Universidade Federal do Ceará

Centro de Ciências

Departamento de Computação

Doutorado em Ciência da Computação

Técnicas para Construção de Árvores Filogenéticas

Gerardo Valdisio Rodrigues Viana

Tese de Doutorado

Fortaleza–Ceará

6 de maio de 2007

Universidade Federal do Ceará

Centro de Ciências

Departamento de Computação

Gerardo Valdisio Rodrigues Viana

Técnicas para Construção de Árvores Filogenéticas

Trabalho apresentado ao Programa de Doutorado em Ciência da Computação do Departamento de Computação da Universidade Federal do Ceará como requisito parcial para obtenção do grau de Doutor em Ciência da Computação.

Orientador: *Prof. Dr. Fernando A. de C. Gomes, DC/UFC*

Co-orientador: *Prof. Dr. Carlos Eduardo Ferreira, IME/USP*

Fortaleza–Ceará

6 de maio de 2007

Gerardo Valdisio Rodrigues Viana

Possui graduação em Engenharia Mecânica pela UFC - Universidade Federal do Ceará (1976), licenciatura em Matemática pela UECE - Universidade Estadual do Ceará (1976) e especialização (1986) e mestrado (1996) em Ciência da Computação pela UFC. Realizou estágio de Doutorado "Sandwich" no IME/USP de Março de 2006 a Fevereiro de 2007. Atualmente é Professor Adjunto da UFC e da UECE.

Universidade Federal do Ceará
Centro de Ciências – Departamento de Computação

Gerardo Valdisio Rodrigues Viana

Técnicas para Construção de Árvores Filogenéticas

Tese apresentada ao Programa de Pós-Graduação do Departamento de Computação da Universidade Federal do Ceará como parte dos requisitos para obtenção do título de **Doutor em Computação**. Aprovada pela Banca Examinadora abaixo assinada.

Prof. Dr. Fernando Antônio de Carvalho Gomes – **Orientador**
Departamento de Computação - UFC

Prof. Dr. Carlos Eduardo Ferreira – **Co-Orientador**
Instituto de Matemática e Estatística - IME/USP

Prof. Dr. Nelson Maculan Filho
Programa de Engenharia de Sistemas e Computação - COPPE/UFRJ

Prof. Dr. Ruy Luiz Milidiú
Departamento de Informática - PUC/Rio

Prof. Dr. Tarcisio Haroldo Cavalcante Pequeno
Departamento de Computação - UFC

Prof. Dr. Thales Barbosa Grangeiro
Departamento de Biologia - UFC

Fortaleza, 27 de abril de 2007

Ficha Catalográfica elaborada pela bibliotecária Aline Vieira

V667t Viana, Gerardo Valdisio R.

Técnicas para construção de árvores filogenéticas / Gerardo Valdisio R. Viana, Fernando Antônio de Carvalho Gomes (Orientador) e Carlos Eduardo Ferreira (co-Orientador).

204f.: il.: 30cm

Tese (Doutorado) Computação
Universidade Federal do Ceará, Fortaleza, 2007.

1.Filogenia 2.Otimização Combinatória 3.Programação Paralela (Computação) I. Gomes, Fernando Antônio de Carvalho II. Universidade Federal do Ceará III. Título

CDD 005.2

Dedico esta obra à minha mãe,
Valquíria Viana de Santiago.

Agradecimentos

A Deus por ter me conduzido até aqui.

À minha família pelo carinho, apoio, compreensão e confiança.

A meus orientadores Professores Fernando (UFC) e Carlinhos (USP) pela excelente orientação prestada, além do apoio e incentivo constantes.

Aos colegas professores da UFC e da UECE que me substituíram por ocasião de meu afastamento. Em especial ao Prof. Bitú por conduzir bem as numerosas turmas de Cálculo Numérico.

Aos professores Clécio Thomaz, Fernando Carvalho, Júlio Wilson, Lassance Silva, Manoel Campêlo, Maurício Mota e {Marcelino, Mauro e Tarcísio} Pequeno por seus valiosos ensinamentos nas disciplinas do Mestrado e do Doutorado.

Ao meu amigo Cláudio Meneses, parceiro das pesquisas, colega do dia-a-dia na USP e colaborador na concepção deste trabalho.

A minhas filhas Marina, bióloga, por me esclarecer conceitos básicos de Biologia e Livia, psicóloga, por me ensinar a combater o estresse.

À Funcap pelo auxílio no custeio de minha estada em São Paulo.

A meus comprades Rolim e Lizete e minhas irmãs Valzete e Valzeir pelas acolhidas em suas residências nos momentos de lazer e fugas do frio de São Paulo.

Aos bolsistas e funcionários do Laboratório de Informática do IME/USP e do LIA (Laboratórios de Pesquisa em Ciência da Computação) e LACAD (Laboratório de Computação de Alto Desempenho), ambos da UFC, que anonimamente me auxiliaram ao disponibilizar em tempo integral os recursos de *hardware* e *software*.

Aos pesquisadores Andreatta, Ribeiro e Vianna [AR02, RV05] pela cessão dos arquivos de dados contendo as instâncias utilizadas nos experimentos computacionais.

Embora ninguém possa voltar atrás e fazer um novo começo, qualquer um pode começar agora e fazer um novo fim.

— CHICO XAVIER

Resumo

Árvores filogenéticas são estruturas que expressam a similaridade, ancestralidade e relacionamentos entre as espécies ou grupo de espécies. Conhecidas como árvores evolucionárias ou simplesmente filogenias, as árvores filogenéticas possuem folhas que representam as espécies (táxons) e nós internos que correspondem aos seus ancestrais hipotéticos. Neste trabalho, além das informações necessárias para o entendimento de toda a sistemática filogenética, são apresentadas técnicas algorítmicas para construção destas árvores. Os conceitos básicos de biologia molecular, evolução da vida e classificação biológica, aqui descritos, permitem compreender o que é uma Filogenia e qual sua importância para a Biologia. As informações filogenéticas fornecem, por exemplo, subsídios importantes para decisões relativas aos transplantes de órgãos ou tecidos de outras espécies para o homem e para que testes de reação imunológica ou de toxicidade sejam feitos antes em outros sistemas biológicos similares ao ser humano. Resolver um Problema de Filogenia corresponde à construção de uma árvore filogenética a partir de dados conhecidos sobre as espécies em estudo, obedecendo a algum critério de otimização. A abordagem dada a esse problema envolve duas etapas, a primeira, referente aos casos em que as filogenias são perfeitas cujos procedimentos desenvolvidos serão utilizados na segunda etapa, quando deve ser criada uma técnica de inferência para a filogenia num caso geral. Essas técnicas consideram de forma peculiar as hipóteses sobre o processo de evolução. Para muitas hipóteses de interesse o problema se torna NP-Difícil, justificando-se o uso de métodos de inferência através de heurísticas, meta-heurísticas e algoritmos aproximativos. Nossa contribuição neste trabalho consiste em apresentar uma técnica de resolução desse problema baseada em buscas locais com partidas múltiplas em vizinhanças diversificadas. Foi utilizada a programação paralela para minimizar o tempo de execução no processo de intensificação da busca pela solução ótima do problema. Desta forma, desenvolvemos um algoritmo para obter soluções aproximadas para um Problema da Filogenia, no caso, para inferir, a partir de matrizes de características de várias espécies, uma árvore filogenética que mais se aproxima da história de sua evolução. Uma estrutura de dados escolhida adequadamente aliada à manipulação de

dados em binário em algumas rotinas facilitaram a simulação e ilustração dos testes realizados. Instâncias com resultados conhecidos na literatura foram utilizadas para comprovar a performance do algoritmo. Esperamos com este trabalho despertar o interesse dos pesquisadores da área de Computação, consolidando, assim, o crescimento da Bioinformática.

Palavras-chave: Filogenia, Árvores Filogenéticas, Biologia Computacional, Otimização Combinatória, Algoritmos e Heurísticas.

Abstract

Phylogenetic tree structures express similarities, ancestrality, and relationships between species or group of species, and are also known as evolutionary trees or phylogenies. Phylogenetic trees have leaves that represent species (taxons), and internal nodes that correspond to hypothetical ancestors of the species. In this thesis we first present elements necessary to the comprehension of phylogenetic trees systematics, then efficient algorithms to build them will be described. Molecular biology concepts, life evolution, and biological classification are important to the understanding of phylogenies. Phylogenetic information may provide important knowledge to biological research work, such as, organ transplantation from animals, and drug toxicologic tests performed in other species as a precise prediction to its application in human beings. To solve a phylogeny problem implies that a phylogenetic tree must be built from known data about a group of species, according to an optimization criterion. The approach to this problem involves two main steps: the first refers to the discovery of perfect phylogenies, in the second step, information extracted from perfect phylogenies are used to infer more general ones. The techniques that are used in the second step take advantage of evolutionary hypothesis. The problem becomes NP-hard for a number of interesting hypothesis, what justify the use of inference methods based on heuristics, metaheuristics, and approximative algorithms. The description of an innovative technique based on local search with multiple start over a diversified neighborhood summarizes our contribution to solve the problem. Moreover, we used parallel programming in order to speed up the intensification stage of the search for the optimal solution. More precisely, we developed an efficient algorithm to obtain approximate solutions for a phylogeny problem which infers an optimal phylogenetic tree from characteristics matrices of various species. The designed data structures and the binary data manipulation in some routines accelerate simulation and illustration of the experimentation tests. Well known instances have been used to compare the proposed algorithm results with those previously published. We hope that this work may arise researchers' interest to the topic and contribute to the Bioinformatics area.

Keywords: Phylogeny, Phylogenetic Trees, Computational Biology, Optimization, Algorithms and Heuristics.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

CAPÍTULO 1

Introdução

As árvores filogenéticas devem expressar similaridade, ancestralidade ou parentescos evolutivos entre espécies ou grupos de espécies. Tais árvores são inferidas a partir dos dados disponíveis de tal forma que espécies "mais próximas" tenham nós correspondentes mais próximos na árvore.

A construção de árvores filogenéticas auxilia a Biologia a explicar os possíveis relacionamentos entre as espécies atuais e a deduzir as histórias evolutivas das mesmas. Conhecidas simplesmente como Filogenias [AR02, RV05], tais árvores possuem folhas que representam as espécies e nós internos que correspondem aos seus ancestrais hipotéticos.

A partir de informações catalogadas em grandes bases, é possível inferir filogenias utilizando-se de dados, como distâncias e características, relativos aos genes de cada espécie. As características correspondem aos atributos que possuem cada táxon (denominação dada a uma espécie ou a um grupo de espécies), sendo possível obter as relações entre eles com base na similaridade destes atributos. As distâncias referem-se às diferenças relativas do tempo de evolução entre os táxons, de modo que sua grandeza indica se as espécies serão dispostas na árvore evolutiva, em ramificações próximas ou distantes, de forma proporcioal aos valores dispostos numa "matriz de distâncias", como veremos adiante.

Como existem várias formas de representar estas relações para um determinado conjunto de dados, gera-se o problema de dedução de qual delas é a melhor. Este problema constitui um desafio que é a escolha do diagrama em árvore mais apropriado, ou seja, da filogenia que represente fielmente a história evolutiva das espécies em estudo.

São conhecidos diversos métodos de construção de filogenias, cada um tratando de forma peculiar as hipóteses sobre o processo de evolução. Fitch [Fit84] afirma que cada cientista defende seu método de inferência [DG05, Fel93, GGLD05, SOWH96], mostrando argumentos em seu favor e sendo sempre capaz de encontrar argumentos contra todos os demais.

Um método filogenético consiste basicamente na separação das espécies em grupos, correspondentes a subárvores, que compartilham propriedades comuns. A formação destes grupos é possível em função das transformações que ocorrem ao longo do processo evolutivo.

Um dos métodos adotados, e com bastante aceitação, é o da *parcimônia* que consiste em inferir que a melhor árvore filogenética para uma dada instância é aquela que requer o menor número de transformações evolutivas, correspondendo na natureza ao "caminho" de menor esforço. Portanto, podemos dizer, sob este critério, que o **Problema da Filogenia** é um problema de otimização, no caso de minimização, que consiste em obter a Árvore Filogenética com um número mínimo de passos evolutivos.

Como é possível para uma dada instância que o método de resolução utilizado apresente mais de uma solução, os biólogos devem comparar e decidir qual das árvores obtidas é a melhor. Cabe a nós, da área da computação, criar técnicas algorítmicas eficientes para reconstruir as filogenias, apresentando pelo menos uma árvore como solução, preferencialmente, no menor tempo possível.

O objetivo deste trabalho, além de fornecer todas as informações necessárias para o entendimento do Problema da Filogenia, é mostrar como se constrói uma árvore filogenética, seja a partir de uma matriz de distâncias ou de uma matriz de características; quer a filogenia seja perfeita, ou não. Como na prática, filogenias perfeitas não existem, o objetivo maior deste trabalho é criar uma boa técnica para reconstruir qualquer filogenia baseada em hipóteses. Nosso propósito, portanto, é apresentar algoritmos (no caso heurísticas) que resolvam o problema geral da filogenia obedecendo ao critério de otimização, citado anteriormente, conhecido como parcimônia.

Para atingir esses objetivos, o desenvolvimento deste trabalho é feito da forma como segue. O Capítulo 2 apresenta uma visão geral do estado da arte da área de Sistemática Filogenética, inicialmente através de conceitos básicos de Biologia Molecular e processos de Evolução da Vida, e ao final evidenciando a importância do estudo da Filogenia e suas aplicações.

A seguir, no Capítulo 3, são apresentados outros conceitos, especificamente na área da Biologia Computacional, mostrando sua aplicabilidade em geral e apresentando de forma resumida, em particular, o Problema da Filogenia que, conforme mostrado na Seção 3.4 trata-se

de um problema NP-Difícil [BFW93, DJS86, FG82, GJ79], o que justifica o uso de métodos de inferência através de heurísticas [AR02, BGP05], metaheurísticas [DK87, RV05, Sta05] e algoritmos aproximativos [WJL96, VGMP06, VTS04]. Também neste capítulo é feita uma abordagem de alguns métodos utilizados para resolver outros problemas da bioinformática, especialmente aqueles referentes às técnicas de alinhamento que são procedimentos usados para geração de matrizes de distâncias.

No Capítulo 4 são mostradas as formas de representação existentes para árvores filogenéticas e descritos dois métodos de construção das mesmas. Ao final do capítulo é dada uma definição de Filogenia Perfeita.

No Capítulo 5 é apresentado um algoritmo criado especialmente para construir árvores filogenéticas baseadas em características, restrito às instâncias sem incongruências, ou seja, quando a Filogenia é considerada "perfeita". Esta rotina é usada como auxiliar às técnicas de inferência da Filogenia, de forma geral, para qualquer instância do problema. Os resultados dos primeiros experimentos computacionais são apresentados neste capítulo.

No Capítulo 6 é apresentada a heurística desenvolvida para inferir uma árvore filogenética. As estratégias usadas para obtenção de bons resultados num tempo de execução reduzido são mostradas em detalhes. A comparação através de resultados conhecidos na literatura são apresentados em novos experimentos computacionais que comprovam a boa qualidade da heurística.

No Capítulo 7 foi utilizada a Computação de Alto Desempenho [Via98] com o objetivo de minimizar os tempos de execução dos algoritmos através de processos paralelos. O procedimento utilizado consiste no seguinte: após uma solução ter sido obtida pelo uso da heurística referida no capítulo 6, é executada uma nova heurística criada para tentar a melhoria deste resultado. Como o espaço de busca é enorme, optou-se por utilizar a programação paralela, via PVM (*Parallel Virtual Machine*) [Bar94, GBD94], a fim de que simultaneamente o espaço de configurações viáveis seja explorado em vizinhanças múltiplas. Testes foram executados para as mesmas instâncias utilizadas nos experimentos anteriores a fim de analisar a performance do algoritmo.

Por fim, no Capítulo 8 é apresentada a conclusão deste trabalho bem como propostas su-

gestões para a elaboração de trabalhos futuros e no Apêndice "A", logo após a citação de toda bibliografia consultada, encontram-se as abreviaturas e siglas utilizadas em todo o texto.

Poderíamos considerar que os algoritmos apresentados nos Capítulos 6 e 7 caracterizam uma metaheurística [Via98]. Mesmo sendo específica para o problema de filogenia com base em características com dois estados (binárias), alteração e adaptação nas estruturas de dados e respectivas configurações podem ser feitas para resolver outros problemas, o que tornaria o método criado mais abrangente.

Justifica-se esta classificação pois procuramos usar as potencialidades de diversas técnicas de resolução de problemas de otimização combinatória conhecidas. Por exemplo, a geração de uma população inicial para a heurística serial é uma característica herdada dos algoritmos genéticos [GKL95, Hol75]. A programação dinâmica nos auxiliou a idealizar o uso da tabela que guarda as configurações "visitadas", onde seu índice de entrada é o próprio valor da função custo, além do que, guardar o histórico destas configurações a fim de controlar e "proibir" repetições é uma característica da metaheurística Busca Tabu [GL93]. Processos de busca local e aleatorização estão presentes em GRASP[1] [PR01] e a rotina que troca dois a dois todos os elementos de uma dada configuração faz com que no início desse processo mais trocas sejam aceitas, havendo uma "estabilidade" ao final, da mesma forma como ocorre na metaheurística *Simulated Annealing* [KGV83, vLA87]. De modo geral, procuramos sempre realizar etapas alternadas e controladas das fases de diversificação e intensificação no processo de busca de uma solução, como qualquer metaheurística faz.

[1] Greedy Randomized Adaptive Search Procedures

CAPÍTULO 2

Conceitos Básicos

2.1 Classificação Biológica

O naturalista sueco Carl von Linné (1707-1778), ou Carolus Linnaeus, divulgou suas idéias sobre classificação biológica no livro *Systema Naturae* publicado em 1735. Lineu, como é conhecido no Brasil, ponderava que critérios de semelhança devem ser o ponto de partida de todas as classificações e que características morfológicas e anatômicas são as mais adequadas para agrupar os seres vivos. Na época, ele imaginava que existiam cerca de 10 mil tipos distintos de forma de vida terrestre; desde então, os biólogos já identificaram mais de dois milhões de espécies, o que os levaram a concluir a necessidade de desenvolver um sistema eficiente para organizar a enorme diversividade biológica [Pon90]. Nos dias atuais, acreditam os cientistas, devem existir cerca de 30 milhões de espécies.

Um dos grandes méritos de Lineu foi associar à classificação uma nomenclatura binomial, composta sempre por duas palavras, a primeira um nome **genérico** e a segunda um nome **específico**; assim, *Canis familiaris* (cachorro), *Canis lupus* (lobo cinzento), *Canis mesomelas* (chacal) e *Canis latrans* (coiote), embora **espécies** distintas, têm semelhanças comuns que permitem agrupá-las em uma categoria imediatamente superior, chamada **gênero** [AM04]. A importância desta catalogação é a unicidade da identificação de qualquer espécie independentemente do idioma utilizado ou da região em que a mesma vive.

Denomina-se *táxon* a uma espécie isolada ou a um grupo de espécies que compartilham determinadas características. De maneira geral, este termo é usado para denominar qualquer sistema cujos elementos são populações biológicas [Amo97]; por exemplo, o conjunto de todas as espécies que voam (aves, morcegos, insetos, peixe-voador etc) poderiam compor um táxon; as cobras e as serpentes, em seu conjunto, formam um táxon e assim por diante. A partir deste termo, criou-se a Taxonomia, denominada de "ciência da identificação", que conduziu ao

agrupamento dos gêneros semelhantes em **famílias** e das famílias semelhantes em **ordens**.

Além destas categorias taxônomicas criadas por Lineu, as **classes** ou conjunto de ordens semelhantes estão reunidas em **filos** e estes em **reinos**. Por exemplo, o lobo vermelho (espécie *C. rupus*) pertence ao reino *Animalia*, filo *Chordata*, classe *Mammalia*, ordem *Carnivora*, família *Canidae* e gênero *Canis*. O termo **filo** é substituído pelo termo **divisão**, na classificação taxonômica das plantas, de modo que a batata inglesa (espécie *Solanum tuberosum*) pertence ao reino *Plantae*, divisão *Magnoliophyta*, classe *Magnoliopsida*, ordem *Solanales*, família *Solanaceae* e gênero *Solanum*. A Figura 2.1 apresenta, como exemplo, a classificação de alguns animais da ordem *Perissodactyla* da classe dos mamíferos (*Mammalia*).

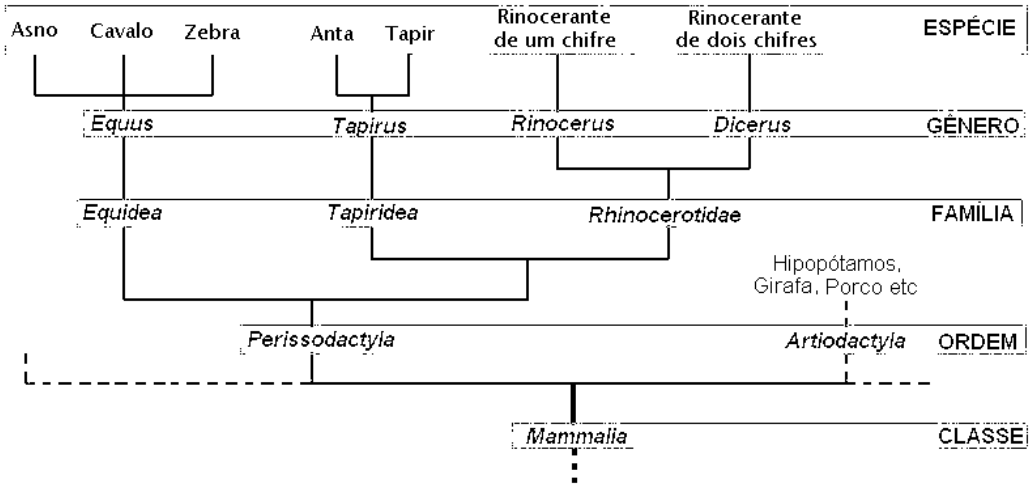


Figura 2.1 Relações de parentesco e classificação de alguns animais da ordem *Perissodactyla* – adaptado da fonte [AM04, p.17]

O ato de dar nomes científicos aos seres vivos é controlado por comissões internacionais de nomenclatura Zoológica (ICZN), Botânica (ICBN), de Bactérias (ICNB) e de Vírus (ICTV)[1]. Para todas as comissões a regra geral é baseada na lei da prioridade: o nome válido é o nome mais antigo conhecido, "único método para a nomenclatura científica não entrar num caos" [Wik06a]. Sob controle dessas comissões foram criadas subclasses, subordens, superclasses, superordens etc; de modo que, por exemplo, a classe **aves** foi dividida em duas subclasses: *Archaeornithes* (aves ancestrais) e *Neornithes* (aves recentes).

