos menores. O problema então é resolvido iterativamente em que, a cada iteração, somente as variáveis associadas à parte atual são consideradas inteiras. A cada iteração, as variáveis correspondentes às partes de iterações anteriores tem seus valores fixados, ou seja, não serão mais alterados. A ideia do método é que, ao final das iterações, todas as variáveis terão seus valores fixados e, portanto, uma solução viável será encontrada, como ilustrado na Figura 3.3. No entanto, pode ocorrer da sequência de variáveis fixadas levar a uma situação em que não há uma solução viável (Pochet e Wolsey, 2006).

Figura 3.3: Modelo de execução do *relax & fix*.

CAPÍTULO

4

Trabalho Desenvolvido

Neste capítulo será descrito o modelo adaptado para o problema da filogenia viva com politomia e as matheurísticas aplicadas ao *relax-and-fix* para gerar as árvores filogenéticas.

4.1 Modelagem do problema da filogenia viva com politomia

Utilizando o trabalho de Hoshino et al. (2018) como base para este trabalho, o terceiro modelo elaborado por Hoshino et al. (2018) foi escolhido para ser adaptado à politomia. Essa escolha foi motivada pelo fato desse modelo apresentar os melhores resultados entre os três apresentados no referido trabalho (mais nós explorados e melhores limitantes superiores).

Algumas restrições do trabalho de referência foram alteradas, e outras foram incluídas para permitir que um objeto interno da árvore pudesse ter 2 ou mais filhos. A seguir, descreve-se o modelo adaptado.

Dados um conjunto S de objetos de estudo, um grafo orientado $G = (V \cup H, A)$, no qual V é é o conjunto dos objetos em S, H o conjunto de objetos hipotéticos e A é o conjunto de arcos (i, j) e (j, i), uma função de distância $M : V \times V \mapsto \mathbb{R}^+$ que contém a distância para cada par de vértice em V de G, o problema irá buscar uma árvore T de G e uma função de peso $c : A \mapsto \mathbb{R}^+$ tal que:

- T contém todos os objetos em V,
- $d(i) \geq 3$ para todo vértice interno i em T e

- $\sum_{i,j \in V} |M_{ij} - p_{ij}|$ seja mínima, onde p_{ij} é o custo somado dos pesos das arestas do único caminho em T que liga os vértices i e j

Os vértices em H podem ou não aparecer em T.

As seguintes notações são adotadas no modelo:

- M_{ij} é a distância entre os objetos i e j da matriz M;
- V representa o conjunto dos objetos vivos;
- H representa os objetos hipotéticos;
- $V' = V \cup H$;
- $\delta^-(i)$ arestas que entram em i;
- $\delta^+(i)$ arestas que saem de i.

As variáveis de decisão do modelo com solução (T, c), onde T é a árvore filogenética e c a função de distância, são:

- x_a é uma variável binária em que x_a é igual a 1 se, e somente se, a aresta a for pertencente à árvore;
- y_a^i é o fluxo no arco a para o vértice i, representando que o arco participa do caminho da raiz até o vértice i;
- z_i é uma variável binária tal que z_i é igual a 1 se e somente se i é um vértice interno em T;
- g_i^{kl} indica que i é o último ancestral comum entre k e l;
- p_{ij} é a distância entre um vértice i e um vértice j na árvore.
- d_{ij} é a diferença entre M_{ij} e p_{ij};

Considere que a raiz r da árvore é um vértice arbitrário de T (e que não tem relação com ancestralidade, sendo selecionado aleatoriamente para realizar a construção da árvore, que será uma árvore filogenética viva com politomia sem raiz), considere também que fluxo é a existência de um caminho de r até um vértice i e que o último ancestral comum é o último vértice pelo qual passa um fluxo comum entre i e j. O modelo de programação linear inteira para o problema da filogenia viva com politomia é definido como se segue:

$$\min \quad \sum_{i\in V}\sum_{j\in V} d_{ij}$$

$$s.a. \sum_{a\in\delta^-(k)} y_a^k - \sum_{a\in\delta^+(k)} y_a^k = 1, \qquad \forall k\in V, k\neq r \quad (01)$$

$$\sum_{a\in\delta^-(i)} y_a^k - \sum_{a\in\delta^+(i)} y_a^k = 0, \qquad \forall k\in V, i\in V', i\neq k \quad (02)$$

$$y_a^k \leq x_a, \qquad \forall a\in V'\times V', \forall k\in V, k\neq r \quad (03)$$