[1] ver descrição de cada sigla no Apêndice A, página 175

No início do século XX, os cientistas sentiram a necessidade de separar os seres vivos em novos reinos, de forma que as pesquisas do biólogo francês Edouard Chatton (1883-1947) promoveram a separação das bactérias em um reino exclusivo, chamado *Monera*, visto que, elas apresentam células **procarióticas**, ou seja, sem núcleo nem organelas membranosas, portanto, sem separação física entre o material nuclear e o citoplasma, diferentemente dos demais seres vivos que apresentam células **eucarióticas**. Em 1969, o biólogo norte-americano Robert H. Whittaker (1924-1980) popularizou o sistema de classificação em cinco reinos quando, além dos reinos já citados *Animalia*, *Plantae* e *Monera*, foram criados os reinos *Fungi* que agrupa os fungos e *Protista* que agrupa os protozoários e as algas. Carl Woese, a partir de 1977 sugeriu a divisão do reino Monera em *Eubacteria* e *Archaebacteria* passando para seis reinos e em 1990 uma nova classificação que é, por enquanto, a mais difundida e aceita considera três domínios (*Bacteria, Archae e Eukarya*). Toda a evolução da classificação biológica está mostrada na Tabela 2.1.

Tabela 2.1 Evolução da classificação científica dos seres vivos

<table>
<tr><th>Três reinos</th><th>Cinco reinos</th><th>Seis reinos</th><th>Três domínios</th></tr>
<tr><td rowspan="2">Protista</td><td rowspan="2">Monera</td><td>Eurobacteria</td><td>Bacteria</td></tr>
<tr><td>Archaebacteria</td><td>Archae</td></tr>
<tr><td></td><td>Protista</td><td>Protista</td><td rowspan="4">Eukarya</td></tr>
<tr><td rowspan="2">Plantae</td><td>Fungi</td><td>Fungi</td></tr>
<tr><td>Plantae</td><td>Plantae</td></tr>
<tr><td>Animalia</td><td>Animalia</td><td>Animalia</td></tr>
<tr><td>Ernst Haeckel</td><td>Robert Whittaker</td><td>Carl Woese</td><td>Carl Woese</td></tr>
<tr><td>1894</td><td>1969</td><td>1977</td><td>1990</td></tr>
</table>

Por serem acelulares, os **vírus** não estão incluídos nesta classificação, eles são chamados de parasitas intracelulares pois somente conseguem se reproduzir no interior das células de outros

seres, e são constituídos por uma ou mais moléculas de ácido nucléico (DNA ou RNA) envoltas por moléculas de proteínas [AM04].

Além da Taxonomia, a Sistemática Moderna da classificação biológica estuda a biodiversidade do planeta, ou seja, as relações entre organismos e táxons, sua catalogação e nomenclatura de novas espécies, enquanto que, outra divisão da Sistemática, a Paleontologia, estuda os organismos fósseis [Fut03, PHM99] que são espécies que viveram num tempo anterior ao recente e que têm partes preservadas. Quando descobertos os fósseis são classificados da mesma forma que as espécies atuais, em classe, ordem, gênero etc. Uma das dificuldades de lidar com seus dados é a impossibilidade de se determinar se o fóssil pertence a uma espécie ancestral, extinta ou derivada, ocasionando uma alteração significante de sua posição em relação às demais espécies estudadas.

Outra conotação, baseada na teoria evolucionista de Charles Darwin (1809-1882), descrita na seção seguinte, conduz a duas correntes principais da Sistemática, chamadas de Fenética e Filogenética, atribuições feitas pelo zoólogo alemão Ernst Mayr (1904-2005). A Fenética agrupa os organismos em categorias taxonômicas com base na sua similaridade fenotípica, estimada exclusivamente na análise quantitativa das características, por isso, também é chamada de taxonomia numérica e em geral utiliza métodos baseados em distâncias (ver Seção 4.2); não se apresenta porém com tanta utilidade quando se necessita de níveis menores de classificação, tal como o gênero, onde nesse caso devem ser utilizado métodos mais apurados.

A Filogenética faz com que a classificação biológica reflita o máximo possível as relações de parentesco evolutivo entre os táxons, de modo que, através da utilização da *cladística* (método detalhado na Seção 2.5), estas relações são estabelecidas pela escolha criteriosa daquelas características que indicam realmente a ancestralidade comum dos grupos estudados representadas em forma de árvores. Este tópico, denominado **Árvore Filogenética** ou **Filogenia**, é o enfoque principal deste trabalho. Conceitos, definições, importância, representação, métodos de construção, abordagem do problema, algoritmos propostos, simulações e resultados de testes computacionais estão descritos nas seções e capítulos seguintes com o objetivo de dar conhecimento sobre esse assunto.

2.2 A Evolução da Vida

A evolução dos seres vivos [Dar98, Ste00] tem toda sua teoria baseada nos trabalhos dos naturalistas ingleses Charles Darwin e Alfred Russel Wallace (1823-1913). Pesquisas em diferentes áreas da Biologia têm evidenciado que o processo evolutivo proposto é o responsável pela diversividade, ou seja, as espécies se diversificam originando outras, algumas delas já extintas, pela seleção natural. A adaptação ao meio ambiente, a existência de fósseis e as semelhanças anatômicas, fisiológicas e bioquímicas entre as espécies tornam esta teoria evolucionista evidente.

Quando se acompanha a formação embrionária comum de diversos tipos de vertebrados ou quando se comparam os esqueletos dos membros anteriores de outros como, o braço humano, a nadadeira do golfinho, a asa do morcego e a asa de uma ave, observa-se que mesmo tendo funções distintas eles têm forma anatômica muito semelhante o que dá indícios que estes animais podem descender de um ancestral comum.

As modernas técnicas de análise bioquímica revelam grande semelhança entre a estrutura molecular de diversos organismos. Os genomas e proteomas, definidos a seguir na Seção 2.4, de todas as espécies sejam elas recente, fóssil ou uma nova que venha a ser descoberta, são compostos por estruturas formadas pelos mesmos quatro tipos de nucleotídeos existentes e os mesmos vinte tipos de aminoácidos, respectivamente. De acordo com o evolucionismo as semelhanças e as diferenças entre seqüências de aminoácidos de organismos distintos, os unem ou os separam na classificação biológica. Uma forma de mostrar isto é através das árvores filogenéticas, objeto de nosso estudo. Para exemplificar, a Figura 2.2 mostra as relações filogenéticas entre os reinos dos seres vivos descritos na Seção anterior, constatando a existência de um ancestral comum a todos, como imaginado por Darwin e Wallace [AM04]. Os nós internos, não identificados na figura, representam os ancestrais dos táxons imediatamente descendentes.

Como a ocorrência do processo denominado **especiação** (origem de um nova espécie a partir de ancestrais) não pode ser observada em função do tempo de evolução e a análise das características dos fósseis, em alguns casos, não é possível devido a seu estado de conservação, então, as relações filogenéticas devem ser inferidas, no sentido de que, as comparações en-

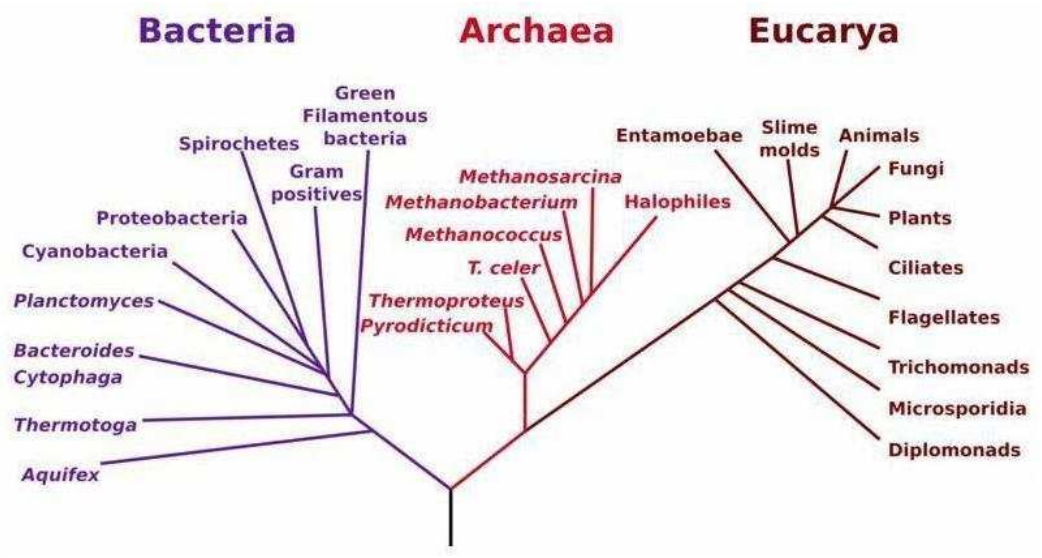


Figura 2.2 Relações filogenéticas entre os três domínios dos seres vivos – retirado da fonte [Wik06d]

tre seqüências e/ou características das espécies em estudo levam a uma estrutura provável de suas origens, tornando então o problema da filogenia bastante complexo em função do grande número de possibilidades de ramificações que representam os relacionamentos.

Mesmo considerando a alta complexidade para inferir uma filogenia, os resultados conseguidos pela análise filogenética são valiosos e importantes pois são aceitos por grande parte dos biólogos que admitem que a formação de novas espécies se dá pela **cladogênese**, também chamada de "especiação por diversificação", cuja idéia é a base da cladística, a ser tratada a seguir na Seção 2.5.

As hipóteses das relações entre os táxons produzem uma árvore evolutiva que representa a história e descreve as características conhecidas. A reconstrução desta história e destas relações combina a lógica da análise filogenética com os dados da evolução molecular e morfológica.

A biologia evolutiva aborda, além da filogenia, a genealogia [Ste00] que é conhecida como "Ciência da História da Família" e tem como objetivo desvendar as origens dos indivíduos por intermédio do levantamento dos relacionamentos interfamiliares de seus antepassados, podendo ser estendido aos descendentes. As árvores filogenéticas diferem das árvores genealógicas, pois nestas o ancestral se encontra na base, indicando que naquele ponto do passado a linhagem era formada apenas por ele. A genealogia, entretanto, não é objeto de nosso estudo, pois conforme

já citamos, todo o contexto deste trabalho tem por objetivo explicar e reconstruir as árvores filogenéticas através de métodos que levem a resultados bastante confiáveis.

Neste texto iremos utilizar indistintamente os termos *espécie* ou *táxon* para referenciar organismos, seres vivos ou qualquer entidade que contenha uma expressão gênica, quer real e identificada, ou fictícia quando gerada de forma aleatória para fins de testes ou experimentos computacionais apresentados nos Capítulos 5, 6 e 7.

2.3 Biologia Molecular

Para um melhor entendimento do estudo da cladística, há necessidade de definir antes os elementos básicos da biologia molecular.

Gregor Mendel (1822–1884), considerado o pai da genética, enunciou suas leis deduzidas a partir de suas experiências com ervilhas. Ele observou que determinadas características da planta eram repassadas para gerações futuras, seguindo regras claras e bem determinadas; por conseguinte, pesquisas posteriores comprovaram que as instruções genéticas são armazenadas dentro das células dos organismos vivos sob a forma de *genes* que contêm as informações que caracterizam as espécies.

Os genes estão nos cromossomos que são componentes moleculares formados pelo DNA (*deoxyribonucleic acid*) e pelas proteínas. Uma molécula de DNA consiste de duas cadeias, ou fitas, ligadas por pontes de hidrogênio. Cada cadeia de DNA é composta por nucleotídeos (ver um exemplo na Figura 2.3-a) que são formados por um tipo de acúcar, chamado *desoxirribose*, um ou mais grupos de fosfato e uma base nitrogenada que pode ser Adenina (A), Timina (T), Citosina (C) ou Guanina (G) [ABJ$^+$99, CRM99].

As proteínas, formadas por aminoácidos, determinam a estrutura da célula, realizam a maioria das funções celulares e regulam a expressão gênica, fazendo com que as células se movam e se "combinem" umas com as outras.

Quando uma proteína em particular é necessitada por uma célula, a seqüência de nucleotídeos é primeiramente copiada (processo de transcrição) para outro tipo de ácido nucléico chamado de RNA (*ribonucleic acid*). Um outro processo, chamado de tradução, usa as informações

contidas no RNA para produzir novas proteínas. Em relação à sua estrutura, no RNA a Timina que está presente no DNA é substituída por outra base nitrogenada, a Uracila (U).

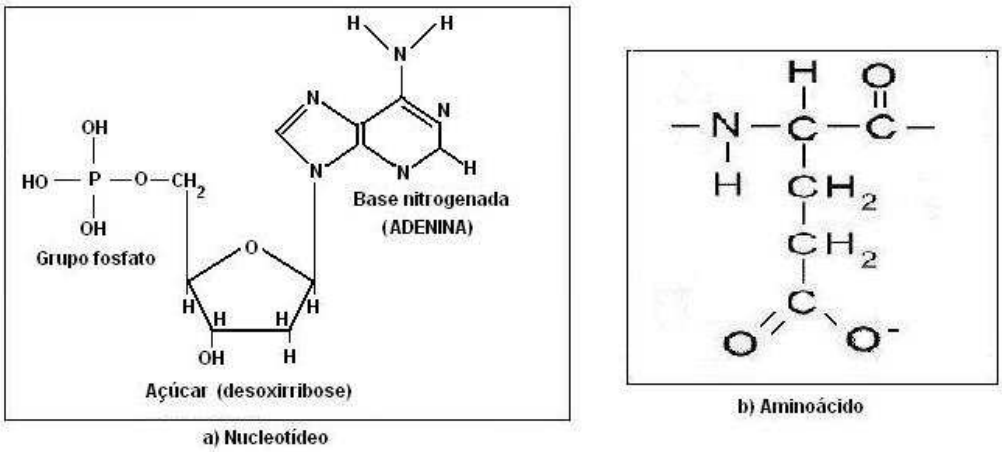


Figura 2.3 a) Estrutura Molecular do Nucleotídeo "Adenina" [AM04, ABJ+99] b) Estrutura Molecular do Aminoácido "Ácido Glutâmico" [Wik06a, ABJ+99]

Os aminoácidos são codificados por uma trinca nucleotídica, ou seja, um arranjo de três nucleotídeos, denominado *códon*. Existem, portanto, 64 (4^3) códons que são formados pelos arranjos "três a três" das quatro bases nitrogenadas (A, C, G e U), porém, apenas vinte tipos diferentes de aminoácidos. Dos 64 códons, três deles (UAG, UGA e UAA) não codificam nenhum aminoácido e são denominados códons finalizadores (*stop codons*). A relação um códon específico para um aminoácido é denominado **código genético**, e todos os organismos vivos usam o mesmo código genético, exceto as mitocôndrias e as bactérias que possuem, para alguns códons, aminoácidos diferentes.

A Tabela 2.2 mostra os vinte aminoácidos naturais encontrados nas proteínas [SM97] e a Figura 2.4 apresenta todos os códons correspondentes.

A Biologia Molecular oferece a prova de que todos os organismos complexos descendem de seres unicelulares, visto que, todos os genes, sejam de bactérias, fungos, plantas e vertebrados são quimicamente iguais [Fut03, Dar98].

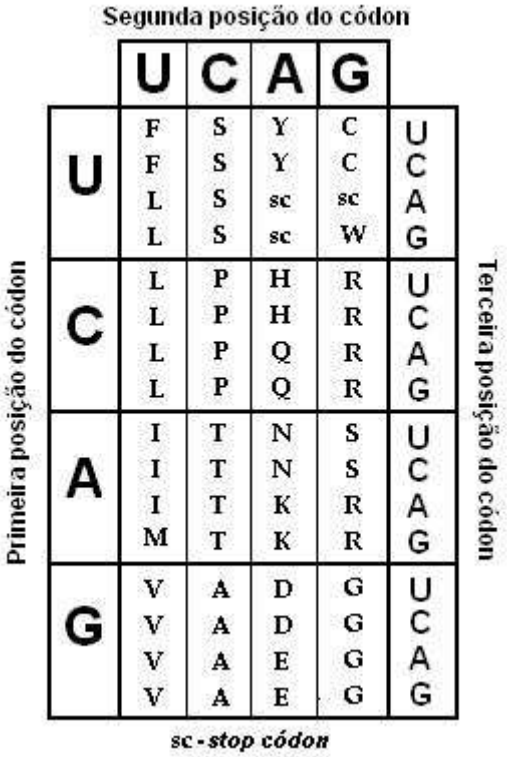


Figura 2.4 Códons e Formação do Código Genético. A codificação de cada letra está representada na Tabela 2.2 e corresponde ao conjunto de três dos nucleotídeos na ordem das posições indicadas. Por exemplo, o códon "AUG" (Adenina + Uracila + Guanina) corresponde à **metionina** (letra "M" na primeira coluna) enquanto que a **serina** (letra "S" na quarta coluna) correspondem aos códons "AGU" e "AGC"(Adenina + Guanina + Citosina)

Tabela 2.2 Os vinte aminoácidos naturais

N.	Letra	Código	Nome	N.	Letra	Código	Nome
1	A	Ala	Alanina	11	M	Met	Metionina
2	C	Cys/Cis	Cisteína	12	N	Asn	Asparagina
3	D	Asp	ácido Aspártico	13	P	Pro	Prolina
4	E	Glu	ácido Glutâmico	14	Q	Gln	Glutamina
5	F	Phe/Fen	Fenilalanina	15	R	Arg	Arginina
6	G	Gly/Gli	Glicina	16	S	Ser	Serina
7	H	His	Histidina	17	T	Thr/The	Treonima
8	I	Ile	Isoleucina	18	V	Val	Valina
9	K	Lys	Lisina	19	W	Trp/Tri	Triptofano
10	L	Leu	Leucina	20	Y	Tyr/Tir	Tirosina

2.4 Genomas e Proteomas

O conjunto de genes, ou seja, o conjunto completo de informações do DNA de um organismo recebe o nome de **genoma** que é a especificação primária de todo ser vivo [SM97]. Proteoma é o conjunto de proteínas expressas por um genoma e, portanto, representa toda a informação genética que está sendo expressa em um determinado momento.

Os genomas, dentre eles o Genoma Humano [NCB06], por serem menos complexos são mais pesquisados do que os proteomas, porquanto o DNA é formado por quatro bases nitrogenadas e as proteínas por até vinte tipos diferentes de aminoácidos. Além disso, o DNA está localizado no núcleo de qualquer célula, o que facilita a sua obtenção e purificação, já muitas proteínas só estão presentes em alguns tipos de células, e somente em certas fases de seu desenvolvimento.

O objetivo principal de um projeto genoma é determinar as seqüências de DNA completas de todos os cromossomos de um organismo. Os seqüenciadores disponíveis em laboratórios de genética são limitados a seqüenciar milhares de bases nitrogenadas enquanto que mesmo os organismos mais simples, como as bactérias, são formados por milhões de bases. Uma maneira de resolver esse problema é “fragmentar o DNA” e depois fazer a “montagem” das

seqüências ou fragmentos obtidos. Existem disponíveis diversos montadores de genomas que são programas que utilizam técnicas para obter o seqüenciamento completo (*Whole Genomic Shotgun Sequencing*)[2] através da sobreposição de fragmentos aleatórios para formar seqüências contínuas. Na Seção 3.2 estão relacionados alguns destes montadores.

2.5 Cladística

Willi Hennig (1913-1976), entomólogo alemão, desenvolveu as idéias que levariam à criação da Cladística, apresentando-as de forma sumária na sua obra *Grundzüge einer Theorie der Phylogenetischen Systematik*, publicada em 1950, precursora de toda a fundamentação da Sistemática Filogenética.

O princípio fundamental da Cladística é que os organismos devem ser classificados de acordo com as suas relações evolutivas através da identificação das características ancestrais "primitivas" e características derivadas "evoluídas". Portanto, a cladística define uma hierarquia estável de parentesco filogenético entre as espécies.

As características ancestrais são próprias dos seres vivos em que todos os elementos de um dado grupo as possuem; por exemplo, a característica "ter quatro membros" é ancestral nos mamíferos, que a herdaram de um outro ancestral comum; no entanto, somente esta informação não teria utilidade para análise dos relacionamentos entre os primatas, pois todos possuem quatro membros, havendo necessidade da busca de outras características que possibilitem um agrupamento mais refinado.

A similaridade resultante da herança de um ancestral comum é chamada de **homologia** que é a ferramenta básica da biologia comparada. Estruturas homólogas, ou homogenéticas, podem ser **idênticas** entre si, como os pés de dois indivíduos de uma mesma espécie embora sem nenhuma relação de parentesco; **semelhantes**, como os braços de um homem e de um chimpanzé; ou **diferentes**, como os bicos de um papagaio e de um beija-flor, onde neste caso, certamente o ancestral comum às duas espécies também tinha bico.

[2]por Frederick Sanger, laureado duas vezes (1958 e 1980) com o prêmio Nobel de Química

Definem-se três métodos básicos [Amo97] de inferência de relações de homologia entre espécies diferentes: primeiro, quando as estruturas homólogas comparadas têm formas parecidas; segundo, quando ocupam a mesma posição relativa a outras estruturas do corpo e, por último, quando elas se formam a partir de células que ocupavam posição similar em estágios embrionários iniciais e fazem parte da mesma seqüência de modificações. De forma que, dentro do paradigma evolutivo, quando se afirma a existência de homologia entre uma estrutura de grupos distintos, está implícita a suposição de que essa estrutura esteve presente na espécie ancestral comum mais recente entre os grupos envolvidos.

Para os biólogos que utilizam a Cladística as características primitivas ou ancestrais são chamadas de **plesiomórficas** e quando uma delas é partilhada por todos os membros de um mesmo grupo, chama-se este compartilhamento de **simplesiomorfia**. Quanto às características derivadas ou avançadas designam-se **apomorfias** e, se pertencem apenas ao grupo a ser estudado, denominam-se **autapomorfias**. Por exemplo, "bípede" é uma característica dos homens, que não é partilhada com os demais primatas (ver Figura 2.5), tratando-se de uma autapomorfia. Se, por outro lado, a característica derivada serve para unir dois grupos, designa-se como sinapomórfica (**sinapomorfia**) como é o caso da característica "perda de cauda" que une o grupo do Homem com o grupo dos Grandes Macacos (gorila, orangotango e chimpanzé) e os distingue do grupo parente mais próximo, o dos Macacos (macaco-aranha, macaco-prego etc) [Wik06a].

Os métodos de análise cladística classificam o estado de cada caráter ancestral como "0" e derivado como "1"; os estados derivados são os mais importantes na análise cladística. Por exemplo, no caso de uma seqüência de DNA, se um nucleotídeo "A" está presente numa posição qualquer de um gene, e uma mutação recente mudou de A para G, então G é o alelo "1" (derivado) que será utilizado na análise cladística e os demais compartilhamentos entre as outras espécies do alelo "0" não são levados em consideração. A Tabela 2.3 apresenta alguns estados ancestrais e derivados contidos nas seqüências de um fragmento do gene DFH (*Drosophila homolog of the Friedreich's ataxia disease gene*) presente em alguns primatas.

Para esta análise assume-se que a espécie separada (orangotango) seja um grupo externo (*outgroup*), ou "grupo de referência", de modo que, nas posições 1-4 destacadas na seqüência

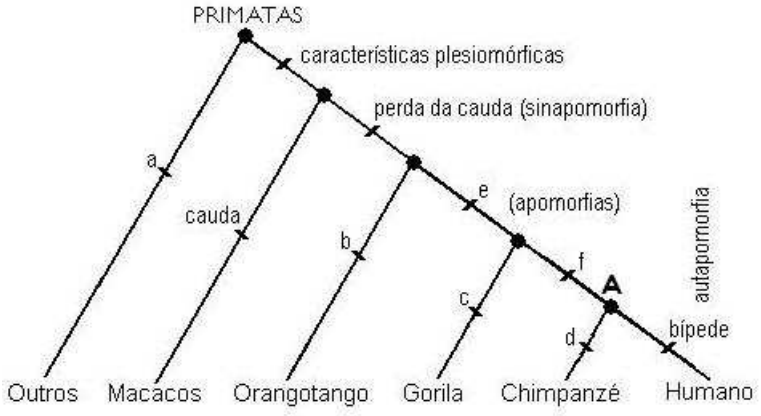


Figura 2.5 Relações homólogas entre alguns primatas. As características a–f não estão identificadas. O nó interno identificado pela letra **A** seria o ancestral comum do homem/chimpanzé.

Tabela 2.3 Seqüência de nucleotídeos de um segmento do gene DFH (*Drosophila homolog of the Friedreich's ataxia disease gene*) presente em alguns primatas

Espécie/Posição	**X**	1	**Y**	2	**Z**	3	**W**	4
Orangotango	AGGA	C	AGAGTGTAG	T	AGTAAAAGGACC	G	CG	T
Gorila	AGGA	T	AGAGTGTAG	T	AGTAAAAGGACC	G	CG	C
Chimpanzé	AGGA	T	AGAGTGTAG	C	AGTAAAAGGACC	G	CG	T
Homem	AGGA	T	AGAGTGTAG	C	AGTAAAAGGACC	A	CG	T

nucleotídica, provavelmente ele possui o estado ancestral. Assim, os outros estados derivados, presentes nas outras espécies, são decididos através da comparação com os nucleotídeos do orangotango. Diz-se, então, que estas espécies analisadas formam um grupo interno (*ingroup*), ou "grupo a ser classificado".

Chama-se **Cladograma** ao diagrama ramificado, ou árvore resultante da análise cladística no qual somente os estados derivados são filogeneticamente informativos. Diferentemente das árvores filogenéticas, os cladogramas não fornecem indicações sobre qual espécie é ancestral de outra nem refletem as dimensões temporais ("distâncias") entre elas.

A Figura 2.6 mostra um cladograma inclinado correspondente às informações contidas na Tabela 2.3.

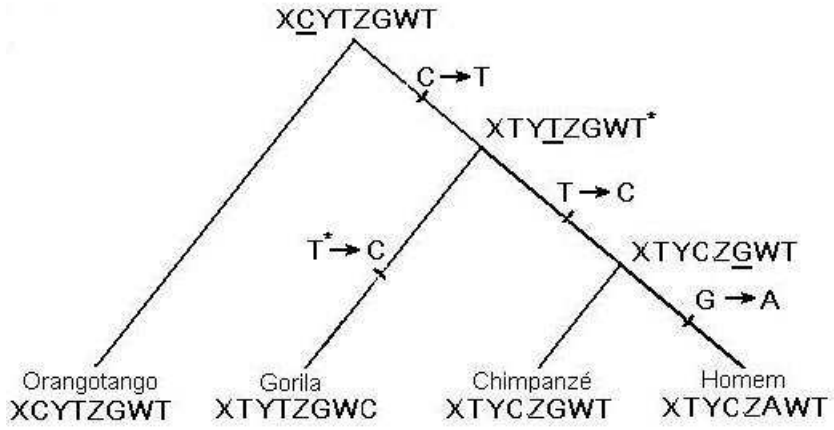


Figura 2.6 Cladograma correspondente às mudanças de estado neste segmento de nucleotídeos do gene DFH em alguns primatas

Aqui neste trabalho, tratamos indiferentemente cladogramas e árvores filogenéticas pois os grupos analisados nos testes computacionais não envolvem as espécies fósseis e não consideramos a necessidade de identificar seus ancestrais. A Figura 2.7 apresenta uma correspondência entre estes dois diagramas, sendo que, a espécie **F** foi incluída como uma identificacão hipotética do ancestral de **A** e **B**.

Um critério, aceito por muitos biólogos, para escolha da Filogenia é baseado no número de mudanças de estados que é chamado de "parcimônia da Árvore Filogenética". Seguindo este

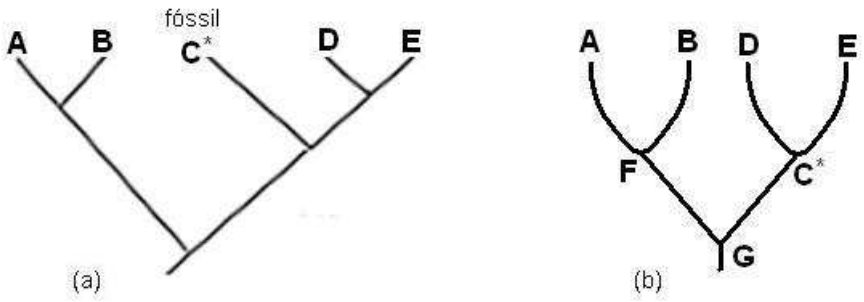


Figura 2.7 a) Cladograma com quatro espécies recentes e um fóssil b) Possível árvore filogenética correspondente – adaptado da fonte [Amo97, p.64]

critério, podemos dizer que dentre as várias árvores possíveis que representam uma filogenia, a melhor delas será aquela que requer a menor quantidade de mudanças dos estados, ou seja, a mais *parcimoniosa* [DJS86, SO90]. Para um melhor entendimento deste conceito citamos a popularmente conhecida **Lei da Parcimônia** que diz: "*quando houver mais de uma explicação válida para um fato, a mais simples em geral é a certa*"[3]

2.6 Sistemática Filogenética

O objetivo da Sistemática Filogenética é entender as relações entre as espécies para inferir a história da vida desde sua origem, representando-a numa estrutura de árvore, chamada Filogenia ou Árvore Filogenética. Diz-se então que a Sistemática infere de forma confiável as relações históricas a partir de similaridades.

Como "as aparências enganam" os cientistas são cuidadosos no tratamento da **homoplasia**, termo que se refere a todos os tipos de similaridades não hereditárias e que aparecem devido

[3]Na realidade, essa "lei" tem origem do "princípio da economia", conhecido como "navalha de Occam", assim chamado em homenagem ao filósofo Guilherme de Occam (1285–1349), monge medieval inglês que defendia com entusiasmo "as virtudes da simplicidade". Uma formulação da "navalha" que agrada aos cientistas atuais seria "quando duas teorias concorrentes podem ser ambas adequadas para explicar um dado fenômeno, deve-se preferir a mais simples" – retirado do livro " *Aprendendo a Filosofar...*", edição em português do original *Zeno and the Tortoise (How to think like a Philosopher)* de Nicolas Fearn [Fea04].

às mutações aleatórias; ou seja, a homoplasia refere-se aos casos de semelhanças entre estruturas de espécies distintas, presentes em cada uma delas devido a ocorrência independente de modificações que resultaram em formas finais semelhantes.

Por exemplo, a Sistemática Molecular, através da análise realizada com auxílio de seqüências de DNA, permitiu concluir recentemente que os pandas, gigante e pequeno, mamíferos que habitam na Ásia, têm origens distintas, relacionando-se respectivamente com ursos e canídeos [SH03, p.233].

Duas razões comuns para aparências enganosas são a **convergência**, que corresponde às estruturas que se parecem mas têm ancestral diferentes, e a **divergência** que são as estruturas diferentes que têm o mesmo ancestral. Como exemplo de convergência, Stearns & Hoekstra [SH03] citam os cactos da América do Sul, da família *Cactacea*, e da África, da família *Euphorbiacea*, portanto, de origem diferentes; eles evoluiram morfologicamente parecidos para se adaptarem a condições ecológicas similares. Em outros grupos, flores e frutas convergiram porque plantas de ancestrais diferentes foram polinizadas por insetos ou pássaros similares. Como exemplo de divergência temos as lobélias do Havaí, plantas do gênero *Cyanea* que possuem 55 espécies com diferentes alturas (de 1 a 14 metros) e folhas de tamanho variados, embora todas descendam de um mesmo ancestral.

Inicialmente, as árvores filogenéticas eram construídas utilizando-se apenas métodos intuitivos com base somente no fenótipo das espécies, o que resultava em conclusões erradas, pois conforme vimos, nada impede que dois organismos que se originam de ancestrais distintos possam ter desenvolvido características semelhantes. O **fenótipo** corresponde às características morfológicas, fisiológicas ou comportamentais apresentadas por um indivíduo, que podem sofrer transformações com o passar do tempo devido, em alguns casos, a fatores ambientais.

Atualmente a construção de filogenias se dá por meio de métodos matemáticos e pela análise do **genótipo** ou constituição genética do indivíduo resultando em árvores mais condizentes com a realidade.

Os **sistematas**, como são chamados os cientistas que estudam a sistemática, utilizam métodos para inferir as relações evolutivas entre os organismos para poder reconstruir filogenias que, conforme já mencionado, são hipóteses, pois é impossível recuperar e ordenar todo o

conhecimento sobre a diversidade biológica.

Quando se discute a relação de parentesco entre as espécies, o conceito mais básico em termos de evolução, é que, para quaisquer duas espécies é necessário formular uma hipótese de que existiu ao menos um ancestral comum a ambas. Já no caso de três espécies a hipótese feita é que há um ancestral comum a duas delas que não é comum à terceira. Ampliando esse raciocínio de forma a abranger todas as espécies, obtém-se uma espécie ancestral de todas as outras que sofreram divisões subseqüentes até a geração das espécies atuais. Claro que existe apenas uma história real de parentesco entre as espécies e é esta história que a filogenia busca.

Portanto, uma filogenia é a história da descendência de um grupo de organismos de um ancestral comum. Árvores filogenéticas são hipóteses sobre os relacionamentos evolutivos e mostram a ordem em que as espécies se separaram. Uma árvore pode retratar a evolução de toda a vida, de todas as grandes linhagens evolutivas ou de somente um pequeno grupo de organismos.

Para se construir a Filogenia de um determinado grupo é necessário considerar as seguintes etapas:

- determinar o grupo monofilético focal, ou seja, o grupo dos organismos cuja filogenia se quer determinar;
- escolher as características que serão usadas na análise filogenética e identificar suas possíveis formas;
- determinar as características ancestrais e derivadas e
- distinguir entre características homólogas e homoplásicas.

Uma das maiores dificuldades é distinguir características ancestrais e derivadas porque algumas características são tão diferentes de seus estados ancestrais que se tornam irreconhecíveis; apesar do que, o que diferencia uma característica da outra é apenas um caráter; por isso, esse processo de escolha é chamado de “polarização do caráter”.

Com base na teoria da evolução, sabe-se que uma característica ancestral é encontrada não apenas entre as espécies do grupo focal, mas também nos grupos externos. Isso ajuda a

distinguir características ancestrais de características derivadas, já que estas são encontradas apenas no grupo focal. Um grupo externo é uma linhagem que está estreitamente relacionada ao grupo focal, mas que derivou de um grupo focal anterior até a sua base na árvore evolutiva [PSOH03], a lampréia mostrada na Figura 2.8 é um exemplo de grupo externo.

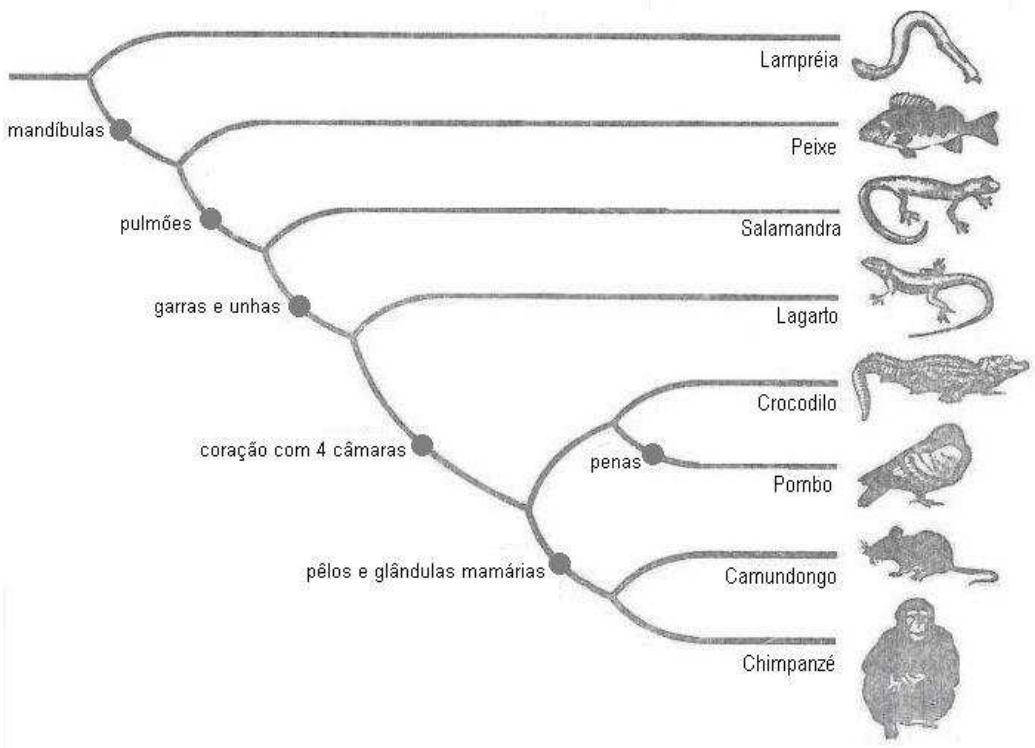


Figura 2.8 Uma filogenia provável entre oito vertebrados – adaptado da fonte [PSOH03, p.429]

Sabemos que a **homologia** refere-se à relação entre estruturas idênticas presentes em espécies distintas, devido à herança dessa estrutura da espécie ancestral comum mais recente, enquanto que, a **homoplasia** é a relação de semelhança entre estruturas de espécies distintas, devido à ocorrência independente de modificações que resultaram numa forma final semelhante.

Quanto ao fato das características serem homólogas ou homoplásicas, a melhor maneira de se determinar essa condição é assumir que as características são homólogas até que alguma evidência prove o contrário. Assim, quanto mais características são estudadas, mais facilmente os biólogos podem distingir entre homoplasias e homologias. Observando a Figura 2.8, é possível notar que a filogenia não descreve os ancestrais e nem coloca uma data nas ramificações

entre as linhagens. Ela mostra apenas a ordem seqüencial das ramificações, onde, as mais antigas estão na esquerda ou acima e as mais recentes na direita ou abaixo.

2.7 Processos de Evolução

Conforme comentado na Seção 2.5, Hennig considera que os caracteres se transformam ao longo do tempo, ou seja, um caráter pode se apresentar de várias maneiras em diferentes ramos evolutivos; são os chamados "estados do caráter" e, em geral, não se sabe o sentido em que se deu a transformação ($A \rightarrow B$, ou $B \rightarrow A$), porém, se isso for possível, diz-se que esse caráter é **polarizado**, indicando que é conhecido seu estado primitivo ou plesiomórfico e seu estado derivado ou apomórfico. Quando existem mais de dois estados, os caracteres são chamados **ordenados** quando por exemplo a transformação $A \rightarrow C$ requer a passagem pelo estado B; se isto não ocorrer sempre, eles são ditos **não-ordenados**.

Para determinar a condição do estado do caráter, Hennig considerou ser fundamental verificar como o caráter no grupo que está sendo estudado (grupo interno) se manifesta nos táxons proximamente relacionados (grupos externos), sendo a decisão baseada no princípio da **parcimônia**, ou seja, utiliza-se a hipótese que requer menos passos evolutivos.

Uma vez que os estados dos caracteres estejam polarizados, a matriz construída mostra a distribuição dos estados dos caracteres nos diferentes táxons (ver modelo na Tabela 2.4), podendo estes serem agrupados exclusivamente com base no compartilhamento de estados apomórficos (derivados). Estes grupos, baseados em sinapomorfias, são **monofiléticos** e podem ser utilizados na classificação denominada "Filogenia Perfeita" (ver Seção 4.4). Grupos não-monofiléticos, ou **merofiléticos**, geram incongruências consideradas homoplasias e não podem ser utilizados diretamente numa classificação filogenética. Neste caso, a hipótese mais parcimoniosa deve ser aceita e as demais rejeitadas.

A Figura 2.9 mostra um caso em que a árvore filogenética deve ser inferida a partir da matriz de caracteres da Tabela 2.4, pois o caráter "7" é incongruente em relação às espécies **A** e **C**. O Algoritmo 3 identifica essas incongruências. Um exemplo é mostrado na Seção 6.3.

Tabela 2.4 Matriz com oito características para três espécies

	CARACTERÍSTICAS							
ESPÉCIES	1	2	3	4	5	6	7	8
A	1	0	0	0	1	0	1	0
B	1	1	0	1	0	0	0	1
C	1	1	1	0	0	1	1	1

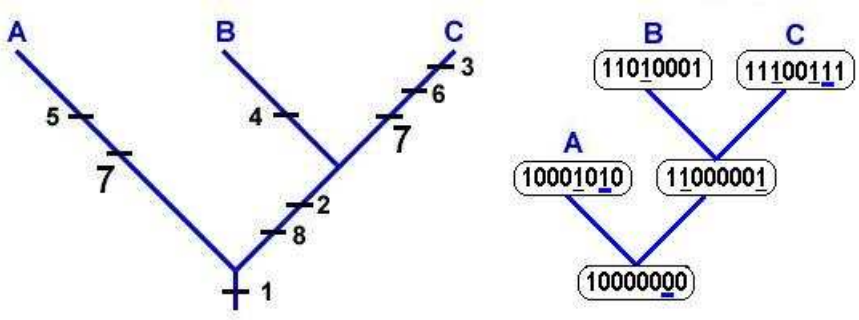


Figura 2.9 Caso de Filogenia Inferida através de uma convergência

Homoplasias ocorrem quando o estado derivado se origina, a partir do primitivo, mais de uma vez, independentemente (caso de **convergência**), ou quando o estado derivado sofre nova modificação, para uma situação semelhante ao estado anterior (caso de **reversão**).

O caráter "7" na Figura 2.9 é um exemplo de convergência, considerada uma imcompatibilidade, em relação à filogenia perfeita, porque está presente em duas sub-árvores distintas. Se a espécie **B** tivesse perdido o caráter "2", teríamos um caso de reversão ou transformação do tipo **0→1→0**.

2.8 A Importância do Estudo da Filogenia

As árvores filogenéticas contêm informações úteis na investigação científica em uma grande variedade de questões biológicas. Uma das áreas que mais avançou com a disponibilidade da filogenia foi a Biogeografia que é o ramo da biologia que estuda a distribuição dos táxons nas regiões e os fatores que determinaram essa distribuição. Através da Filogenia pode-se reconstruir a evolução biogeográfica tanto de ordem geral como de grupos taxonômicos particulares, sendo que, a precisão e o refinamento deste processo estão diretamente ligados ao rigor das reconstruções filogenéticas dos grupos estudados [Amo97].

Técnicas filogenéticas também são usadas pela biologia molecular para compreender a origem e diferenciação de macromoléculas em grandes grupos, através do uso de técnicas de seqüenciamento de proteínas e ácidos nucléicos.

Árvores filogenéticas auxiliam na resolução de problemas práticos tais como o controle e combate de parasitas[4] responsáveis por doenças. Como a análise filogenética ordena as informações disponíveis sobre caracteres compartilhados é possível obter uma ação farmacológica e imunológica de alcance mais amplo. A justificativa para este alcance é que podem ser determinados quais parasitas pertencem ao grupo **monofilético**, ou seja, ao conjunto de espécies que inclui um ancestral e suas espécies descendentes com características enzimáticas comuns, podendo portanto, ser alvo de uma única ação farmacológica.

[4] vírus, protozoários, fungos e metazoários

Outra utilidade da filogenia é que através da ordenação dos caracteres para se determinar o nível evolutivo do surgimento de enzimas e mecanismos de controle encontrados no ser humano é possível que testes de reação imunológica ou de toxicidade sejam feitos em outros sistemas biológicos similares antes de sua aplicação no homem. Da mesma forma a informação filogenética também pode fornecer subsídios importantes para decisões relativas a transplantes de órgãos ou tecidos de outras espécies para o homem [Amo97].