$$\sum_{a\in\delta^-(r)} x_a = 0, \quad (04)$$

$$\sum_{a\in\delta^-(i)} x_a = z_i, \qquad \forall i\in H \quad (05)$$

$$\sum_{a\in\delta^+(r)} x_a \geq 1+2z_r \quad (06)$$

$$\sum_{a\in\delta^+(r)} x_a \leq |V'|z_r+1, \quad (07)$$

$$\sum_{a\in\delta^+(i)} x_a \geq 2z_i, \qquad \forall i\in V' \quad (08)$$

$$\sum_{a\in\delta^+(i)} x_a \leq |V'|z_i, \qquad \forall i\in V' \quad (09)$$

$$d_{ij} \leq M_{ij} - p_{ij}, \qquad \forall i,\forall j\in V \quad (10)$$

$$d_{ij} \leq p_{ij} - M_{ij}, \qquad \forall i,\forall j\in V \quad (11)$$

$$p_{rk} \geq p_{ri} + p_{ij} + p_{jk} - M(1-y_{ij}^k), \qquad \forall k\in V\setminus\{r\}, \forall i,j\in V', i\neq j, i\neq k, j\neq k \quad (12)$$

$$p_{rk} \leq p_{ri} + p_{ij} + p_{jk} + M(1-y_{ij}^k), \qquad \forall k\in V\setminus\{r\}, \forall i,j\in V', i\neq j, i\neq k, j\neq k \quad (13)$$

$$p_{kl} \geq p_{ik} + p_{il} - M(1-g_i^{kl}), \qquad \forall k,l\in V\setminus\{r\}, k\neq l, \forall i\in V', i\neq k, i\neq l \quad (14)$$

$$p_{kl} \leq p_{ik} + p_{il} + M(1-g_i^{kl}), \qquad \forall k,l\in V\setminus\{r\}, k\neq l, \forall i\in V', i\neq k, i\neq l \quad (15)$$

$$g_i^{kl} \geq (y_{ij}^k - y_{ij}^l) + (\sum_{x\in V'} y_{xi}^l - 1), \qquad \forall k,l\in V\setminus\{r\}, k\neq l, \forall i\in V', i\neq k, i\neq l \quad (16)$$

$$g_i^{kl} \geq (y_{ij}^l - y_{ij}^k) + (\sum_{x\in V\cup H} y_{xi}^k - 1), \qquad \forall k,l\in V\setminus\{r\}, k\neq l, \forall i\in V', i\neq k, i\neq l \quad (17)$$

$$\sum_{a\in\delta^-(i)} x_a = 1 \qquad \forall i\in V \quad (18)$$

$$x_{ij} + x_{ji} \leq 1, \qquad \forall i,\forall j\in V \quad (19)$$

$$x_a, g_k^{ij}, z_i \in \{0,1\} \quad (20)$$

$$y_a^k \in \mathbb{Z} \quad (21)$$

$$p_{ij}, d_{ij} \in \mathbb{R}, \qquad \forall i\in V, \forall j\in V \quad (22)$$

As restrições (01) e (02) aplicam restrição de fluxo, garantindo uma unidade de fluxo ao vértice k, saindo de r até k . A restrição (03) garante a existência do fluxo para o vértice (caso $y_a^k = 1$), ou seja, caso o arco a de G exista na árvore T. As restrições (04) , (06) e (07) dizem respeito à raiz da árvore. A restrição (04) garante que nenhum fluxo chegue a raiz, a (06) garante que da raiz saiam pelo menos 3 arcos com fluxo e a (07) garante que, caso a raiz seja um vértice folha, que saia apenas um arco com fluxo da raiz. A restrição (05), obriga que todo vértice hipotético tenha um arco de fluxo chegando nele e a restrição (18) aplica o mesmo aos vértices de objetos reais. As restrições (08), (09) referem-se a todos os vértices menos a raiz, com a (08) garantindo que pelo menos dois arcos de fluxos saiam dos vértices internos e a (09) que caso sejam folhas, não tenham fluxos saindo dos mesmos. As restrições (10) e (11) forçam a distância entre dois vértices reais a ser o módulo da diferença entre a distância na árvore e a distância na matriz M. As restrições (12) e (13) forçam que a distância da raiz a um vértice qualquer seja igual à soma das distâncias de todas as arestas no caminho entre raiz e dito vértice. Já as restrições (14) e (15) forçam que a distância entre dois vértices, em que nenhum deles é a raiz, seja a soma da distância das arestas no caminho entre o último ancestral comum e eles. As restrições (16) e (17) definem que o último ancestral i comum a dois vértices k e l é aquele que recebe fluxo para os dois e passa fluxo para um através de um arco [i, k] e para o outro através de [i, l]. A restrição (19) força a chegada de um fluxo ao vértice j a partir de i. Por fim, as restrições (20), (21) e (22) definem o domínio das variáveis.