Em geral, o conhecimento biológico pode ser expresso sobre o enfoque filogenético, de modo que a geração de árvores filogenéticas contribui imensamente para inferir a história evolutiva das espécies.

Ao final deste capítulo, esperamos ter comentado os tópicos relevantes para o entendimento da Biologia Molecular em geral e da Sistemática Filogenética em particular. No capítulo seguinte daremos uma abordagem da aplicação destes conceitos no âmbito da computação, tanto para o problema da filogenia como para o problema do alinhamento de seqüências entre outras, além de mostrar que muitas destas informações estão disponibilizadas em bancos de dados públicos.

CAPÍTULO 3

Biologia Computacional

3.1 Introdução à Bioinformática

A Bioinformática e a Biologia Computacional envolvem o uso de técnicas computacionais e conceitos da Matemática, da Estatística e da Informática para resolver problemas biológicos [Wik06b].

Os dois termos são usados freqüentemente sem distinção em função da linha comum de projetos de pesquisa e desenvolvimento de ferramentas de atualização e recuperação de informações em banco de dados públicos. A Biologia Computacional é mais abrangente pois busca técnicas algorítmicas e métodos computacionais para criar estas ferramentas. A utilização em laboratório de equipamentos como seqüenciadores automáticos de DNA e de Proteínas, analisadores de expressões gênicas e sintetizadores de ácidos nucléicos, entre outros, pertencem ao campo da Bionformática.

Devido ao avanço da genética molecular e genômica, a medicina molecular e a biotecnologia constituem duas áreas prioritárias do desenvolvimento científico e tecnológico estreitamente relacionados com a Bioinformática e Biologia Computacional. Aplicações, como alinhamento de seqüências, projetos genomas, reconhecimento de padrões em seqüências depositadas em bancos de dados públicos e construção de árvores filogenéticas exigem muitos recursos computacionais. As pesquisas desenvolvidas nestas áreas têm por objetivo criar técnicas eficientes para estas aplicações.

Um problema típico da Bioinformática refere-se ao estudo dos genomas e proteomas onde até hoje já foram analisadas, pesquisadas e armazenadas seqüências de nucleotídeos e de aminoácidos de diversas espécies e outras são analisadas continuamente em laboratórios de grandes centros de pesquisa.

Os Bancos de Dados públicos são fontes inesgotáveis de pesquisa; o uso da *Internet* vem assumindo uma importância cada vez maior. O seqüenciamento de DNA de diversas espécies, em larga escala, a partir da década de 1990, induziu à construção de bancos de dados robustos para abrigar a explosão no número de seqüências obtidas pelos pesquisadores. Esses bancos de dados funcionam como repositórios, permitindo a recuperação rápida de qualquer consulta. As informações contidas são, além da descrição e classificação da espécie, a seqüência propriamente dita, comentários gerais e literatura associada. Alguns bancos se especializam em um tipo particular de organismo ou célula, outros em funções biológicas particulares, onde são registradas as mutações e diferenças para um dado gene ou um conjunto de genes.

Relacionamos a seguir os principais bancos de dados públicos com informações da Bioinformática.

• NCBI (*National Center for Biotechnology Information*), criado em 1988 pelo NIH (*National Institute of Health*) possui dados que conduzem as grandes pesquisas em Biologia Computacional permitindo ligações para todas outras bases de dados (*All Databases*), mantém atualizadas informações sobre os genomas conhecidos, desenvolve ferramentas específicas para alinhamentos (*BLAST* – ver seção seguinte), dissemina informações sobre a biomedicina (*link PubMed*), através de citações e resumos de grande parte da literatura da área. Disponibiliza ainda uma ferramenta que mostra a visualização de diversos componentes moleculares e possui um completo detalhamento sobre taxonomia, permitindo busca por nome da espécie, código de identificação, símbolos etc. Seguramente, podemos afirmar que a página do NCBI deve ser a primeira a ser visitada por aqueles que desejam pesquisar sobre assuntos da área de biologia molecular [NCB06].

• EMBL (*European Molecular Biology Laboratory*) ou Laboratório Europeu de Biologia Molecular foi criado em 1974 pelos estados constituintes da Conferência Européia de Biologia Molecular (EMBC) com o objetivo de promover a cooperação entre os Estados nas pesquisas em Biologia Molecular com ênfase em análises experimentais em múltiplos níveis da organização biológica, da molécula ao organismo, assim como na Biologia Computacional e Bioinformática. Faz parte da Colaboração Internacional dos Bancos de Dados de Seqüências de Nucleotídeos [EMB06].

• EBI (*European Bioinformatics Institute*) ou Instituto de Bioinformática Europeu é um centro de pesquisas e serviços na área de bioinformática, que faz parte do EMBL, trabalha com uma base de dados que inclui ácidos nucléicos, seqüências de proteínas e estruturas macromoleculares [EBI06].

• GenBank (*Genetic Sequence Database*) ou Banco de Dados de Seqüências Genéticas é um banco de dados de seqüências do NIH (*National Institutes of Health*) que armazena todas as seqüências de DNA publicamente disponíveis. O GenBank [Gen06] faz parte do "*International Nucleotide Sequence Database Collaboration*", que inclui também o DDBJ (DNA Data Bank of Japan) ou Banco de Dados Japonês de DNA e o EMBL. Estas três organizações compartilham os seus dados diariamente, e uma nova atualização do GenBank é feita a cada dois meses. Cada entrada no GenBank inclui uma descrição concisa da seqüência, o nome científico e a taxonomia do organismo fonte e uma tabela de caracteríscas que identifica regiões codificadas da seqüência e outros locais de significância biológica, assim como unidades de transcrição, locais de repetições e mutações.

• PDB (*Protein Data Bank*), criado nos Estados Unidos em 1993, é o mais importante banco de dados de proteínas. Ele funciona como um repositório de processamento e distribuição de dados. Existem, atualmente [PDB06], mais de 20.000 estruturas de proteínas disponíveis no banco de dados PDB e este número aumenta a cada ano. O PDB faz parte da Pesquisa Colaborativa para o Consórcio de Bioinformática Estrutural (RCSB) mantida pela Universidade Rutgers de Nova Jersey, pelo Centro de Supercomputadores de San Diego (SDSC) e pelo Instituto Nacional de Padrões e Tecnologia .

• DDBJ (*DNA Data Bank of Japan*) ou Banco de Dados Japonês de DNA, mantido pelo Instituto Nacional de Genética (NIG) no Japão, é também um dos mais importantes bancos de dados de seqüências de DNA. Faz parte da Colaboração Internacional dos Bancos de Dados de Seqüências de Nucleotídeos [DDB06].

3.2 Alinhamento de Seqüências

3.2.1 Comparação de Seqüências

Comparar seqüências de caracteres é um problema de interesse da Ciência da Computação [Gus97] com aplicações nos processadores de texto, gerenciadores de banco de dados e em diversas ferramentas da Biologia Computacional, especialemente no estudo da evolução e de funções da microbiologia em que é comum a necessidade de comparar moléculas de várias espécies.

Uma das formas de comparar duas ou mais seqüências é através da construção de um alinhamento entre elas [MS95]. No contexto da Biologia, seqüências aparecem nas estruturas primitivas que indicam como os aminoácidos se encontram combinados em um gene ou em uma proteína. Um alinhamento procura determinar o grau de similaridade entre as seqüências comparadas, na sua totalidade ou entre fragmentos das mesmas. Os resultados destes alinhamentos são úteis para a análise de regiões conservadas dos genes ou que sofreram mutações, bem como, no estudo das estruturas secundárias de proteínas e na construção de árvores filogenéticas [FD87].

Algoritmos básicos para fazer alinhamento de seqüências [BLP97, KTN04] buscam sempre sua melhor forma, ou seja, aquela que corresponda ao maior grau de similaridade, ou a "melhor" solução entre as viáveis. A idéia central é minimizar as diferenças entre as seqüências comparadas após os deslocamentos. Smith e Waterman [SW81] idealizaram um algoritmo utilizando técnicas de programação dinâmica para obter o chamado alinhamento "ótimo".

Os alinhamentos podem ser simples (entre duas seqüências) ou múltiplos (entre três ou mais seqüências). Os processos de alinhamentos consistem em introduzir espaços, chamados *gaps*, representados aqui pelo símbolo "–", que correspondem aos deslocamentos de segmentos a fim de que a maioria dos caracteres sejam idênticos em algumas posições. Esses espaços, ou traços, podem ser inseridos nas extremidades ou no interior das seqüências de modo que numa mesma posição os símbolos das duas seqüências não sejam simultaneamente iguais a "–" (alinhamento livre de colunas em branco).

A Tabela 3.1 contém um exemplo de alinhamento que possui três segmentos, delineados por linhas verticais simples. Observa-se que a primeira seqüência é menor que as outras duas, havendo necessidade de deslocar os caracteres para que um alinhamento seja possível. O objetivo do exemplo é fazer alinhamentos simples (comparação duas a duas).

Tabela 3.1 Alinhamento de três seqüências de aminoácidos com três segmentos. Caracteres sublinhados apresentam as diferenças entre as seqüências e traços (–) indicam espaços provenientes de deslocamentos

Seq.	Caracteres																			Tamanho
r	M	R	L	T	L	L	C	C	–	–	–	E	G	E	E	G	S	P	V	L	17
s	M	R	L	T	L	L	C	C	C	T	W	M	G	E	E	G	S	P	V	C	20
t	M	R	L	T	C	L	C	C	R	T	W	M	G	E	E	G	S	P	V	C	20

Em geral, é computado um escore para cada resultado da comparação durante o procedimento de alinhamento, sendo que o maior valor desse escore determinará o grau de similaridade entre as seqüências comparadas. Pode-se por exemplo, atribuir um valor positivo $(+1)$ para cada par de caracteres idênticos (*match*) e valores negativos nos casos contrários (*mismatch*). Essa penalização poderia ser (-1) para cada par de caracteres divergentes e (-2) para o par que contiver um espaço em uma das duas seqüências.

Utilizando este critério de pontuação para o exemplo da Tabela 3.1 computam-se os escores, através da função $G(x,y)$ que indica o grau de similaridade entre as seqüências **x** e **y**. Os resultados são mostrados na Tabela 3.2.

$$\begin{aligned} G(r,s) &= 15(+1)+2(-1)+3(-2)=7 \\ G(r,t) &= 14(+1)+3(-1)+3(-2)=5 \\ G(s,t) &= 18(+1)+2(-1)+0(-2)=16 \\ G_{max} &= 20(+1)+0(-1)+0(-2)=20 \end{aligned} \tag{3.1}$$

Verifica-se que o grau de alinhamento máximo (G_{max}) é igual ao tamanho da maior seqüência (no caso igual a 20 caracteres). Os resultados contidos na Tabela 3.2 permitem concluir

Tabela 3.2 Cálculo do escore do alinhamento entre as seqüências da Tabela 3.1 com indicação do total de caracteres iguais, diferentes e *gaps* (representados por traços). O primeiro índice refere-se ao escore obtido em relação ao escore máximo possível; no segundo, o "% de semelhanças" não considera os deslocamentos e refere-se ao número de caracteres iguais em relação ao tamanho da menor das duas seqüências comparadas

Seqüências	Iguais	Dif.	*Gaps*	Escore	% (pelo escore)	% semelhanças
r ‖ **s**	15	2	3	7	7/20 = 35%	15/17 = 88,2%
r ‖ **t**	14	3	3	5	5/20 = 25%	14/17 = 82,3%
s ‖ **t**	18	2	0	16	16/20 = 80%	18/20 = 90,0%

que o maior escore (igual a 16) indica que as seqüências **s** e **t** são mais similares do que as demais relações, correspondendo a uma "distância" evolutiva menor. Uma matriz de "distâncias" poderia ser gerada para mostrar estas relações; a mesma conclusão seria obtida se fosse considerado o "% de semelhanças". Na Seção 4.2, é apresentado outro exemplo juntamente com a matriz de distâncias gerada.

Vale ressaltar que, um alto escore de alinhamento, apesar de ser uma boa indicação de grande similaridade entre as seqüências, não implica que existe uma homologia entre elas, pois esta é uma hipótese evolutiva e não possui gradação. Como já definimos anteriormente, duas seqüências são homólogas se têm um ancestral comum, independente de sua similaridade.

3.2.2 Tipos de Alinhamento

Um dos tipos mais comuns do alinhamento simples é o chamado **global** e recebe essa denominação porque as seqüências envolvidas são consideradas de uma extemidade a outra. Assim, após a inclusão dos espaços tomam-se as seqüências a serem alinhadas e coloca-se "uma sobre a outra" de forma que um caráter da primeira alinhe-se a um outro, ou espaço, da segunda, e vice-versa. Exemplos de programas que utilizam este alinhamento são "ClustalW" e "Multialin" comentados a seguir. As Tabelas 3.1 e 3.3 são exemplos de alinhamento global.

No alinhamento simples **local**, o propósito é encontrar e extrair um ou mais pares de regiões de cada uma das duas seqüências, que exibam alta similaridade. No caso a avaliação

Tabela 3.3 Exemplo de alinhamento simples global

G	A	A	–	G	G	A	T	T	A	G
G	A	A	C	G	G	A	–	–	A	G

seria feita de forma "parcial" (somente dos segmentos selecionados). Por exemplo, dadas duas seqüências **s** e **t**, encontrar subseqüências **a** e **b** de **s** e **t** respectivamente, cuja similaridade seja máxima entre estes pares. Um exemplo desse tipo de alinhamento é mostrado na Tabela 3.4 onde as seqüências **a** e **b** são iguais, no caso, com grau máximo igual a três. O programa BLAST [AMS+97], descrito a seguir é um exemplo de uma ferramenta que realiza esse tipo de alinhamento.

Tabela 3.4 Exemplo de alinhamento simples local, com indicação das subseqüências **a** e **b** das seqüências **s** e **t** respectivamente

		a	
s		AAG	ACGG
t	GATC	AAG	
		b	

Conforme já citado, o alinhamento **múltiplo** é usado quando se está interessado em comparar três ou mais seqüências e a utilização de alinhamentos simples não é satisfatória [MS95]. Técnicas semelhantes às anteriores são estendidas para realizar este tipo de alinhamento. O alinhamento global é frequentemente usado para determinar regiões conservadas entre seqüências homólogas, enquanto que o alinhamento local é geralmente usado na busca de segmentos homólogos em Banco de Dados (BD); um argumento (segmento) é comparado com parte das seqüência disponíveis nos BDs, outra utilização é na montagem de genomas.

3.2.3 Ferramentas de Alinhamento

A seguir citamos as ferramentas de alinhamento mais comuns:

• BLAST (*Basic Local Alignment Search Tool*) ou Ferramenta Básica de Busca de Alinhamento Local é uma das ferramentas para análise de alinhamento mais usada em pesquisa de bancos de seqüências. O programa compara seqüências de nucleotídeos ou proteínas com seqüências existentes num banco de dados específico através de uma heurística que maximiza uma função que calcula o "grau de similaridade". O BLAST pode ser usado tanto para inferir relações funcionais e evolucionárias entre as seqüências, como para identificar táxons [BLA06]. O tempo de execução deste programa é proporcional aos comprimentos das seqüências contidas no banco de dados consultado.

• FASTA (*Fast Alignment*) ou Alinhamento Rápido, da mesma forma que o Blast, compara uma seqüência de proteínas ou de DNA com outra seqüência correspondente contida num banco de dados de proteínas ou de DNA utilizando outra heurística; seu formato é bastante difundido [FAS06, Wik06c].

• CLUSTALW é um programa que tem como propósito geral fazer alinhamento múltiplo de seqüências de DNA ou proteínas. Ele calcula e realiza o melhor emparelhamento das seqüências selecionadas, alinhando-as de forma que suas identidades, semelhanças e diferenças possam ser notadas [CLU06].

• MUMMER é um pacote de código fonte aberto, que realiza alinhamento rápido de seqüências de DNA e proteínas em larga escala. A grande vantagem de seu uso é a rapidez e uso de poucos recursos computacionais com que o alinhamento é obtido. Sua eficiência, em comparação com outras ferramentas de alinhamento, é notada quando as seqüências são muito grandes, [MUM06].

• MULTIALIN é um programa para ser utilizado quando se deseja fazer um alinhamento múltiplo de um grupo de seqüências relacionadas. Para se obter boas soluções seu tempo de execução é bastante elevado [Mul06]. A heurística utiliza como critério de parada um limite do número de iterações sem melhoria.

Outros problemas relacionados com seqüências referem-se ao Problema da Seqüência Mais Próxima [VGMP04], descrito a seguir, e à Montagem de Fragmentos (comentados na Seção 2.4), onde as ferramentas mais conhecidas são: Cross-Match e BLAST2 Sequence [BLA06], PHRAP (*Phragment Assembly Program*) [Gre06] e TIGR Assembler [Sut95, Adi00].

3.3 O Problema da Seqüência Mais Próxima

Em algumas pesquisas na área de Biologia Molecular há necessidade de comparar e encontrar propriedades nas seqüências de caracteres (*strings*). Obter fragmentos comuns, ou que sejam próximos, a um conjunto de seqüências de DNA, RNA e proteínas tem aplicações importantes como identificação de drogas genéticas, busca de região conservadas em seqüências não alinhadas etc.

Assim, quando se dispõe de um conjunto S de *strings* de tamanho fixo e se deseja encontrar uma outra *string* x de mesmo tamanho, tal que x seja a mais "parecida" com todas as sequências de S, deparamos com o chamado problema da seqüência mais próxima ou CSP (*Closest String Problem*) [GNR01, LMW02, MLOP04]. No caso, x, em geral não pertencente a S, seria uma "representante" de S e deve ser tal que a maior quantidade de caracteres distintos entre x e qualquer elemento de S seja mínima.

Considerando duas *strings* s e t de mesmo tamanho ($|s| = |t|$), denotamos por $d_H(s,t)$ a "distância de Hamming" entre elas, que é definida pelo número de posições em que os caracteres são distintos. Por exemplo, se s ="ACTGACT"e t ="CCAGTCT", então $d_H(s,t) = 3$.

Desta forma, podemos definir [VGMP] o Problema da Seqüência mais Próxima (PSMP) assim :

- Dados:

  $\mathscr{A} = \{a_1, a_2, \ldots\}$: alfabeto de caracteres

  $S = \{s_1, s_2, \ldots, s_n\}$: conjunto de *strings* de tamanho m

- Seja:

  $d_H(s,t)$: função que calcula a "distância de Hamming" entre s e t

- Obter:

  $x \in \mathscr{A}$ de tamanho m, que minimiza d tal que $\forall s \in S \quad d_H(x,s) \leq d$.

Como exemplo, se $S = \{$"ACGT","TTAC","CCGC"$\}$, e $\mathscr{A} = \{A, B, C, D\}$ então $x =$ "TCGC", com $d = 2$, é uma solução ótima.

3.4 O Problema da Filogenia

Há dois aspectos a considerar na abordagem do Problema da Filogenia; primeiro se a mesma é baseada em matrizes de distâncias, tópico a ser comentado na Seção 4.2 e segundo se é baseada em características que será detalhado na Seção 4.3.

1. Problema da Filogenia baseada em matriz de distâncias (PFD)

   - Dados:
     M: matriz simétrica de ordem $m \times m$ com elementos m_{ij} reais, cada um deles correspondente à distância evolutiva estimada entre as espécies "i" e "j", $\forall i, j \in \{1, \ldots, m\}$
   - Obter:
     A: árvore filogenética associada à matriz M

   Neste caso, $\forall m \geq 2$ o problema PFD é resolvido em tempo polinominal, diz-se que PFD $\in P$ pois métodos tradicionais como o *Neighbor-Joining* [SN87] e WPGMA / UPGMA [DG05] transformam a matriz M na árvore A em tempo polinomial.

   A maior dificuldade, no entanto, está relacionada com a geração da matriz de distâncias [JC69, Tam92, TN84], pois a mesma é obtida por métodos de Alinhamento de Várias Seqüências (AVS) [BLP97, MS95], visto na seção 3.2. Wang & Jiang [WJ94] e Bonizzoni & Vedova [BV01], separadamente, mostram que o problema AVS é NP-Difícil [GJ79].

2. Problema da Filogenia baseada em características (PFC)

   - Dados:
     T: conjunto com m táxons a serem analisados
     C: conjunto com n características, cada uma delas com até r estados
     M: matriz de características ($T \times C$)
   - Obter:
     A: árvore filogenética associada à matriz M

Problema de Decisão para a Filogenia Perfeita:

Existe uma Filogenia Perfeita (*Perfect Phylogeny*) para a instância: (T,C,m,n,r) ?

Teorema 1. *"Para $r = 2$ (matriz binária) o problema da filogenia perfeita (PFP) é resolvido em tempo polinomial"*

Demonstração. Algoritmo 3 $\rightarrow \mathcal{O}(nm)$

Na Seção 4.4, página 59, será caracterizada a Filogenia Perfeita.

□

Teorema 2. *"Para $r \geq 3$ (mais de dois estados) PFP é NP-Completo "*

Demonstração. Bodlaender et al. [BFW93] mostram que:

- $PFP \in NP$
- $TCG \propto PFP$

O problema conhecido como *TCG (Triangulating Colored Graph)* [KW92] baseia-se na definição: "$G(V,E)$ é um grafo triangulado se ele não possui ciclos simples de tamanho maior ou igual a quatro" [Boa03].

Garey & Johnson [GJ79] citam um problema associado *k-Chromatic Number*; sendo que para $k = 3$ (3CN), juntamente com a clássica redução mestre, baseada no Teorema de Cook [Coo71], temos a prova de que PFP $\in NPC$.

$$Sat \propto 3Sat \propto 3CN \propto TCG \propto PFP$$

Steel [Ste92] também apresenta uma prova para o Teorema-2.

□

Problema de Otimização para o Problema da Filogenia:

Se para (T,C,m,n,r) não existe uma filogenia perfeita, então obter a árvore A da seguinte forma:

- Sejam:

  S: conjunto de soluções viáveis do problema da filogenia

  $f : S \to \mathscr{R}$: função que associa a cada $s \in S$ um valor real baseado em algum critério de cálculo.

- Obter:

  $s^* \in S$ tal que $f(s^*) \leq f(S), \forall s \in S$

Os critérios mais utilizados para o cálculo de f são o da parcimônia [DJS86, JN90, MS95, PHY06, SM87, SO90] com ou sem consenso [Amo97, QM06, SDB00], o da máxima verossimilhança[1] [BGP05, GGLD05, KNT+05, SLMW02, SHB+01] e o de índices de consistências [Far89]. Na definição acima, s^* é a melhor árvore A, que no caso é a solução do problema.

Teorema 3. *"$\forall r \geq 2$ PFC é NP-Difícil"*

Demonstração. Foulds & Graham [FG82] e Day, Johnson & Sankoff [DJS86] mostram separadamente que:

- $PFC \in NP$
- $STG \propto PFC$ (STG= Steiner Tree in Graphs)

Garey & Johnson citam as seguintes reduções:

$$Sat \propto 3Sat \propto 3DM \propto X3C \propto STG \propto PFC$$

Onde: 3DM = *Three-dimensional Matching* e X3C = *eXact Cover by 3-Sets.*

□

O problema da filogenia, considerado no enfoque deste trabalho, é um problema de otimização que consiste em minimizar o número de passos evolutivos para obter a filogenia mais próxima da real. Como vimos, esse critério escolhido é chamado de **parcimônia**.

[1] Em inglês, Maximum Likelihood

Ao finalizar este capítulo, podemos dizer que o estudo das técnicas de alinhamento, mesmo não sendo o foco do trabalho, foi abordado aqui, primeiro com o intuito de mostrar uma das aplicações mais utilizadas em Bioinformática, segundo por gerar matrizes de distâncias que servem como entrada para a construção de filogenias. No capítulo seguinte serão apresentadas formas de construir e representar as árvores filogenéticas.

CAPÍTULO 4

Árvores Filogenéticas

4.1 Formas de Representação das Árvores Filogenéticas

As árvores filogenéticas podem ser enraizadas ou não. Em uma árvore com raiz é possível introduzir a noção de traços ancestrais (plesiomórficos) e derivados (apomórficos), o que não ocorre nas árvores sem raiz.

Uma árvore filogenética pode ser representada por uma das seguintes formas:

- cladograma inclinado;
- cladograma retangular;
- árvore radial;
- árvore livre e
- filograma.

A Figura 4.1 contém um modelo de **cladograma inclinado** correspondente a uma filogenia de alguns tetrápodes (vertebrados terrestres possuidores de quatro membros). Observa-se na figura que a seqüência evolutiva é clara e o grupo de controle "peixes" é identificado, o que não ocorre numa árvore não enraizada, na forma **radial**, mostrada na Figura 4.2. Neste tipo de representação, a distância de qualquer extremidade (folha) a um ponto central, que em geral não é raiz da árvore, é fixa.

Quando for necessário que uma árvore filogenética expresse a "distância" evolutiva entre as espécies, onde o comprimento de cada um de seus ramos (ou arestas) são proporcionais a esta distância, deve-se usar a forma de um **filograma**. Um modelo desta forma de representação pode ser visto na Figura 4.3. Outras formas para este modelo são mostradas na Figura 4.4.

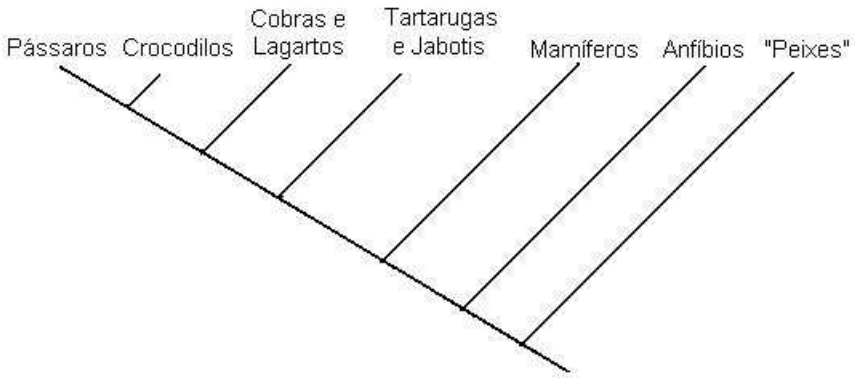


Figura 4.1 Modelo de uma árvore filogenética enraizada na forma de cladograma inclinado com indicação do grupo externo "peixes"

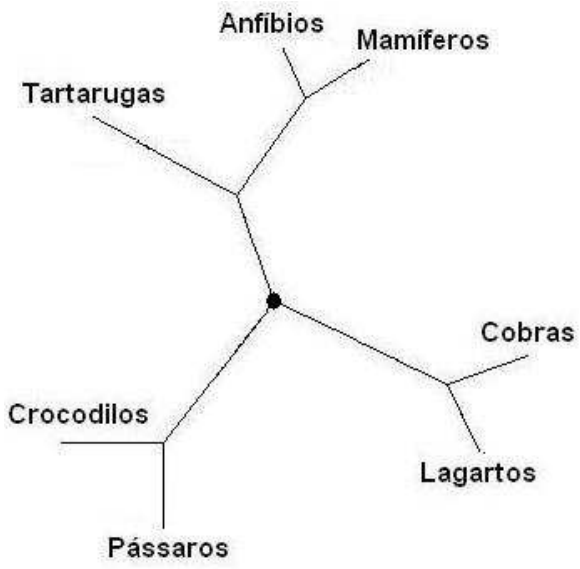


Figura 4.2 Árvore filogenética sem raiz na forma radial

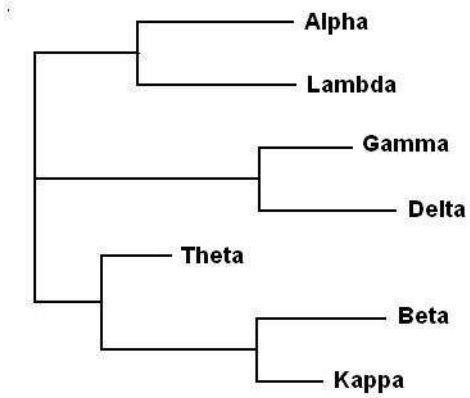


Figura 4.3 Modelo de uma árvore filogenética com raiz na forma de filograma

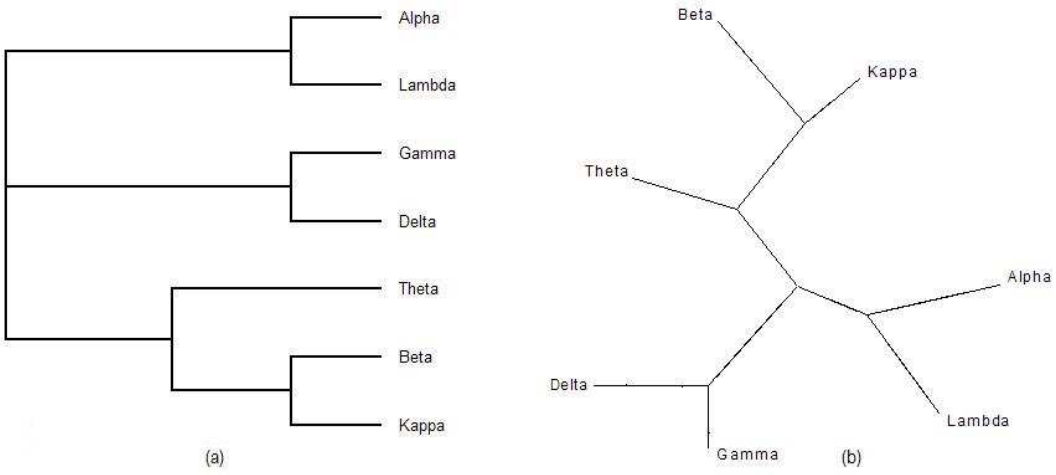


Figura 4.4 Árvores filogenéticas equivalentes ao filograma da Figura 4.3 nas seguintes formas de representação: a) cladograma retangular b) árvore livre.

A diferença básica entre os cladogramas (inclinado ou retangular) e as demais formas de representação, é que nestas últimas os táxons que aparecem nas folhas da árvore não estão alinhados.

4.2 Árvores Filogenéticas a partir de Matrizes de Distâncias

4.2.1 Geração das matrizes de distâncias

Um método de construção de uma árvore filogenética a partir de distâncias funciona basicamente em dois passos, primeiro são feitos os cálculos das "distâncias" genéticas cujos resultados são dispostos numa matriz e a seguir a árvore é reconstruída a partir desses dados.

Uma "matriz de distâncias" é uma matriz triangular superior com diagonal nula contendo as distâncias evolutivas, em termos relativos, entre as espécies em estudo. Ela pode ser gerada, conforme vimos na Seção 3.2, a partir de programas que analisam e fazem alinhamentos de seqüências de nucleotídeos ou aminoácidos como PROTDIST e DNADIST do pacote Phylip [Fel93] e DISTMAT do Emboss [RLB00]. Esses programas utilizam métodos universalmente conhecidos, como Jukes-Cantor [JC69, SOWH96], Kimura Two-Parameters [Kim80], Tajima-Nei [TN84], Tamara [Tam92] e Jin-Nei Gamma [JN90].

Em geral os modelos de alinhamento levam em consideração que as mutações de um nucleotídeo para outro são mais prováveis de ocorrerem através de transições, A $\leftrightarrow$ G (purinas) e T $\leftrightarrow$ C (pirimidinas) do que de transversões, T $\leftrightarrow$ A, T $\leftrightarrow$ G, C $\leftrightarrow$ A, C $\leftrightarrow$ G.

Entretanto, neste trabalho, não temos o propósito de detalhar estes métodos. Concentramos nossos estudos na fase de reconstrução da árvore filogenética associada a uma matriz de distâncias já conhecida; para isso, os métodos a serem utilizados são o UPGMA e o Neighbor-joining descritos nas subseções seguintes.

Uma outra forma de geração de uma matriz de distância, diferente daquela apresentada na Seção 3.2 para gerar a árvore filogenética correspondente pode ser vista através do exemplo a seguir. Considere um teste de DNA para identificar qual de dois suspeitos "A" e "B" transmitiram o vírus HIV/Aids para uma vítima "V" de estupro. Na Tabela 4.1 estão disponíveis

seqüências com 30 nucleotídeos do vírus infectado nas pessoas envolvidas no teste, onde a seqüência "X" pertence a uma pessoa com o vírus, porém, não relacionada com o crime; neste caso, como já mencionado, ela corresponde ao chamado grupo de controle, ou grupo externo.

Tabela 4.1 Seqüências de DNA's para teste de HIV – exemplo retirado da fonte [SH03, p.234]

Táxons	Parte da seqüência do DNA do vírus
X - grupo externo	AAGCTTCATAGGAGCAACCATTCTAATAAT
A - suspeito1	AAGCTTCACCGGCGCAGTTATCCTCATAAT
B - suspeito2	GTGCTTCACCGACGCAGTTGTCCTTATAAT
V - vítima	GTGCTTCACCGACGCAGTTGCCCTCATGAT

Sem perda de generalidade, considerando que transições e transversões ocorrem com igual probabilidade e comparando estas seqüências, duas a duas, temos na Tabela 4.2 os percentuais de caracteres distintos. Esses resultados com arredondamento para o inteiro mais próximo são apresentados na Tabela 4.3 que corresponde então à "matriz de distâncias" em valores relativos. Matrizes semelhantes, com grandezas proporcionais, pode ser obtida pela execução dos programas *Clustal* e *ProtDist*, do pacote PHYLIP [Fel93].

Tabela 4.2 Distâncias comparativas entre os táxons da Tabela 4.1. O percentual de caracteres distintos é chamado de dissimilaridade ou divergência

Táxons Comparados	Caracteres Distintos	Razão	Valor	%
X–A	8	8/30	0.266	26.6
X–B	12	12/30	0.400	40.0
X–V	13	13/30	0.433	43.3
A–B	5	5/30	0.166	16.6
A–V	6	6/30	0.200	20.0
B–V	3	3/30	0.100	10.0

A Figura 4.5 mostra duas formas da filogenia correspondentes aos valores da Tabela 4.2. De acordo com essa filogenia conclui-se que o suspeito2 (B) é o culpado pelo crime pois é dele a seqüência de DNA do vírus que possui mais características comuns com a "amostra"

Tabela 4.3 Matriz de distâncias correspondente à Tabela 4.2

Táxons	X	A	B	V
X	0	27	40	43
A		0	17	20
B			0	10
V				0

da vítima (V). O método utilizado para gerar esta árvore filogenética foi o *Neighbor-Joining* [DG05, SN87] cuja construção utiliza-se do conceito de distância num espaço métrico descrito a seguir.

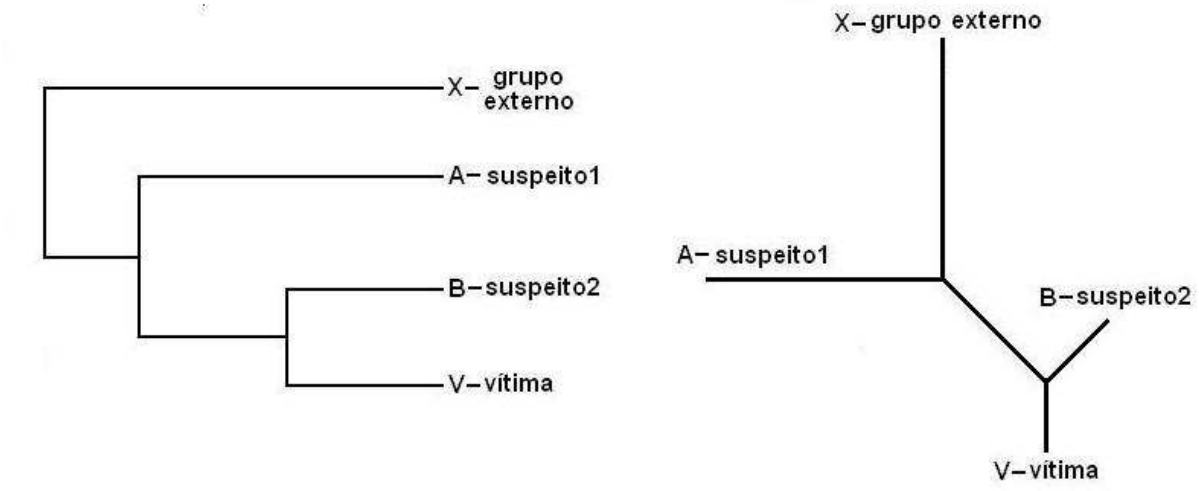


Figura 4.5 Filogenias correspondentes à matriz de distância da Tabela 4.2 nas formas de cladograma retangular e árvore livre – adaptado da fonte [SH03, p.235]

4.2.2 Espaço Métrico

Um espaço métrico [SM97] é dado pela função: $d : E \times E \rightarrow R$, tal que as seguintes propriedades são válidas para quaisquer elementos **distintos** do conjunto $\{x, y, z\} \subseteq E$:

1. $d(x,x) = 0 \quad e \quad d(x,y) > 0$ (d é uma função não-negativa)

2. $d(x,y) = d(y,x)$ (d é simétrica)

3. $d(x,y) \leq d(x,z) + d(y,z)$ (satisfaz a desigualdade triangular)

Diz-se que um espaço métrico é aditivo se, e somente se, dados quatro objetos: i, j, k e l, representados por exemplo na Figura 4.6, são válidas as relações:

$$d(i,j) + d(k,l) = d(i,l) + d(k,j) \geq d(i,k) + d(j,l) \tag{4.1}$$

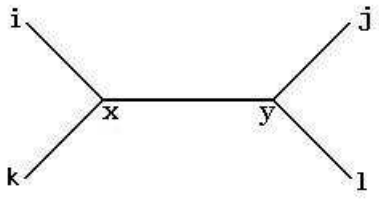


Figura 4.6 Exemplo de elementos pertencentes a um Espaço Métrico Aditivo onde são válidas as relações: $d_{ij} + d_{kl} = d_{il} + d_{kj} \geq d_{ik} + d_{jl}$ *e* $d_{xy} \geq 0$

4.2.3 Método *Neighbor-Joining*

O Método *Neighbor-Joining* consiste em pesquisar sucessivamente na matriz de distâncias os vizinhos mais próximos e utilizar as relações no espaço métrico aditivo para obter um novo segmento, atualizando a matriz de acordo com os procedimentos indicados no Algoritmo 2. Por exemplo, na matriz da Tabela 4.2, "B" e "V" são os vizinhos mais próximos, e $AV > AB$; logo, num espaço métrico aditivo teríamos as seguintes relações que correspondem à Figura 4.7.

$$\begin{cases} AR + RV = AV = 20 \\ AR + RB = AB = 17 \\ RB + RV = BV = 10 \end{cases}$$

Observa-se que o sistema representado por estas equações sempre é determinado, com solução única dada pelas equações:

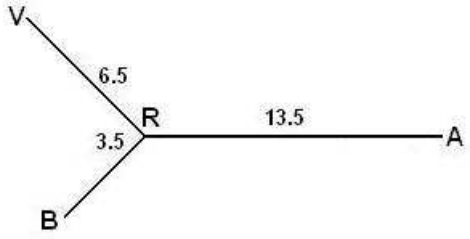


Figura 4.7 Determinação do ponto de ramificação (R) do segmento BV num espaço métrico aditivo

$$AR = \frac{AV + AB - BV}{2} = \frac{20 + 17 - 10}{2} = 13.5 \tag{4.2}$$

$$RV = \frac{AV + BV - AB}{2} = \frac{20 + 10 - 17}{2} = 6.5 \tag{4.3}$$

$$RB = \frac{AB + BV - AV}{2} = \frac{17 + 10 - 20}{2} = 3.5 \tag{4.4}$$

Nessa situação, a ramificação AR a partir de BV sempre existe se a solução do sistema for positiva; caso isto não ocorra, o espaço não seria métrico e o ponto R não pertenceria ao segmento BV. Para solucionar este problema, de modo a mostrar o relacionamento entre as espécies, são utilizados os métodos UPGMA e WPGMA descritos a seguir.

4.2.4 Métodos UPGMA e WPGMA

O Método UPGMA (*Unweighted Pair-Group Method using Arithmetic average*) faz uso da média aritmética entre os segmentos e é usado quando as taxas de evolução são aproximadamente constantes entre diferentes espécies; enquanto que o método WPGMA (*Weighted Pair-Group Method using Arithmetic average*) correspondente a uma média ponderada entre os elementos não pertencentes a um espaço métrico, de modo a refletir melhor as vizinhanças [DG05, SH03] entre as espécies em análise.

Por exemplo, para $AB = 19, AV = 36$ e $BV = 15$ a solução do sistema seria: $AR = 20$, $RV = 16$ e $RB = -1$; utilizando os métodos descritos poderíamos apresentar as soluções indicadas

pelas seguintes equações:

- UPGMA

$$BR = \frac{BV}{2} = \frac{15}{2} = 7.5 \tag{4.5}$$

$$RV = BV - BR = 15 - 7.5 = 7.5 \tag{4.6}$$

$$AR = \frac{(AB - BR) + (AV - RV)}{2} = 20 \tag{4.7}$$

- WPGMA

$$BR = \frac{AB}{AB + AV}.BV = \frac{19}{19 + 36}.15 = 5.18 \tag{4.8}$$

$$RV = BV - BR = 15 - 5.18 = 9.82 \tag{4.9}$$

$$AR = \frac{(AB - BR) + (AV - RV)}{2} = 20 \tag{4.10}$$

As representações gráficas para os exemplos anteriores podem ser vistas na Figura 4.8.

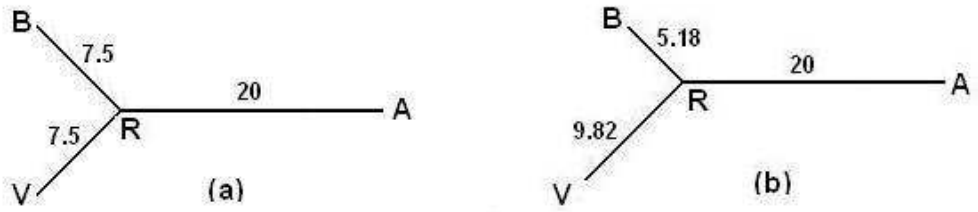


Figura 4.8 a) Solução usando o método UPGMA b) Solução usando o método WPGMA

Observa-se que o Método UPGMA divide o segmento ao meio, enquanto que o WPGMA o faz de forma proporcional, refletindo melhor as similaridades; por este motivo, escolhemos este segundo método para nossa implementação.

O Algoritmo 1 descrito a seguir calcula a interseção de um segmento de acordo com estes métodos descritos.

4.2.5 Interseção de segmentos numa matriz de distâncias

O algoritmo a seguir obtém inicialmente a solução do sistema com base nos segmentos de entrada, de acordo com o método *Neighbor-Joining* quando os segmentos pertencerem a um mesmo espaço métrico através das equações 4.2 a 4.4 (página 48), ou utiliza o método WPGMA, de acordo com as equações 4.8 a 4.10 (página 49)

Algoritmo 1: Determina o ponto de interseção de segmentos numa matriz de distâncias

Input: (AB, AC, BC)
$AB > 0; \quad AC > 0; \quad BC > 0$

Output: (AR, BR, CR)
$AR + RB = AB; \quad AR + RC = AC; \quad BR + RC = BC$ (se espaço métrico) ou
divide o segmento de forma proporcional, caso contrário

```
1.1  begin
1.2      AR = (AC + AB − BC)/2
1.3      BR = (AB + BC − AC)/2
1.4      CR = (AC + BC − AB)/2
1.5      if (AR < 0) or (BR < 0) or (CR <0) then
1.6          BR = (AB ∗ BC)/(AB + AC)
1.7          CR = BC − BR
1.8          AR = (AB + AC − BR − CR)/2
1.9      endif
1.10 end
```

4.2.6 Formato Padrão Newick

O formato Newick[1], idealizado em 1857 pelo matemático inglês Arthur Cayley (1821–1895), foi adotado como padrão por um Comitê[2] informal da SSE (Society for the Study of Evolution) em Durham, New Hampshire em 26/06/1986 [NEW07].