4.2 Relax & fix

A qualidade das soluções geradas pela heurística *relax & fix* são diretamente dependentes dos critérios de particionamento das variáveis e da ordem em que elas serão fixadas. Para o problema da filogenia viva com politomia, foram propostas três heurísticas com o *relax & fix*, sequencial, aleatória e PAI, descritas a seguir.

4.2.1 Heurística sequencial

A heurística sequencial particiona o conjunto de variáveis seguindo a ordem em que é construído. Dado um conjunto ordenado *Vars* que contém as v variáveis, um conjunto ordenado *Part* vazio que armazenará as partes com cada subconjunto de variáveis, N sendo duas vezes o número de vértices e tam o tamanho de cada parte definida como $N-1$: a heurística sequencial irá armazenar nas N partes de *Part* as tam variáveis na mesma ordem que estão ordenadas em *Vars*.

4.2.2 Heurística aleatória

A heurística aleatória organiza o conjunto de variáveis de forma aleatória. Dados os mesmo conjuntos e valores anteriormente mencionados, a heurística aleatória seleciona uma variável aleatoriamente de *Vars* e adiciona na primeira parte de *Part*, seleciona aleatoriamente outra variável de *Vars* e novamente adiciona na primeira parte de *Part* até que a primeira parte e as demais estejam preenchidas com todas as variáveis.

4.2.3 Heurística PAI

A lógica da heurística PAI é fixar as variáveis de cada vértice que representem as arestas que chegam naquele vértice primeiro. Além dos conjuntos e valores já mencionados, aqui também serão usados os conjuntos V e H, contendo todos os vértices reais e hipotéticos, respectivamente. Primeiro um vértice v de V é sorteado aleatoriamente, então na primeira parte de *Part* são adicionadas as variáveis relacionadas as arestas de chegada em v. Um outro vértice de V é sorteado e as variáveis das suas arestas de chegada são adicionadas a segunda parte de *Part*. Após todos os vértices de V terem sido sorteados, faz-se o mesmo com os vértices de H, armazenando as variáveis das arestas nas partições seguinte, até que todos os vértices hipotéticos também tenham sido sorteados.

4.2.4 Utilização do algoritmo branch-and-bound

O algoritmo *branch-and-bound* é usado para buscar a solução do programa linear inteiro apresentado anteriormente. A cada iteração do algoritmo, ele irá buscar a solução utilizando as matheurística descritas nas subseções anteriores deste capítulo em cada nó da árvore que ele irá gerar. Para cada matheurística terá uma árvore do *branch-and-bound*.

CAPÍTULO

5

Resultados Computacionais

Neste capítulo serão descritas as instâncias utilizadas, o ambiente computacional, os resultados encontrados e análise desses resultados.

5.1 Instâncias

Para os testes computacionais foram usadas 23 instâncias. Uma instância possui 4 objetos, uma possui 12 e uma outra 14, dez instâncias possuem 8 objetos e outras dez possuem 10 objetos.

5.2 Ambiente computacional e implementação

Os algoritmos *relax & fix* foram implementados em C, utilizando a biblioteca de otimização SCIP (*Solving Constraint Integer Programs*) (Gamrath et al., 2016) com o resolvedor Soplex.

A máquina utilizada para rodar todos os testes tem um processador Intel(R) Core(TM) Ultra 7, com 16 GB de memória RAM e sistema operacional Linux.

Os tempos de execução foram fixados em 30 minutos e dois testes foram executados: o teste 1 estabeleceu como critério de parada para o *relax & fix* achar a primeira solução e com o tempo máximo de 60 segundos por iteração e o teste 2 teste rodou com tempo limite de 120 segundos e sem o critério de parada na primeira solução.

5.3 Resultados

Os resultados são apresentados nas tabelas abaixo. Na tabela 5.1 as colunas LB*n*, onde *n* é o número do teste, dizem respeito aos limitantes inferiores obtidos durante a execução do *branch and bound* e foram iguais para as três heurísticas testadas. As colunas UP_*x*, onde *x* é a abreviação dos nomes das matheurísticas (Seq para sequencial, Ale para aleatória e PAI) apresentam os limitantes superiores das três heurísticas nos dois testes, sendo Ale a heurística aleatória, Seq a sequencial e a PAI. O símbolo "*" foi usado para indicar que um limitante não foi encontrado.

Tabela 5.1: Resultados comparativos dos limitantes entre os dois testes

	Resultados com 60s por iteração				**Resultados com 120s por iteração**			
Instâncias	**LB1**	**UP_Ale**	**UP_Seq**	**UP_PAI**	**LB2**	**UP_Ale**	**UP_Seq**	**UP_PAI**
d_10_2_1_124_466_1.filo	637.00	*	*	*	637.00	*	*	*
d_10_4_1_209_928_1.filo	0.00	*	*	*	0.00	*	*	*
d_4_0_1_1.filo	0.00	4.00	0.00	3.00	0.00	0.00	0.00	0.00
d_6_2_1_10.filo	2.80	919.48	559.68	*	2.80	650.73	496.98	555.78
d_6_2_1_1.filo	0.00	*	*	415.80	0.00	672.20	173.80	*
d_6_2_1_2.filo	1.70	*	673.30	*	1.70	433.30	417.30	*
d_6_2_1_3.filo	0.00	*	672.45	1799.00	0.00	*	193.00	630.60
d_6_2_1_4.filo	0.00	*	415.80	*	0.00	*	313.00	417.50
d_6_2_1_5.filo	0.00	1810.40	671.40	1810.40	0.00	765.80	671.40	423.10
d_6_2_1_6.filo	0.00	*	571.85	544.70	0.00	*	423.10	1817.70
d_6_2_1_7.filo	0.00	674.69	*	*	0.00	*	435.08	1824.08
d_6_2_1_8.filo	0.00	*	688.08	*	0.00	*	469.00	*
d_6_2_1_9.filo	2.80	1541.20	320.78	514.18	2.80	715.98	330.78	*
d_6_4_1_10.filo	3.24	*	*	*	3.24	*	*	*
d_6_4_1_1.filo	0.00	*	*	*	0.00	*	*	*
d_6_4_1_2.filo	2.70	*	*	*	2.70	*	*	*
d_6_4_1_3.filo	0.00	*	*	*	0.00	*	*	*
d_6_4_1_4.filo	0.00	*	*	*	0.00	*	*	*
d_6_4_1_5.filo	0.00	*	*	*	0.00	*	*	*
d_6_4_1_6.filo	0.00	*	*	*	0.00	*	*	*
d_6_4_1_7.filo	0.00	*	*	*	0.00	*	*	*
d_6_4_1_8.filo	2.30	*	*	*	2.30	*	*	*
d_6_4_1_9.filo	3.24	*	*	*	3.24	*	*	*
Média	50.44	674.69	320.78	415.80	50.44	647.60	392.34	1050.25

Os limitantes inferiores foram ruins para todas as instâncias, sem diferença de valores entre os dois testes. Com relação aos limitantes superiores, todas as instâncias que acharam limitantes tiveram valores ainda muito altos, mas a heurística sequencial obteve valores melhores: 320,78 em média no teste com 60s de iteração. A heurística PAI teve melhor resultado que a aleatória: 415.80 e 674.69, respectivamente. No teste com 120s por iteração a heurística sequencial também teve valores melhores do limitante superior, seguida da aleatória e PAI, com o pior resultado: 392.34, 647.60, 1050.25.

Nos dois testes a heurística a achar mais soluções foi a sequencial: achou solução em 9 instâncias no teste com critério de parada na primeira solução encontrada e com 60 segundos por iteração, e em 11 instâncias no teste com 120 segundos por iteração. Já a heurística PAI achou solução em uma instân-

cia a mais que a heurística aleatória (6 e 5 no teste com 60s e 7 e 6 no teste com 120s, respectivamente). Nenhuma heurística achou solução viável para instâncias com mais de 10 objetos. Na instância com 14 objetos o algoritmo ultrapassou o limite de 30 minutos.

Com relação ao tempo de execução as tabelas 5.2 e 5.3 abaixo trazem os tempo do *branch and bound* e das heurísticas para achar uma solução. A coluna Time_*x*, na qual *x* é a abreviação dos nomes das matheurísticas, apresenta o tempo de execução total de cada instância e as colunas TimeHeur_*x* o tempo de execução de cada heurística *x*.