Esse formato representa uma árvore enraizada através de uma cadeia de caracteres, delimitada ao final por um "ponto e vírgula", contendo parênteses "aninhados". É uma forma prática bastante usada para entrada e saída de dados em programas que manipulam árvores. A Figura 4.9 dá uma idéia clara do uso desta representação.

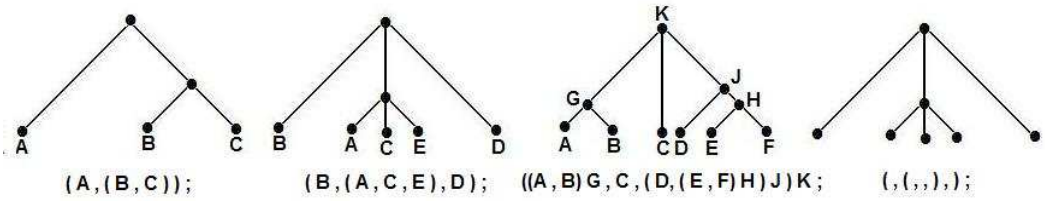


Figura 4.9 Exemplos de representação de árvores usando o formato padrão *Newick*

Cada par de parênteses corresponde a um nó interno, ou à raiz da árvore. Todos os nós, inclusive as folhas, são separados por vírgulas e seus rótulos sem parênteses os identificam. Um rótulo ou um identificador é uma subcadeia (pode ser vazia) que contém caracteres diferentes de "branco", "vírgula", "ponto e vígula", "dois pontos", "parênteses", "colchetes" e "chaves".

"Nós" internos podem ser identificados; neste caso seu rótulo deve aparecer logo após o par de parênteses correspondente. O tamanho dos arcos (arestas ou ramos) são números inteiros ou reais com ponto decimal que podem aparecer após cada "nó" separados por "dois pontos". O valor indicado corresponde a uma medida relativa de unidade de comprimento da distância deste nó (filho) até seu ascendente imediato (pai). A Figura 4.10 mostra um exemplo destes casos.

[1]Esta denominação deve-se ao nome de um restaurante em Dover - New Hampshire, USA, onde o comitê esteve reunido na época em que adotou este padrão.

[2]Formado pelos cientistas James Archie, William H.E. Day, Wayne Maddison, Christopher Meacham, F. James Rohlf, David Swofford e Joseph Felsenstein.

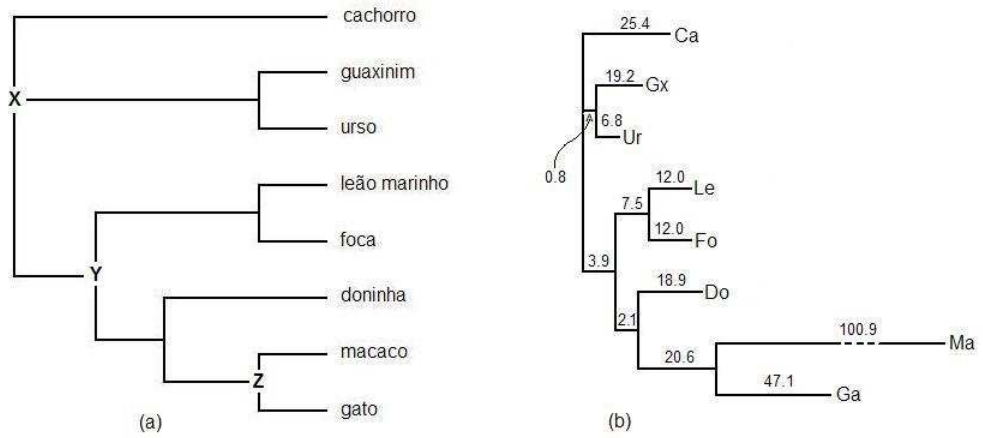


Figura 4.10 Árvores filogenéticas e suas formas de representações no formato *Newick* – adaptado da fonte [NEW07]

Ao utilizar este formato para identificar uma árvore filogenética, como na Figura 4.10, deve-se atentar que, em geral, uma mesma árvore tem mais de uma forma no padrão *Newick*. Por exemplo "(A,(B,C),D);" e "(D,(C,B),A);" representam a mesma árvore. Vale também lembrar que, para muitas aplicações biológicas não há necessidade de identificar a raiz da árvore; nesse caso é sugerido que ela seja escolhida arbitrariamente para iniciar sua codificação.

Programas que desenham árvores, como o *TreeView* [Pag01], utilizam este formato como entrada. O "pacote" *Phylip* [Fel93] também o faz de forma generalizada; daí, em alguns trechos deste trabalho o formato *Newick* é chamado também de "padrão *Phylip*".

4.2.7 Algoritmo para construir uma Árvore Filogenética a partir de uma Matriz de Distâncias

O Algoritmo 2 apresentado a seguir [VGM+06] faz uma busca na matriz de distâncias dos vizinhos mais próximos, a seguir é calculado o ponto de ramificação pelo Algoritmo 1 (página 50), inicialmente de acordo com o critério apresentado na Figura 4.7(*Neighbor-Joining*), senão, com aquele descrito na Figura 4.8 (WPGMA), observando que o segmento "BV" é dividido

de forma proporcional e exata, enquanto que o segmento "RA" tem tamanho reduzido para tornar os pontos considerados pertencentes a um "novo" espaço métrico; estes procedimentos são repetidos sucessivamente até que toda a matriz tenha sido considerada. Opcionalmente o algoritmo desenha uma filogenia a partir de uma entrada de dados no formato *Newick* (ou como chamamos aqui "padrão Phylip"), descrito na subseção anterior.

Algoritmo 2: Desenha uma Árvore Filogenética [VGM+06] a partir de uma matriz de distância ou de uma *string* no formato padrão Newick [NEW07]. O Procedimento *intersect* (linha 10) corresponde ao Algoritmo 1

Input: $(n, A/S)$
$S =$ *string* no formato padrão (entrada opcional)
$E =$ conjunto de vetores binários que identificam o conteúdo das arestas de $\mathscr{T}$
$A =$ matriz simétrica de ordem $n \times n$

Output: (Desenho da Árvore Filogenética)
$\mathscr{T}$ nos formatos: .jpg, .png ou .svg

```
2.1  begin
2.2      if S ≠ " " then
2.3          View(S)                    {Visualiza a árvore filogenética}
2.4      else
2.5          S ← " "
2.6          while n > 1 do
2.7              for k ← 1 to n step 2 do
2.8                  (ii, jj) ← {(i, j)  tal que |a_ij|_min, ∀{i, j} ⊂ {k,...n}}
2.9                  swap(A(k,1:n),A(ii,1:n)); swap(A(1:n,k),A(1:n,jj))
2.10                 (x, y, z) ← intersect(a_{k,k+1}, a_{k,k+2}, a_{k+1,k+2})
2.11                 A(k+1,1:n) ← huge;    A(1:n,k+1) ← huge
2.12                 A(k,1:n) ← A(k,1:n) − x;    A(1:n,k) ← A(1:n,k) − x)
2.13             endfor
2.14             S ← concat(S, segment(x, y, z))
2.15             n ← n − 2
2.16         endw
2.17         if n = 1 then S ← concat(S, segment(a12, a13, 0))
2.18     endif
2.19     View(S)                        {Visualiza a árvore filogenética}
2.20 end
```

Aplicando este método para a matriz da Tabela 4.2, obtivemos a árvore filogenética indicada de duas formas na Figura 4.11 cuja representação no formato texto, no padrão Phylip, é dada por:

$$(X : 25, A : 2, (B : 3.5, V : 6.5) : 11.5)$$

.

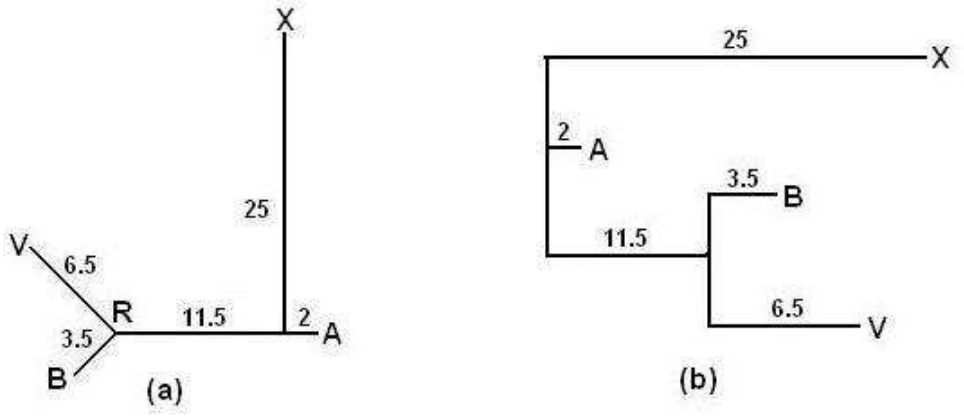


Figura 4.11 Árvore Filogenética com quatro táxons gerada pelo Algoritmo 2 para a matriz da Tabela 4.3, página 46. a) Representação por árvore livre b) Representação na forma de Filograma

A entrada de dados do Algoritmo 2 é uma matriz (simétrica) de distâncias num formato de texto simples com a estrutura indicada na Tabela 4.4; observe que após a primeira linha que contém o "título" as demais começam com uma identificação da espécie em estudo, seguido dos valores das distâncias no formato de matriz triangular superior, partindo do valor zero que é o elemento da diagonal principal em cada linha; uma barra invertida (\) no final da linha deve ser usada quando for necessário para indicar continuação desta na linha seguinte conforme mesmo exemplo na Tabela 4.5 em outro formato de entrada. A Figura 4.12 mostra a árvore correspondente à matriz da Tabela 4.4.

Este algoritmo é utilizado para desenhar todas as árvores geradas por ocasião dos experimentos computacionais.

Tabela 4.4 Modelo-1 de entrada para o Algoritmo 2 em sua forma simples – exemplo retirado da referência [SM97, p.195]

	$TITULO-1: Teste\ com\ 7\ especies$						
A	0	63	94	111	67	23	107
B		0	79	96	16	58	92
C			0	47	83	89	43
D				0	100	106	20
E					0	62	96
F						0	102
G							0

Tabela 4.5 Modelo-2 de entrada para o Algoritmo 2 com uso de continuação de linha (idêntico ao modelo da Tabela 4.4)

	$TITULO-2: Teste\ com\ 7\ especies$					
A	0	63	94	111	67	/
	23	107				
B	0	79	96	16	58	/
	92					
C	0	47	83	89	43	
D	0	106	106	20		
E	0	62	96			
F	0	102				
G	0					

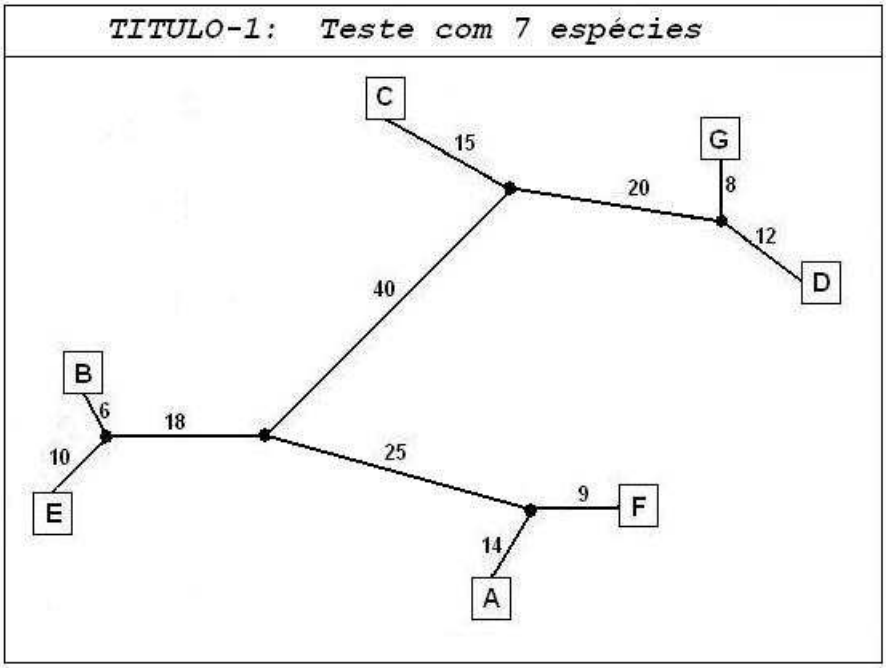


Figura 4.12 Árvore filogenética com sete táxons gerada pelo Algoritmo 2 a partir da Tabela 4.4 – teste retirado da referência [SM97, p.196]

4.3 Filogenia baseada em Características

Inicialmente iremos conceituar as "matrizes de características", ou seja, qual sua origem e o que elas representam. Para fazer uma análise filogenética é necessário antes construir as listas das características das espécies a serem analisadas, ou seja, estabelecer as homologias, sua codificação e ordenação, o que resulta numa matriz de dados onde suas linhas contêm as espécies e suas colunas as características, ou vice-versa. A construção destas matrizes é um passo muito importante na obtenção de filogenias, pois a qualidade da árvore filogenética final depende diretamente delas.

Após o estabelecimento das homologias é necessário fazer a atribuição de um símbolo numérico a cada um dos diferentes estados do caráter, em um processo que é conhecido por codificação. As codificações mais simples têm apenas dois estados, aos quais são atribuídos códigos binários, usualmente "0" para o estado ancestral e "1" para o estado derivado, ou ainda, correspondendo à ausência ("0") ou presença ("1") da característica da coluna "j" na espécie da linha "i". Pode-se atribuir ainda a esse elemento um terceiro valor, em geral igual a "?", que indica o desconhecimento da existência de tal característica para aquela espécie.

A matriz de características da filogenia da Figura 2.8 da página 22 teria a forma indicada na Tabela 4.6 e cladograma retangular correspondente na Figura 4.13.

Algumas observações devem ser destacadas nestas representações; primeiro, nota-se uma similaridade entre o cladograma da Figura 4.13 e a Filogenia da Figura 2.8; segundo, as características estudadas não permitem separar as espécies "camundongo" e "chimpanzé"; observe que ao considerar cada linha da matriz da Tabela 4.6 como um identificador da espécie, poderíamos pressupor que se trata da mesma espécie. Este equívoco mostra a necessidade de incluir novas características.

Existem também os caracteres multiestados, que por sua vez podem ser classificados em ordenados e não ordenados. Seqüências de caracteres não ordenados possibilitam apenas a descoberta das possíveis transformações, já que não se assume nada quanto à mudança de estados (veja Seção 2.6), ou seja, qualquer estado pode mudar para outro; porém, os testes (experimentos computacionais) do Capítulo 5 não consideram este tipo de instância. Para exemplificar

Tabela 4.6 Matriz com sete características e oito espécies

	CARACTERÍSTICAS						
ESPÉCIES	Ma	Pu	GU	C4	Pe	Pê	GM
Lampréia	0	0	0	0	0	0	0
Peixe	1	0	0	0	0	0	0
Salamandra	1	1	0	0	0	0	0
Lagarto	1	1	1	0	0	0	0
Crocodilo	1	1	1	1	0	0	0
Pombo	1	1	1	1	1	0	0
Camundongo	1	1	1	1	0	1	1
Chimpanzé	1	1	1	1	0	1	1

Legenda para as características

Ma	Pu	GU	C4	Pe	Pê	GM
Mandíbulas	Pulmões	Garras e Unhas	Coração com 4 câmaras	Penas	Pêlos	Glândulas Mamárias

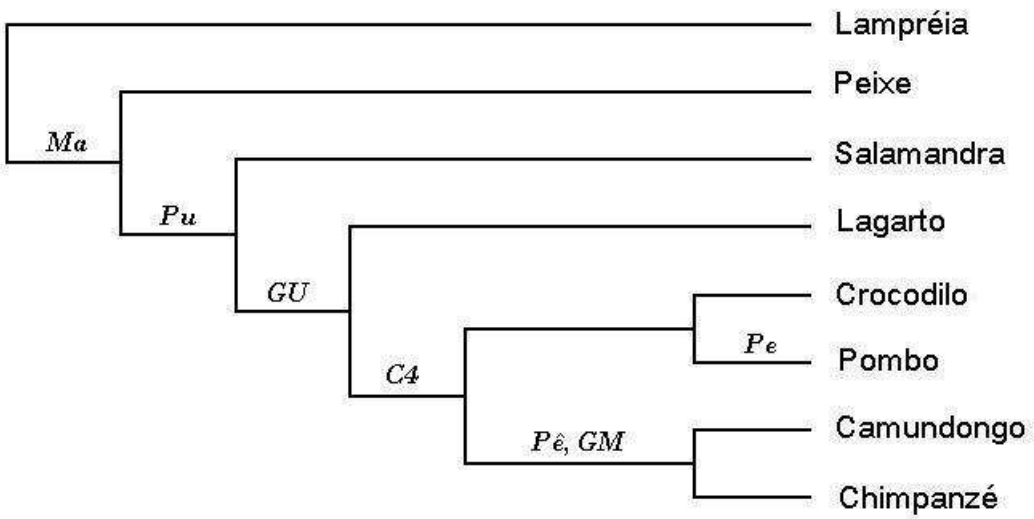


Figura 4.13 Cladograma retangular com sete características e oito espécies correspondente à matriz da Tabela 4.6 – adaptado da fonte [PSOH03, p.429]

estes casos, observe as informações contidas na Tabela 4.7.

Tabela 4.7 Exemplo de matriz com caracteres múltiplos (não binários)

	TÁXONS			
CARACTERÍSTICAS	Mamíferos	Lagartos	Crocodilos	Anfíbios
Aberturas Temporais	1 par	2 pares	2 pares	ausente
Ventana Preorbitária	ausente	ausente	presente	ausente
Pélvis	normal	normal	modificada	normal
Arco Aórtico Principal	esquerdo	esq. / dir.	direito	esq. / dir.
	Codificação Multiestado			
Aberturas Temporais	1	2	2	0
Ventana Preorbitária	0	0	1	0
Pélvis	0	0	1	0
Arco Aórtico Principal	0	2	1	2

A partir da seção seguinte, até o final deste trabalho, tratamos do problema de reconstrução de árvores filogenéticas baseada em características com dois estados (matrizes binárias).

4.4 Filogenia Perfeita

O Problema da Filogenia Perfeita, baseado em matrizes de características, conforme visto na seção 3.4, página 37, é definido por:

Dados um conjunto $\mathscr{T}$ com **m** táxons e um conjunto $\mathscr{C}$ com **n** características, onde cada característica possui no máximo **r** estados, **determinar** se existe uma filogenia perfeita $\mathscr{A}$ para a matriz $\mathscr{M}_{m\times n}$.

Antes da definição de "Filogenia Perfeita" devemos observar que durante o processo de reconstrução de uma filogenia é necessário considerar os seguintes aspectos:

- existem exatamente **m** táxons terminais ou folhas;
- as características podem ser herdadas independentemente umas das outras;

- estados de uma característica evoluem sempre do estado do ancestral comum mais próximo ou mais recente;
- os nós internos da árvore representam espécies ancestrais hipotéticas que em geral não necessitam de identificação;
- a distância entre um nó interno (ancestral hipotético) e uma folha (táxon) pode ser interpretada como uma estimativa do tempo do processo de evolução;
- considerar a existência de processos de evolução paralela e casos de convergência e reversão de estados.

Se for possível evitar os eventos de convergência e reversão de estados, fazendo com que o conjunto de todos os nós da árvore final $\mathscr{A}$ que possuam o mesmo estado para uma determinada característica formem uma subárvore de $\mathscr{A}$, teríamos uma filogenia perfeita.

Assim, diz-se que uma filogenia é perfeita quando é possível construí-la em exatamente $n = |\mathscr{C}|$ passos com suas características compatíveis, ou seja, de forma a evitar conflitos nas informações filogenéticas. A compatibilidade [DS86, MS95] entre caracteres de uma matriz binária pode ser verificada na Figura 4.14.

Quando o número de estados é maior ou igual a três (multi-estado) há necessidade de fazer uma análise mais detalhada para verificação da compatibilidade entre as características; a Figura 4.15 dá uma idéia como isto poderia ser feito e a Tabela 4.8 mostra como uma matriz multi-estado poderia ser convertida numa matriz binária.

Podemos então dizer que "uma árvore filogenética A na qual todos suas características são compatíveis corresponde a uma Filogenia Perfeita" ou "existe uma Filogenia Perfeita para uma matriz de características se cada estado de cada caráter ocupa uma subárvore específica".

No exemplo da Tabela 4.9 existe filogenia perfeita para os conjuntos Υ e Ψ e não existe para Γ; neste caso, a filogenia deve ser inferida através de mudanças dos estados dos caracteres de acordo com as transformações evolutivas descritas na Seção 2.7 (página 23).

Vamos considerar $\mathscr{T} = \{A, B, C\}$, $\mathscr{C} = \{1, \ldots, 12\}$ e $r = 2$ estados, caracterizando como exemplo a matriz de estados binários da Tabela 4.9; o conjunto $\mathscr{C}$ foi separado em três partições $\Upsilon = \{1,2,3\}$, $\Psi = \{4,5,6,7\}$ e $\Gamma = \{8,9,10,11,12\}$.