Tabela 5.2: Resultados do tempo de execução no teste de 60 segundos e critério de parada na primeira solução encontrada

	Resultados com 60s por iteração					
Instâncias	**Time_Ale**	**Time_Seq**	**Time_PAI**	**TimeHeur_Ale**	**TimeHeur_Seq**	**TimeHeur_PAI**
d_10_2_1_124_466_1.filo	62.12	62.20	62.71	60.81	60.80	61.44
d_10_4_1_209_928_1.filo	1803.01	1801.31	1806.29	0.00	0.00	0.00
d_4_0_1_1.filo	0.09	0.29	0.36	0.06	0.27	0.32
d_6_2_1_10.filo	59.56	14.35	64.79	54.20	9.18	60.08
d_6_2_1_1.filo	70.68	60.49	67.58	69.53	60.08	66.70
d_6_2_1_2.filo	70.66	28.85	116.58	69.50	27.93	115.78
d_6_2_1_3.filo	18.77	20.35	120.87	17.97	19.44	119.84
d_6_2_1_4.filo	72.09	21.26	99.84	71.09	20.41	99.00
d_6_2_1_5.filo	25.68	53.61	77.93	24.98	52.77	76.98
d_6_2_1_6.filo	23.05	15.59	96.75	22.20	14.78	95.79
d_6_2_1_7.filo	38.36	60.49	59.46	37.41	60.08	58.51
d_6_2_1_8.filo	70.82	51.21	111.49	69.81	50.19	110.57
d_6_2_1_9.filo	22.72	11.68	51.28	19.05	7.97	47.61
d_6_4_1_10.filo	472.98	438.53	437.51	92.40	60.27	60.29
d_6_4_1_1.filo	206.95	60.98	60.94	205.16	60.28	60.29
d_6_4_1_2.filo	60.97	60.97	60.97	60.31	60.29	60.29
d_6_4_1_3.filo	96.71	60.93	61.03	95.83	60.27	60.27
d_6_4_1_4.filo	194.92	61.03	61.03	193.29	60.29	60.31
d_6_4_1_5.filo	1019.85	978.80	974.74	98.34	60.26	60.25
d_6_4_1_6.filo	61.19	61.21	61.15	60.29	60.36	60.28
d_6_4_1_7.filo	61.08	60.99	61.04	60.31	60.28	60.27
d_6_4_1_8.filo	169.04	61.20	61.16	167.57	60.29	60.29
d_6_4_1_9.filo	60.88	60.80	60.91	60.29	60.26	60.33
Média	206.18	178.57	201.58	70.02	42.90	65.89

Com relação ao tempo de execução, no primeiro teste a heurística sequencial obteve os melhores tempos de forma geral, com uma média de 178.57s por instância durante o *branch and bound* e 42.90s na execução das iterações da heurística. As heurísticas aleatórias e PAI tiveram pouca diferença entre si em seus tempos médios, mas quando comparadas o tempo de execução das heurísticas nas instâncias que acharam solução, a aleatória teve um tempo médio mais baixo que a PAI (27.14 segundos em comparação com 67.87 segundos). Já no segundo teste, a heurística aleatória teve melhores tempos médios por instância, com uma diferença pequena para a heurística sequencial, e a PAI os piores (281.75, 299.44, 540.86). O mesmo é válido para o tempo de execução da heurística (144.82, 162.65, 404.16).

Analisando os resultados, nenhuma heurística obteve valores bons (baixo número de instâncias onde uma solução foi encontrada, limitantes inferiores muito baixos e superiores muito altos). A heurística sequencial foi a que

Tabela 5.3: Resultados do tempo de execução no teste de 120 segundos

	Resultados com 120s por iteração					
Instâncias	**Time_Ale**	**Time_Seq**	**Time_PAI**	**TimeHeur_Ale**	**TimeHeur_Seq**	**TimeHeur_PAI**
d_10_2_1_124_466_1.filo	122.20	123.13	123.00	120.80	121.72	121.52
d_10_4_1_209_928_1.filo	1801.39	1801.41	1802.96	0.00	0.00	0.00
d_4_0_1_1.filo	0.11	0.30	0.84	0.06	0.25	0.80
d_6_2_1_10.filo	155.72	72.63	1331.71	150.02	66.76	1321.83
d_6_2_1_1.filo	147.53	541.47	371.06	145.50	536.48	369.71
d_6_2_1_2.filo	265.35	145.52	965.35	263.38	143.35	961.96
d_6_2_1_3.filo	10.12	430.89	848.19	9.24	426.47	845.19
d_6_2_1_4.filo	151.82	38.94	767.35	150.47	37.99	763.08
d_6_2_1_5.filo	129.18	130.09	1018.03	127.80	128.84	1013.03
d_6_2_1_6.filo	148.75	154.75	621.71	147.47	151.89	619.17
d_6_2_1_7.filo	138.51	568.09	618.07	137.26	563.26	615.52
d_6_2_1_8.filo	143.39	236.37	735.35	141.96	233.70	732.80
d_6_2_1_9.filo	76.36	132.58	613.37	72.10	127.38	607.95
d_6_4_1_10.filo	502.41	502.74	496.97	120.32	120.28	120.29
d_6_4_1_1.filo	218.42	120.94	121.02	217.03	120.27	120.27
d_6_4_1_2.filo	274.46	120.94	120.99	273.25	120.28	120.30
d_6_4_1_3.filo	121.09	121.03	121.05	120.32	120.31	120.29
d_6_4_1_4.filo	257.46	120.96	121.11	255.98	120.27	120.31
d_6_4_1_5.filo	1054.17	1039.89	1036.46	120.29	120.28	120.27
d_6_4_1_6.filo	241.72	121.23	121.17	240.58	120.29	120.24
d_6_4_1_7.filo	121.15	121.10	121.14	120.29	120.26	120.36
d_6_4_1_8.filo	277.95	121.22	241.83	276.55	120.28	240.56
d_6_4_1_9.filo	121.02	120.95	120.94	120.28	120.29	120.30
Média	281.75	299.44	540.86	144.82	162.65	404.16

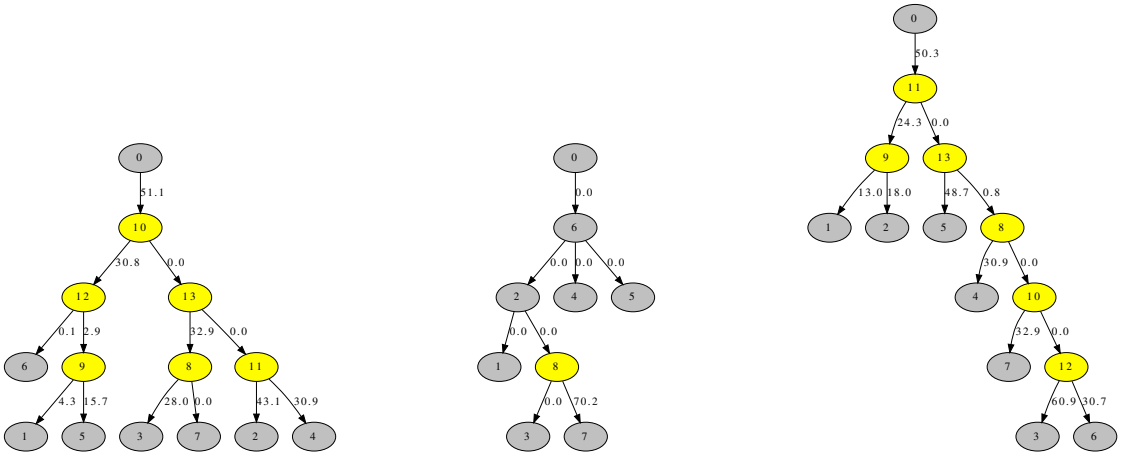
teve melhores resultados nos dois testes executados. Achou mais solução, e mesmo tendo um tempo superior no teste com 120 segundos por iteração, a diferença foi pequena e ela achou solução em quase o dobro de instâncias que a heurística aleatória.

A pretensão deste trabalho, era que através da adaptação para permitir a politomia nas árvores filogenéticas do terceiro modelo de programação linear inteira proposto por Hoshino et al. (2018), o modelo achasse soluções em mais instâncias e que as soluções fossem melhores. No entanto, esse trabalho segue parcialmente os resultados obtidos por Hoshino et al. (2018). As instâncias com mais de 10 objetos tiveram resultados ruins em Hoshino et al. (2018) e neste trabalho as matheurística não acharam solução, os limitantes inferiores tiveram os mesmos valores em ambos os trabalhos, e comparativamente os valores dos limitantes superiores pro presente trabalho foram piores que o do modelo adaptado indicando que a politomia e as matheurísticas implementadas não auxiliaram a encontrar melhores limitantes inferiores e superiores, o que sugere que as soluções ótimas não estavam perto dos valores encontrados pelas matheurísticas.

Algumas árvores filogenéticas obtidas podem ser observadas abaixo.

Os nós amarelos são vértices hipotéticos. As três árvores divergem bastante (A, B e C). Observe que a árvore gerada pela heurística aleatória (B) possui nós internos vivos e politomia no nó 6 (Figura 5.1).

As duas árvores também são diferentes, as duas possuem politomia (D e E), mas apenas a árvore gerada pela heurística aleatória (E) gerou uma árvore com nós internos vivos, dessa vez no nó raiz (Figura 5.2).