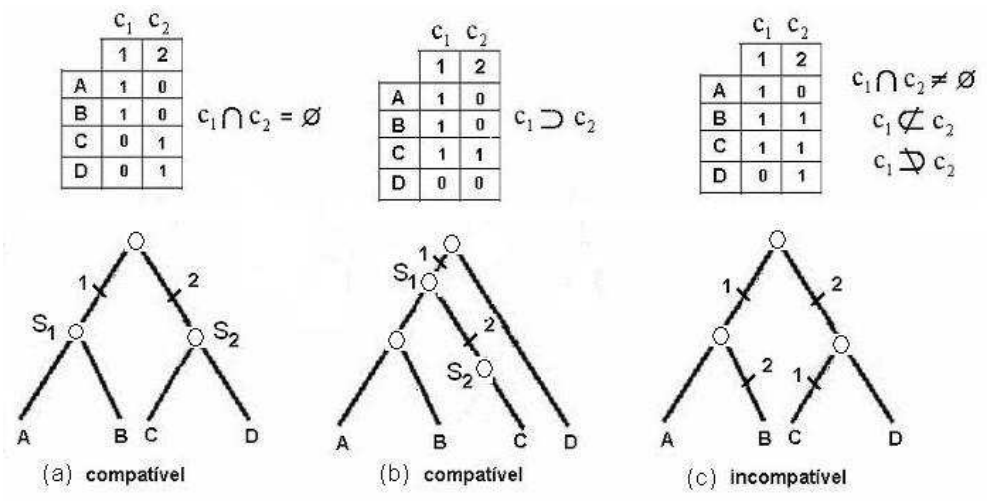


Figura 4.14 Filogenia baseada em característica para dois estados (matriz binária); observe que no caso de compatibilidade as características definem as subárvores S_1 e S_2 (a e b), o que não é possível quando são incompatíveis ou incongruentes (c)

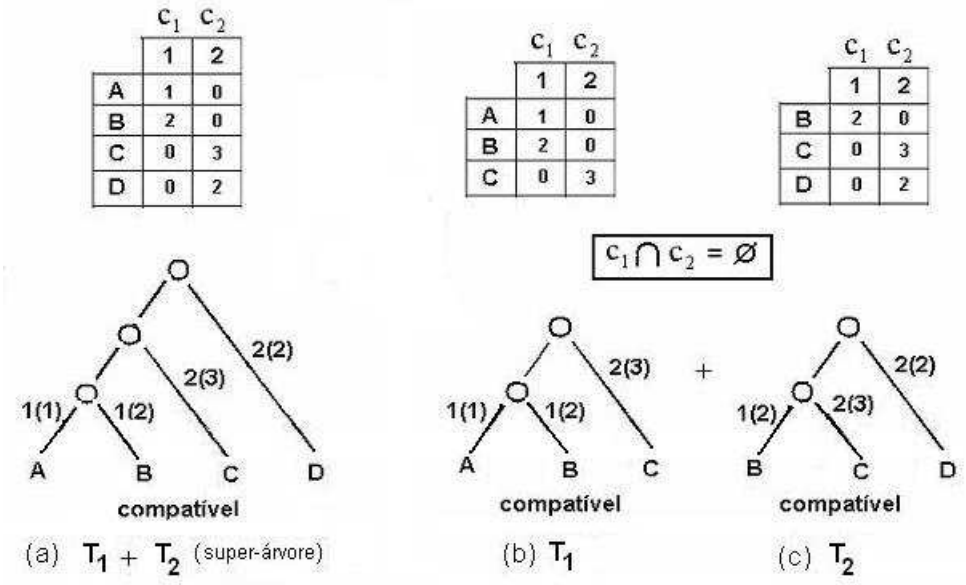


Figura 4.15 Filogenia baseada em característica para três estados (uma generalização para caracteres multi-estado); observe que após a separação da super-árvore (a) em duas (ou mais) subárvores (b e c) o problema se reduz a dois (ou mais) sub-problemas com estados binários

Tabela 4.8 Conversão de uma matriz multi-estado numa matriz binária; exemplo referente à matriz da Figura 4.15, onde pode ser observada a compatibilidade entre suas características.

	Características			
Táxons	1(1)	1(2)	2(2)	2(3)
A	1	0	0	0
B	0	1	0	0
C	0	0	0	1
D	0	0	1	0

Tabela 4.9 Matriz de estados binários

	Características											
Táxons	1	2	3	4	5	6	7	8	9	10	11	12
A	1	0	0	1	0	0	0	1	0	0	0	1
B	0	1	0	0	1	0	1	0	1	0	1	0
C	0	0	1	0	0	1	1	0	0	1	1	1
Conjuntos	Υ			Ψ				Γ				

Nas Figuras 4.16 e 4.17 são mostradas, respectivamente, as filogenias possíveis para os conjuntos Υ e Ψ.

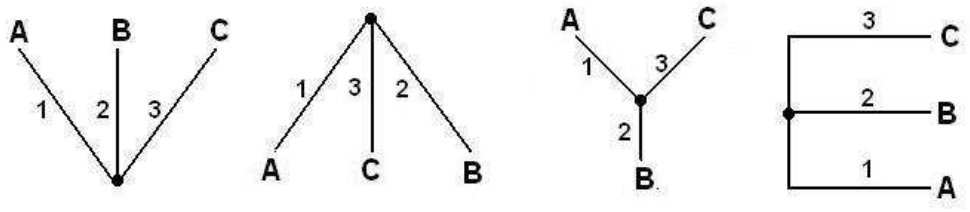


Figura 4.16 Formas de representação de uma mesma Filogenia com três táxons e três características distintas sem grupo monofilético para o conjunto Υ da Tabela 4.9 – esta forma caracteriza uma politomia, onde os três táxons partem da raiz. Observa-se que a ordem de disposição dos vértices juntamente com suas arestas não altera a árvore filogenética

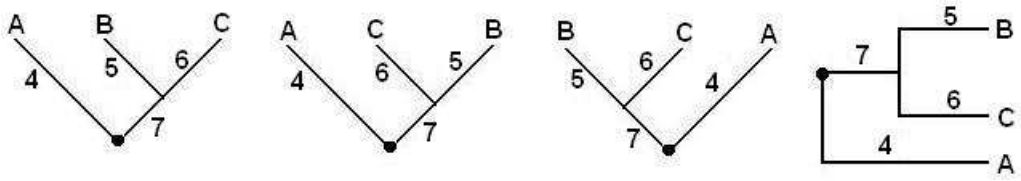


Figura 4.17 Formas de representação de uma mesma Filogenia com três táxons e quatro características distintas com um grupo monofilético (B,C) para o conjunto Ψ da Tabela 4.9. Neste caso a ordem de disposição das folhas pode alterar a árvore filogenética

Observa-se, na Figura 4.18, que com quatro passos é possível construir a árvore filogenética da Figura 4.17; cada passo corresponde a uma mudança de estado (ver seção 2.6).

Para três táxons existem somente três formas distintas de representar uma filogenia conforme indicado na Figura 4.19, visto que, as formas dispostas separadamente nas Figuras 4.16 e 4.17 são redundantes.

A forma para a filogenia da Figura 4.19-b mesmo correspondendo a uma politomia, portanto, sem um grupo monofilético definido, apresenta as mesmas relações filogenéticas da Figura 4.16, apesar de sugerir a existência de um ancestral "recente" interno; neste caso entende-se que ele é idêntico ao ancestral "distante" (raiz) pois a aresta que os une não possui nenhuma

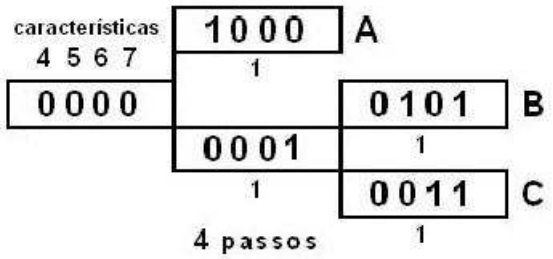


Figura 4.18 Número de passos na construção de uma árvore filogenética perfeita com três táxons e quatro características (representada na Figura 4.17); cada passo corresponde a uma transformação do tipo $0 \rightarrow 1$

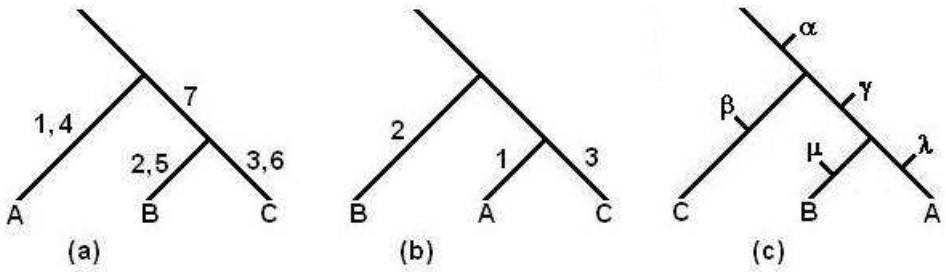


Figura 4.19 Formas de representação distintas de Filogenias para três táxons independente do número de características. (a) Filogenia com três táxons e sete características correspondente ao conjunto $(\Upsilon \cup \Psi)$ da Tabela 4.9 e grupo monofilético (B,C). (b) Forma redundante da filogenia da Figura 4.16 sem distinção do grupo monofilético (A,C). (c) Forma genérica com indicação das opções de locais de inclusão de um novo (quarto) táxon e grupo monofilético (A,B).

característica que os separe.

As cinco posições (α, β, γ, λ e μ) da Figura 4.19-c para cada uma das configurações com três táxons permitem concluir que podemos ter até 15 árvores filogenéticas distintas com quatro táxons; três delas estão representadas na Figura 4.20, sendo que a do item-c mostra sete posições para incluir um quinto táxon; seguindo este raciocínio teríamos nove posições para incluir um sexto táxon e assim por diante.

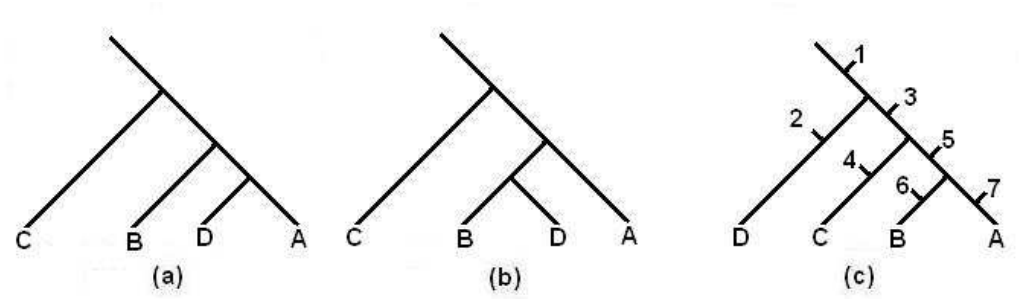


Figura 4.20 Algumas formas de representação de Filogenias para quatro táxons independente do número de características. (a) Inclusão na posição γ da Figura 4.19-c. (b) Idem para posição μ. (c) Idem para a posição α e indicação das opções de inclusão de um novo táxon

.

A Tabela 4.10 contém as quantidades (χ_m) de formas possíveis em função do número de táxons m. Os valores da tabela foram obtidos através da fórmula 4.11 a partir de $m = 2$ (que tem uma única configuração possível, conforme mostra a Figura 4.21).

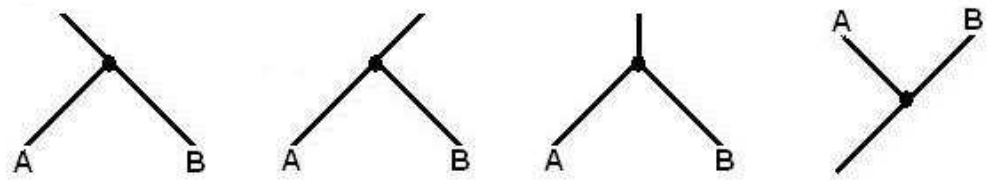


Figura 4.21 Única configuração possível para Filogenia com dois táxons; todas as formas representam a mesma árvore filogenética, sem identificação das características nas arestas

$$\chi_m = \prod_{k=2}^{m}(2k-3) \Rightarrow \chi_m = 1 \times 3 \times 5 \times 7 \times 9 \ldots (2m-3) = \frac{(2m-3)!}{2^{m-2}.(m-2)!} \tag{4.11}$$

Tabela 4.10 Relação entre o número de árvores filogenéticas/cladogramas (χ_m) de acordo com a Equação 4.11 $\times$ número de táxons (m) e comparação com a função fatorial ($m!$)

m	χ_m	$m!$	$\chi_m/m!$
2	1	2	0,50
3	3	6	0,50
4	15	24	0,63
5	105	120	0,88
6	945	720	1,31
7	10.395	5.040	2,06
8	135.135	40.320	3,35
9	2.027.025	362.880	5,59
10	34.459.425	3.628.800	9,50
20	8,20e0021	2,43e0018	3370,79
100	3,35e0184	9,33e0157	3,55e026
500	1,01e1280	1,27e1134	8,27e145
1000	3,85e2863	4,02e2567	9,56e295

Conforme apresenta a Tabela 4.10, o número de cladogramas cresce exponencialmente em função do número de táxons, numa "explosão" combinatória superior a função fatorial, conforme pode ser verificado na última coluna desta tabela.

Cabe observar que num processo de busca é levado em consideração o tipo da instância do problema, de modo que, em função dos dados, a maior parte destas configurações não serão analisadas; outro fator importante a lembrar é que há apenas uma **única** filogenia real, e se estes dados conduzem à filogenia perfeita ela será a encontrada, senão, um método de inferência deve ser utilizado da forma mais inteligente possível para selecionar, num tempo hábil, a melhor solução do problema, ou seja, aquela filogenia mais "parecida" com a "real".

Na Biologia Computacional/Otimização usamos os termos "solução ótima" para indicar uma solução "real" e o termo verossimilhança [BGP05, GGLD05, KNT+05, SLMW02, SHB+01] ou máxima parcimônia [DJS86, MS95, PHY06, SM87, SO90] para indicar a solução mais "parecida" com a real, sendo portanto a mais provável.

Nesta seção será mostrado como identificar se uma matriz de estados binários admite uma filogenia perfeita com base no seguinte Lema [SM97, Gus97]:

Lema 1. *"Uma matriz binária $\mathscr{M}$ admite uma Filogenia Perfeita se, e somente se, para cada par de características distintas (colunas $i \neq j$), $\forall\{i,j\} \subseteq \{1,\ldots,n\}$, os conjuntos $\beta_i = \mathscr{M}[1:m,i\,]$ e $\beta_j = \mathscr{M}[1:m,j\,]$ são disjuntos, $(\beta_i \cap \beta_j) = \varnothing$, ou um deles contém o outro, $(\beta_i \subseteq \beta_j)$ ou $(\beta_j \subseteq \beta_i)$*

Demonstração. $\forall\{i,j,\kappa\} \subseteq \{1,\ldots,n\}$, $i \neq j$

$\Rightarrow$ Vamos assumir que $\mathscr{M}$ admite uma Filogenia Perfeita.

Seja A a árvore correspondente a essa filogenia.

Como $\mathscr{M}$ é binária significa que todos os táxons τ que contém um determinado caráter κ ($m_{\tau\kappa} = 1$) pertencem a uma mesma subárvore de A (ver Figura 4.22)

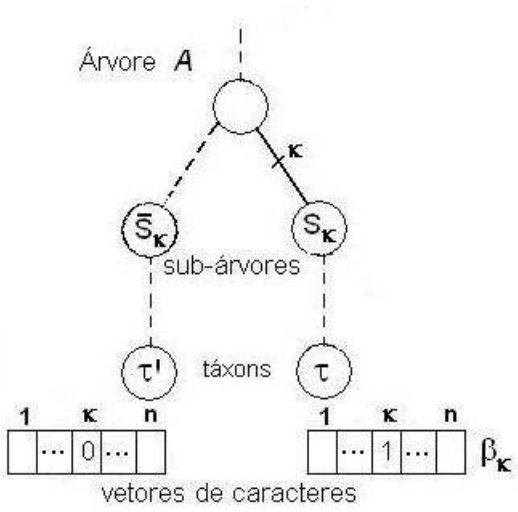


Figura 4.22 Árvore correspondente a uma Filogenia Perfeita com indicação de uma subárvore com um caráter compatível

Hipótese 1. *suponhamos a existência de três táxons τ_1, τ_2 e τ_3, apresentados na Tabela 4.11, cujas características "i" e "j"* **não** *satisfazem as condições impostas pelo Lema.*

Tabela 4.11 Matriz de características para a hipótese-1; observe que $(\beta_i \cap \beta_j) \neq \varnothing$, $(\beta_i \nsubseteq \beta_j)$ e $(\beta_j \nsubseteq \beta_i)$"

	Características						
Táxons	1	.....	i	...	j	.....	n
...	...	.....	...	...	...	.....	...
τ_1	...	.....	1	...	0	.....	...
...	...	.....	...	...	...	.....	...
τ_2	...	.....	1	...	1	.....	...
...	...	.....	...	...	...	.....	...
τ_3	...	.....	0	...	1	.....	...
...	...	.....	...	...	...	.....	...

Observe que em relação à característica "i", τ_1 e τ_2 pertencem à subárvore "S_i" e $\tau_3 \notin S_i$; e em relação à característica "j" τ_2 e τ_3 pertencem à subárvore "S_j" e $\tau_1 \notin S_j$; o que é **absurdo** pois o táxon τ_2 não pode pertencer ao mesmo tempo a duas subárvores distintas ($i \neq j$). O absurdo se dá pela hipótese-1; logo, se a Filogenia é Perfeita, as condições do lema devem ser satisfeitas.

$\Leftarrow$ 1. Vamos assumir que $(\beta_i \cap \beta_j) = \varnothing$

Hipótese 2. *Suponhamos que* $\mathscr{M}$ **não** *corresponde a uma Filogenia Perfeita; logo* $\exists \tau$ *e* τ*' que contém a característica "i" e pertencem a subárvores distintas de A*; *s.p.g.*, *digamos que uma dessas subárvores seja* $S_j \neq S_i$, *com* $\tau' \in S_j$, *conforme Tabela 4.12*

Observe que neste caso $(\beta_i \cap \beta_j) \neq \varnothing$ o que é **absurdo** pelo que foi assumido, logo a hipótese-2 é falsa; ou seja, se $(\beta_i \cap \beta_j) = \varnothing$, a filogenia é perfeita (ver Figura 4.23-a).

2. *s.p.g.* vamos assumir que $(\beta_i \subseteq \beta_j)$

   Neste caso, a demonstração pode ser direta, da seguinte forma:

   (a) se $(\beta_i = \beta_j)$ então as subárvores são coincidentes $(S_i = S_j)$ pois para todos os táxons τ, em relação a estas características, temos $(m_{\tau i} = m_{\tau j})$; não existindo

Tabela 4.12 Matriz de características para hipótese-2

	Características						
Táxons	1	.....	i	...	j	.....	n
...	...	.....	...	...	...	.....	...
τ	...	.....	1	...	0	.....	...
...	...	.....	...	...	...	.....	...
τ'	...	.....	1	...	1	.....	...
...	...	.....	...	...	...	.....	...

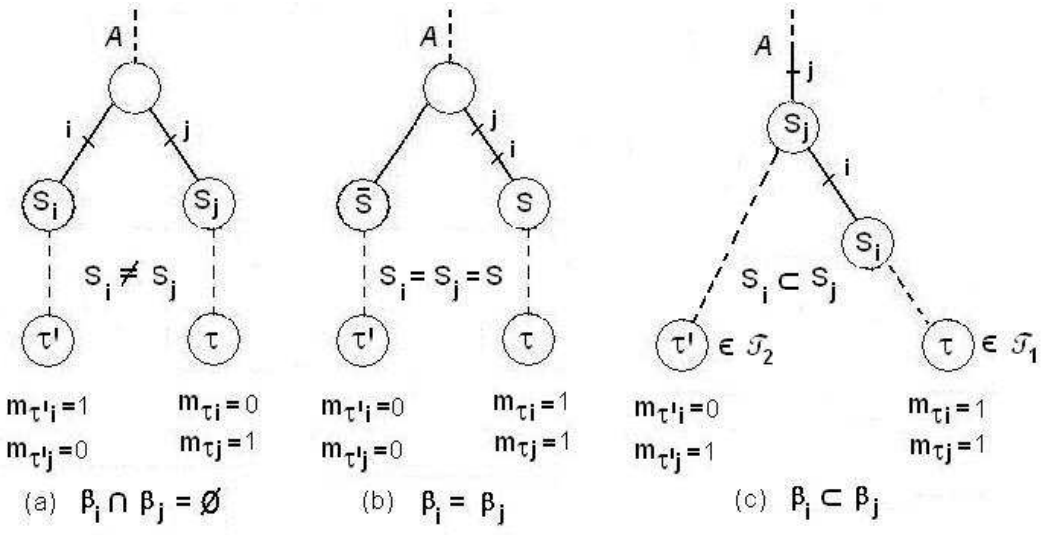


Figura 4.23 Situações que auxiliam a demonstração do Lema-1

portanto incompatibilidade, logo a Filogenia é Perfeita (ver Figura 4.23-b).

(b) se $(\beta_i \subset \beta_j)$ temos que $|\beta_i|_1 < |\beta_j|_1$, e podemos separar o conjunto de táxons $\mathscr{T}$ em duas partições (não vazias) tais que: $\mathscr{T}_1 \cup \mathscr{T}_2 = \mathscr{T}$ e $\mathscr{T}_1 \cap \mathscr{T}_2 = \varnothing$, com as seguintes propriedades:

i. $\forall \tau \in \mathscr{T}_1$, $m_{\tau i} = m_{\tau j}$ e o caso é igual ao do item-2a ($\beta_i = \beta_j$);

ii. $\forall \tau' \in \mathscr{T}_2$, $m_{\tau' i} = 0$ e $m_{\tau' j} = 1$ e o caso é igual ao do item-1 ($\beta_i \cap \beta_j = \varnothing$) (ver Figura 4.23-c).

□

O Algoritmo 3 apresentado a seguir verifica o Lema 1 e **decide** se uma dada instância admite uma filogenia perfeita. Facilmente observa-se que a ordem de complexidade [CLRS02, Via98] de tempo deste algoritmo é $\mathscr{O}(nm)$.

Algoritmo 3: Verifica se uma dada Matriz de Características Binárias corresponde a uma filogenia perfeita de acordo com o critério do Lema 1 (página 67) – algoritmo adaptado da referência [SM97, p. 184]

Input: (m, n, A)
A = matriz binária $T \times C$ com m táxons e n características; $T = \{1, \ldots, m\}$ e $C = \{1, \ldots, n\}$

Output: (Key)
True – indica que A corresponde a uma Filogenia Perfeita; Key = *false* – caso contrário

```
3.1  begin
3.2      c_j ← Σ_{i=1}^{m} a_ij     {vetor inteiro que indica quantos táxons (a_ij = 1) tem a característica j, ∀j ∈ C}
3.3      radix_sort(A)     {ordena a matriz A por colunas com mais 1s, com base no vetor c_j}
3.4      L_ij ← 0   ∀i ∈ T, ∀j ∈ C     {matriz auxiliar}
3.5      for i ← 1 to m do
3.6          k ← −1                    {k é a coluna mais a direita à esquerda de j tal que a_ik = 1}
3.7          for j ← 1 to n do
3.8              if a_ij = 1 then L_ij ← k;   k ← j
3.9          endfor
3.10     endfor
3.11     Key ← true
3.12     for j ← 1 to n do
3.13         maxval ← ||L[1 : m, j]||_∞
3.14         for i ← 1 to m do
3.15             if (L_ij ≠ 0) and (L_ij ≠ maxval) then Key ← false
3.16         endfor
3.17     endfor
3.18 end
```

Ao testar o Algoritmo 3 para as instâncias correspondentes às particões contidas na Tabela 4.9 obtivemos os resultados mostrados na Tabela 4.13.