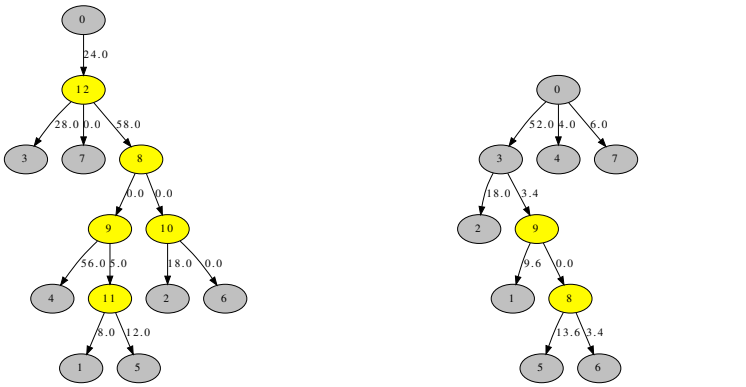


(A) Heurística sequencial
Valor da solução: 320.78

(B) Heurística aleatória
Valor da solução: 1541.20

(C) Heurística PAI
Valor da solução: 514.18

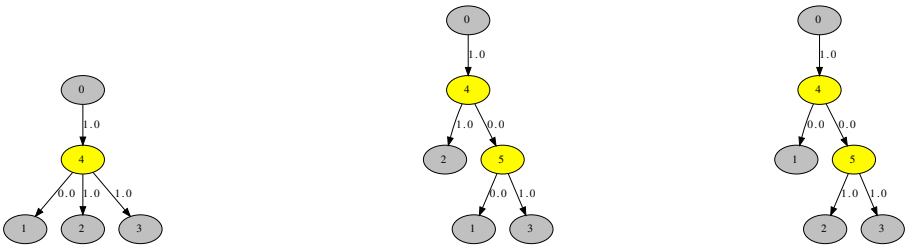
Figura 5.1: Árvores geradas no teste de 60s com critério de parada na primeira solução para a instância d_6_1_9.



(D) Heurística sequencial
Valor da solução: 173.80

(E) Heurística aleatória
Valor da solução: 672.20

Figura 5.2: Árvores geradas no teste de 120s para a instância d_6_2_1_1. A heurística PAI não teve solução para essa instância.



(F) Heurística sequencial
Solução ótima: 0

(G) Heurística aleatória
Solução ótima: 0

(H) Heurística PAI
Solução ótima: 0

Figura 5.3: Árvores geradas no teste de 120s para a instância d_4_1_1.

No caso da menor instância, com 4 objetos, no teste com 120 segundos por iteração, as três heurísticas acharam a solução ótima, mas geraram três árvores diferentes (F, G e H - os nós folhas das árvores geradas pela heurística aleatória e PAI são diferentes) (Figura 5.3).

Analisando a topografia das árvores e comparando as árvores obtidas para uma mesma instâncias pelas três matheurísticas propostas, pode ser que as diferenças de configuração para as mesmas instância seja um problema para as matheurística propostas, já que é os valores das soluções também não são bons e mesmo na instância em que foi encontrado o valor ótimo, não há como dizer qual árvore representa melhor a instância.

CAPÍTULO

6

Contribuições e Conclusões

Neste capítulo serão discutidas as contribuição deste trabalho ao problema, alguns possíveis trabalhos futuros e as conclusões.

6.1 Contribuições

Este trabalho abordou um problema ainda pouco explorado na literatura. O problema da filogenia viva é um problema relativamente recente, datando de 2013. O uso da heurística *relax-and-fix* também é nova na pesquisa do problema da filogenia viva.

Próximos passos para este trabalho incluem, mas não se limitam, a testar as matheurísticas propostas em instâncias reais para analisar se soluções serão achadas e caso sejam, como são as árvores geradas quando comparadas com as árvores já conhecidas para as instâncias testadas; analisar o modelo matemático em busca de melhorias e testar a heurística proposta por Joncour et al. (2023): uma generalização do *relax-and-fix* que particiona o conjunto de variáveis em globais e locais e organiza a ordem de execução das variáries locais, chamada pelos autores de *generalized relax-and-fix*.

6.2 Conclusão

Conclui-se com este trabalho que a aplicação da heurística *relax-and-fix* como foi feita não trouxe resultados interessantes para o problema da filogenia viva com politomia, mas ainda há espaço para melhorias serem feitas. Os resultados obtidos são similares a outros trabalhos com o problema da filoge-

nia viva, mas sem politomia: instâncias com mais de 8 objetos tem resultados piores, acham menos soluções e não encontram limitantes bons.

Referências Bibliográficas

Araújo, G. S., Telles, G. P., Walter, M. E. M., e Almeida, N. F. (2017). Distance-based live phylogeny. In *International Conference on Bioinformatics Models, Methods and Algorithms*, volume 4, p. 196–201. SCITEPRESS. Citado nas páginas 7, 8, e 10.

Coyne, J. A., Elwyn, S., Kim, S. Y., e Llopart, A. (2004). Genetic studies of two sister species in the drosophila melanogaster subgroup, d. yakuba and d. santomea. *Genetics Research*, 84(1):11–26. Citado na página 5.

Davidson, R. e Sullivant, S. (2014). Distance-based phylogenetic methods around a polytomy. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 11(2):325–335. Citado na página 5.

Gamrath, G., Fischer, T., Gally, T., Gleixner, A. M., Hendel, G., Koch, T., Maher, S. J., Miltenberger, M., Müller, B., Pfetsch, M. E., Puchert, C., Rehfeldt, D., Schenker, S., Schwarz, R., Serrano, F., Shinano, Y., Vigerske, S., Weninger, D., Winkler, M., Witt, J. T., e Witzig, J. (2016). The SCIP Optimization Suite 3.2. ZIB-Report 15-60, Zuse Institute Berlin. URL http://nbn-resolving.de/urn:nbn:de:0297-zib-57675. [Acessado em 06-12-2025]. Citado na página 22.

Graver, J. E. (1975). On the foundations of linear and integer linear programming i. *Mathematical Programming*, 9(1):207–226. Citado na página 12.

Hennig, W. (1999). *Phylogenetic systematics*. University of Illinois Press. Citado na página 2.

Hodge, T. e Cope, M. J. T. (2000). A myosin family tree. *Journal of cell science*, 113(19):3353–3354. Citado na página 4.

Hoshino, E. A., Araujo, B. A., e Freitas, V. O. (2018). Algoritmos exatos para o problema da filogenia viva. In *XIX Latin-Iberoamerican Conference on Operations Research*, p. 276–283. Citado nas páginas 8, 17, e 25.

Joncour, C., Kritter, J., Michel, S., e Schepler, X. (2023). Generalized relax-and-fix heuristic. *Computers & Operations Research*, 149:106038. Citado na página 28.

Lenzini, G. e Marianelli, S. (1997). Algorithms for phylogeny reconstruction in a new mathematical model. *Calcolo-A Quarterly on Numerical Analysis and Theory of Computation*, 1(34):1–24. Citado nas páginas 2, 6, e 10.

Papamichail, D., Huang, A., Kennedy, E., Ott, J.-L., Miller, A., e Papamichail, G. (2017). Live phylogeny with polytomies: Finding the most compact parsimonious trees. *Computational Biology and Chemistry*, 69:171–177. Citado na página 8.

Pochet, Y. e Wolsey, L. A. (2006). *Production planning by mixed integer programming*, volume 149. Springer. Citado na página 15.

Riebesell, J. e Bringuier, S. (2020). Collection of scientific diagrams. URL https://github.com/janosh/diagrams. 10.5281/zenodo.7486911 - https://github.com/janosh/diagrams. Citado na página 15.

Robinson, D. e Foulds, L. (1981). Comparison of phylogenetic trees. *Mathematical Biosciences*, 53(1):131–147. ISSN 0025-5564. URL https://www.sciencedirect.com/science/article/pii/0025556481900432. [Acessado em 06-12-2025]. Citado nas páginas 7 e 10.

Setubal, J. C. e Meidanis, J. (1997). *Introduction to computational molecular biology*. PWS Pub. Boston. Citado nas páginas 9 e 10.

Stiller, J., Feng, S., Chowdhury, A.-A., Rivas-González, I., Duchêne, D. A., Fang, Q., Deng, Y., Kozlov, A., Stamatakis, A., Claramunt, S., et al. (2024). Complexity of avian evolution revealed by family-level genomes. *Nature*, 629 (8013):851–860. Citado na página 2.

Telles, G. P., Almeida, N. F., Minghim, R., e Walter, M. E. M. (2013). Live phylogeny. *Journal of Computational Biology*, 20(1):30–37. Citado nas páginas 4, 6, 10, e 28.

University of California Museum of Paleontology. Phylogenetic systematics (evolutionary trees). https://evolution.berkeley.edu/phylogenetic-systematics/ [Acesado em 06-12-2025]. Citado nas páginas 5 e 6.