Tabela 4.13 Resultados dos testes do Algoritmo 3 para as instâncias da Tabela 4.9 da página 62

Instância	Resultado	Comprovação
Υ	Filogenia Perfeita	Figura 4.16 da página 63
Ψ	Filogenia Perfeita	Figura 4.17 da página 63
$\Upsilon \cup \Psi$	Filogenia Perfeita	Figura 4.19-a da página 64
Γ	Não é filogenia perfeita	Figura 4.26-a da página 73
$\Upsilon \cup \Psi \cup \Gamma$	Não é filogenia perfeita	Figura 4.26-b da página 73

Para $\Gamma = \{8,9,10,11,12\}$, as colunas $(\beta_{11} \cap \beta_{12}) \equiv (0,1,1).and.(1,0,1) = (0,0,1) \neq \varnothing = (0,0,0)$; o que contraria o Lema 1, induzindo a uma "incongruência" entre as características 11 e 12.

Tomando como base que o subconjunto $\{8,9,10,11\} \subset \Gamma$ é idêntico ao conjunto Ψ teríamos a Figura 4.24-a semelhante a qualquer uma das configurações da Figura 4.17 da página 63.

Ignorando a instância Γ, os locais onde a característica "12" poderia aparecer seria um dos indicados na Figura 4.24-b, de modo que para que a instância Γ corresponda a uma "Filogenia Perfeita" os valores possíveis da coluna "12" seriam as indicadas na Tabela 4.14.

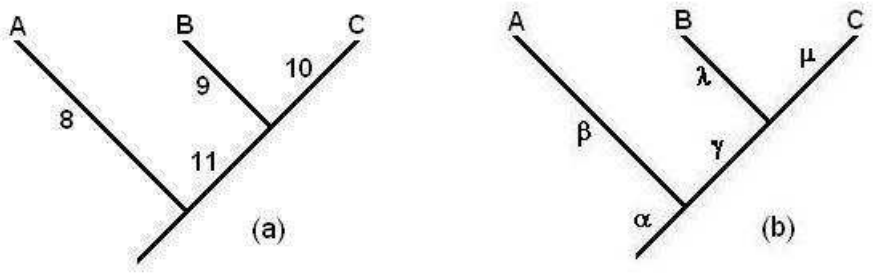


Figura 4.24 Árvore Filogenética para análise de inferência onde: a) Considera a eliminação da característica "12" (filogenia perfeita) b) Indica os possíveis locais para inserção da característica "12"

Tabela 4.14 Possíveis valores para a característica "12" para que a instância Γ da Tabela 4.9 corresponda a uma "Filogenia Perfeita"

Opção	β'_{12}	N. de *bits* de β'_{12} diferentes de $\beta_{12} = (1,0,1)$
α	(1,1,1)	1
β	(1,0,0)	1
γ	(0,1,1)	2
λ	(0,1,0)	3
μ	(0,0,1)	1

O número de *bits* distintos indicados na Tabela 4.14 corresponde ao número de modificações que deveriam ocorrer em cada espécie, ou seja, as espécies deveriam "ganhar" ($0 \rightarrow 1$) ou "perder" ($1 \rightarrow 0$) a característica "12". Na prática este procedimento não pode ser adotado pois haveria alteração da instância. A opção então de escolha deve considerar apenas as transformações evolutivas, onde as mudanças correspondentes a "ganhar" e "perder" correspondem às mutações ou mudança de estados.

Todas as variações possíveis estão mostradas na Figura 4.25 e de acordo com o critério da parcimônia, a opção α indicada é a mais provável de acontecer pois necessita de menor esforço (cinco passos), quando comparadas com as demais. A opção λ da Figura 4.24 não pode ser considerada pois o táxon "B" não contém a característica "12".

Por fim, apresentamos na Figura 4.26 uma Árvore Filogenética inferida correspondente à matriz de características da Tabela 4.9, ou seja, do conjunto $\Psi \cup \Upsilon \cup \Gamma$; poderíamos dizer que esta figura é uma "junção" das Figuras 4.19-a e 4.26-a.

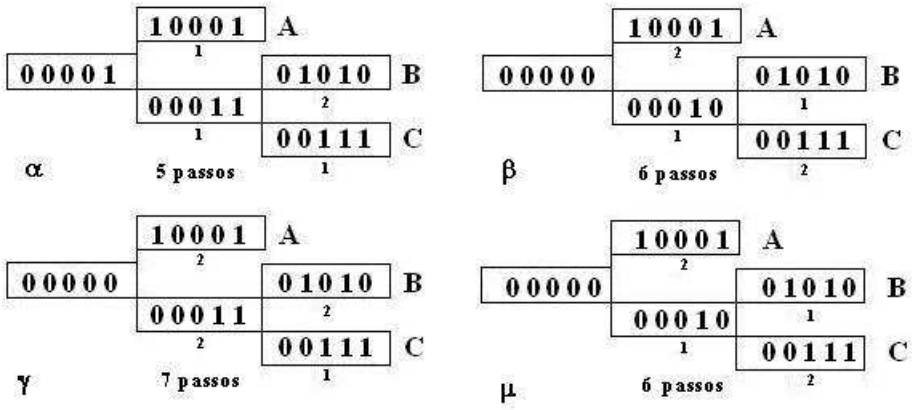


Figura 4.25 Possibilidades de inserção de uma nova característica e número de passos evolutivos nos locais indicados na Figura 4.24-b (α, β, γ e μ)

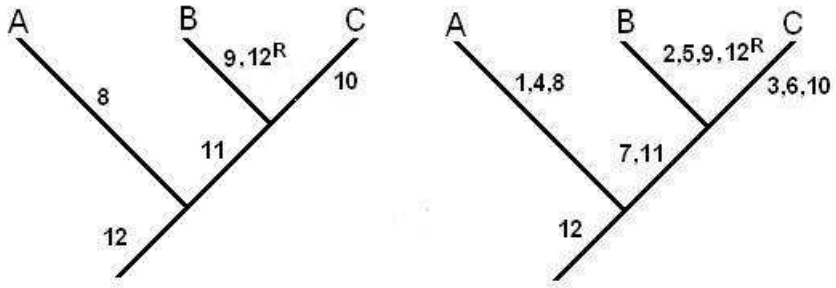


Figura 4.26 Árvore Filogenética inferida usando o critério da parcimônia para a matriz de características da Tabela 4.9 com cinco passos (aplicado para as possibilidades da Figura 4.25). a) considerando as cinco características do conjunto $\Gamma = \{8, 9, 10, 11, 12\}$. b) considerando todas as doze características ($\Psi \cup \Upsilon \cup \Gamma$)

CAPÍTULO 5

Construindo Filogenias Perfeitas

5.1 Estrutura de Dados para armazenar Árvores Filogenéticas

Um grafo $G = (V, E)$ é uma estrutura de dados formada por um conjunto V de vértices ou "nós" e um conjunto E de arestas que indicam as relações entre estes vértices; uma "árvore" é um grafo conexo sem ciclos [Boa03].

Numa árvore filogenética os vértices externos, ou "folhas", devem identificar as espécies ou táxons a serem analisados; os nós internos, identificam os ancestrais hipotéticos de cada subárvore, e a "raiz", o ancestral comum a todos os organismos representados na árvore. As arestas da árvore devem conter as características que os táxons que se encontram nos vértices descendentes possuem.

Para definir uma estrutura para armazenar uma árvore vamos considerar a Tabela 5.1 contendo uma matriz com três espécies e sete características correspondente à árvore filogenética da Figura 5.1.

Tabela 5.1 Exemplo de uma Matriz de Características cuja árvore correspondente está representada na Figura 5.1

	Características						
Espécies	1	2	3	4	5	6	7
1	0	0	0	0	0	1	0
2	0	1	1	0	0	0	1
3	1	1	1	1	1	0	0

Na árvore da Figura 5.1 identificamos os seguintes elementos:

- vértices terminais (táxons) $\rightarrow$ 1, 2 e 3;

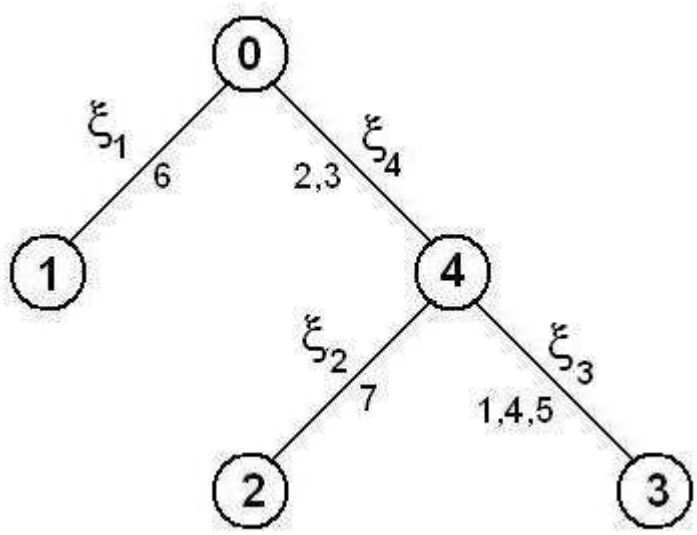


Figura 5.1 Árvore com quatro arestas e cinco vértices com identificação de três táxons (vértices 1, 2 e 3); duas espécies ancestrais (vértices 0 e 4); quatro arestas (ξ_1, ξ_2, ξ_3 e ξ_4) e sete características {1, 2, 3, 4, 5, 6, 7}

- vértice intermediário (ancestral recente) $\rightarrow 4$;
- vértice raiz (ancestral distante) $\rightarrow 0$;
- arestas $\rightarrow \xi_1$, ξ_2, ξ_3 e ξ_4
- conteúdo das arestas $\xi_1 = \{6\}$; $\xi_4 = \{2,3\}$; $\xi_2 = \{7\}$ *e* $\xi_3 = \{1,4,5\}$;

No caso em que a uma aresta "ξ_k" não esteja associada alguma característica seu conteúdo deve ser vazio, sendo representada por $\xi_k = \varnothing$ quando usada a notação de conjunto ou $\xi_k = 0$ na notação em binário. Conforme comentado na Seção 2.5 na página 18 as identificações dos nós intermediários, como o vértice "4" na Figura 5.1, não devem aparecer na árvore filogenética final; sua função no algoritmo de geração da árvore é de auxiliar na definição de sua estrutura, conexões entre os vértices e processamento.

Para consolidar a estrutura da árvore há necessidade de ser criado um vetor com "ponteiros" na forma $z_i =$j, indicando que o vértice "j" é antecessor do vértice "i". No exemplo da Figura 5.1 teríamos: $z_1 = 0$, $z_2 = 4$, $z_3 = 4$ e $z_4 = 0$.

As arestas ξ_i serão armazenadas como vetores binários cujos *bits* iguais a "1" indicam que a característica da respectiva posição está presente na aresta; a Tabela 5.2 mostra o conteúdo das arestas da árvore da Figura 5.1, onde a última linha seria o vetor de inteiros $\xi(j)$ cujo valor indica qual aresta contém a característica "j".

Tabela 5.2 Conteúdo de arestas usando vetores binários correspondente à árvore da Figura 5.1. A última linha contém o vetor que identifica em qual aresta cada característica se encontra

	Características						
Arestas	1	2	3	4	5	6	7
ξ_1	0	0	0	0	0	1	0
ξ_2	0	0	0	0	0	0	1
ξ_3	1	0	0	1	1	0	0
ξ_4	0	1	1	0	0	0	0
ξ	3	4	4	3	3	1	2

Veremos que a escolha desta estrutura para armazenar uma árvore filogenética terá grandes vantagens tanto na economia de espaço e na manipulação dos dados como na rapidez de execução dos processos, visto que serão realizadas operações binárias.

Outro fator importante a observar é a forma de abstração dos dados, onde a utilização de um único vetor de inteiros é possível armazenar toda a árvore filogenética, evitando o uso de listas, filas ou pilhas [SM94] que seriam as primeiras estruturas a se pensar em utilizar na manipulação de árvores.

Para as características presentes nas arestas foram utilizados vetores binários cujos *bits* iguais a "1" indicam que a característica correspondente à sua posição no vetor está presente na aresta; de modo que, com apenas dois vetores (Z e ξ) representamos qualquer árvore filogenética; assim, a árvore da Figura 5.1 seria definida pelos vetores: $Z = (0,4,4,0)$ e $\xi = (3,4,4,3,3,1,2)$.

A técnica desenvolvida para inferir filogenias, descrita no Capítulo 6, utiliza vetores na forma de Z. Especialmente para essa técnica será apresentada esta representação de dados conforme descrição na Seção 6.3, página 112.

5.2 Operações com Vetores Binários

Nesta seção iremos descrever quais as operações binárias utilizadas em nossos algoritmos de reconstrução de árvores filogenéticas. Para um melhor entendimento destas operações iremos considerar um exemplo de inclusão de um nó "k" numa aresta existente ξ_j de acordo com a Figura 5.2.

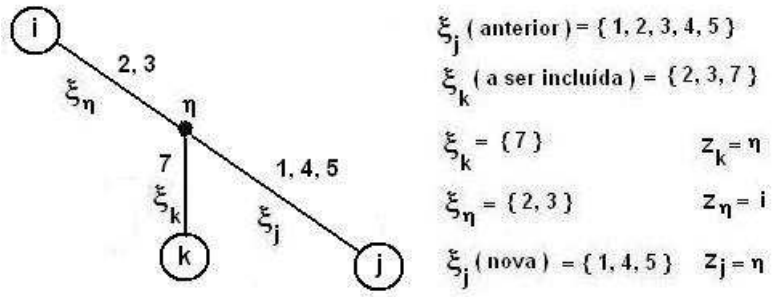


Figura 5.2 Exemplo de inclusão de um novo vértice numa aresta; a notação $z_j = i$ indica que o nó "i" precede o nó "j"

Podemos observar que os resultados após a inclusão do nó "k" podem ser obtidos a partir das seguintes operações com conjuntos e atribuição do vetor de ponteiros; na ordem:

$$\begin{array}{lllll}
\xi_\eta \leftarrow \xi_j \cap \xi_k & \Rightarrow & \xi_\eta = \{1,2,3,4,5\} \cap \{2,3,7\} & \Rightarrow & \xi_\eta = \{2,3\} \\
\xi_k \leftarrow \xi_k \backslash \xi_\eta & \Rightarrow & \xi_k = \{2,3,7\} \backslash \{2,3\} & \Rightarrow & \xi_k = \{7\} \\
\xi_j \leftarrow \xi_j \backslash \xi_\eta & \Rightarrow & \xi_j = \{1,2,3,4,5\} \backslash \{2,3\} & \Rightarrow & \xi_k = \{1,4,5\} \\
z_\eta \leftarrow i & & z_k \leftarrow \eta & & z_j \leftarrow \eta
\end{array}$$

Verificando a árvore da Figura 5.1 da página 76 constatamos que os resultados ali apresentados foram obtidos de forma semelhante através das seguintes operações correspondentes (no caso $i = 0$, $j = 3$, $k = 2$ e $\eta = 4$) :

$$\begin{array}{lll}
\xi_4 \leftarrow \xi_3 \cap \xi_2 & \Rightarrow \quad \xi_4 = \{1,2,3,4,5\} \cap \{2,3,7\} & \Rightarrow \quad \xi_4 = \{2,3\} \\
\xi_2 \leftarrow \xi_2 \backslash \xi_4 & \Rightarrow \quad \xi_2 = \{2,3,7\} \backslash \{2,3\} & \Rightarrow \quad \xi_2 = \{7\} \\
\xi_3 \leftarrow \xi_3 \backslash \xi_4 & \Rightarrow \quad \xi_3 = \{1,2,3,4,5\} \backslash \{2,3\} & \Rightarrow \quad \xi_2 = \{1,4,5\} \\
z_4 \leftarrow 0 & \quad z_2 \leftarrow 4 & \quad z_3 \leftarrow 4
\end{array}$$

Usando a notação β_j para indicar o vetor binário contendo as características da aresta "j", de acordo com a Tabela 5.2, teríamos as seguintes operações binárias a serem utilizada nos algoritmos correspondentes às operações anteriores com conjuntos.

$$\begin{array}{lll}
\beta_4 \leftarrow \beta_3.and.\beta_2 & \Rightarrow \quad \beta_4 = (1,1,1,1,1,0,0).and.(0,1,1,0,0,0,1) & \\
& \Rightarrow \quad \beta_4 = (0,1,1,0,0,0,0) \quad \equiv \quad \xi_4 = \{2,3\} & \\
\beta_2 \leftarrow \beta_2.and.(not(\beta_4)) & \Rightarrow \quad \beta_2 = (0,1,1,0,0,0,1).and.(1,0,0,1,1,1,1) & \\
& \Rightarrow \quad \beta_2 = (0,0,0,0,0,0,1) \quad \equiv \quad \xi_2 = \{7\} & \\
\beta_3 \leftarrow \beta_3.and.(not(\beta_4)) & \Rightarrow \quad \beta_3 = (1,1,1,1,1,0,0).and.(1,0,0,1,1,1,1) & \\
& \Rightarrow \quad \beta_2 = (1,0,0,1,1,0,0) \quad \equiv \quad \xi_3 = \{1,4,5\} &
\end{array}$$

Outra operação com conjuntos presente nos algoritmos é $\xi_i \leftarrow \xi_j \cup \xi_k$ que corresponde, na notação aqui utilizada, à operação binária: $\beta_i \leftarrow \beta_j$ *.or.* β_k.

Nos algoritmos apresentados neste capítulo, utilizamos a notação ξ_j para representar o vetor binário β_j.

5.3 Eliminação e Inclusão de Táxons com características comuns

Táxons com características comuns estarão dispostos num mesmo ramo da árvore filogenética que os relaciona. Por exemplo, sejam os táxons A com características $\{1,2,3\}$ e B com características $\{1,2\}$, comuns a A ($B \subseteq A$). A aresta $\xi_A =< X,A >$ da Figura 5.3-a seria uma ramificação da árvore filogenética representativa desta situação, enquanto que a aresta $\xi_B =< \eta,B >$ seria incluída ao final do processo através das operações: $\xi_\eta = B$, $\xi_B = \varnothing$ e $\xi_A = A \backslash B$; conforme

configuração representada na Figura 5.3-b.

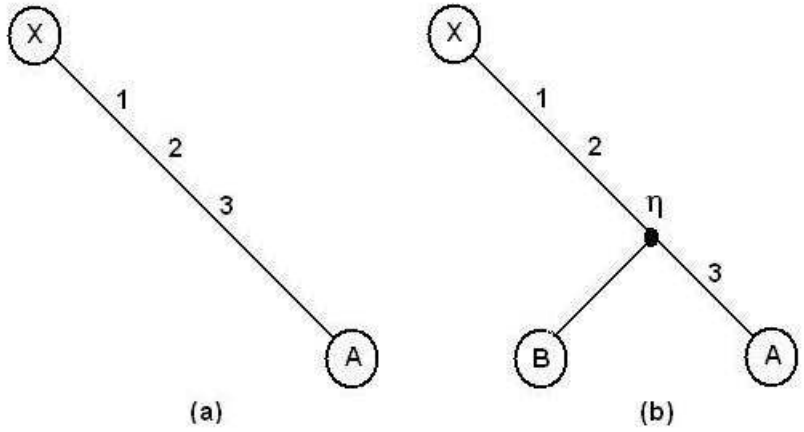


Figura 5.3 Situação em que dois táxons têm características comuns. a) Considerando somente o táxon A b) Incluindo o táxon B

Como $B \subseteq A$ será desconsiderado, ou seja, eliminado da instância e mantido somente o táxon A, há necessidade dos dois procedimentos apresentados a seguir para realizar essas operações. Um (algoritmo 4) para **eliminar** ou retirar B da instância, e outro (algoritmo 5) para **incluir** B "de volta" na aresta da árvore final que contém A.

5.4 Algoritmo para reconstruir Filogenias Perfeitas

A entrada do algoritmo para reconstruir filogenias, aqui apresentado, é uma matriz $\mathscr{M}$ de ordem $m \times n$ correspondente ao conjunto $\mathscr{T}$ com **m** táxons e **n** características, onde cada elemento m_{ij} da matriz contém um dos valores dos **r** estados que cada uma das característica pode assumir, ou seja, $m_{ij} \in \{0, \ldots, r-1\}$; para os testes computacionais apresentados na Seção 5.6, iremos considerar $r = 2$ ou seja, matrizes de estados binários ($m_{ij} \in \{0, 1\}$).

Para a fase de construção da filogenia perfeita, apresentamos o algoritmo com seus passos descritos formalmente a seguir, onde se observa que sua complexidade no pior caso [GJ79, CLRS02] é da ordem $O(m^2 n)$ correspondente à complexidade do Algoritmo 6 que é o de maior complexidade entre os módulos ou sub-algoritmos chamados.

Algoritmo 4: Elimina táxons que contém todas características comuns a um outro

Input: (m, n, A)
$A =$ matriz binária $T \times C$ com m táxons e n características; $T = \{1, \ldots, \text{m}\}$ e $C = \{1, \ldots, \text{n}\}$

Output: (T_1, T_2) $\{Obs: T_1 \text{ e } T_2 \text{ são partições de T }\}$
$T_1 =$ conjunto contendo m_1 táxons eliminados
$T_2 =$ conjunto de entrada alterado contendo $(m - m_1)$ táxons restantes

```
4.1  begin
4.2      t_i ← A[i, 1 : n] {vetor binário contendo as características do táxon i, ∀i ∈ T}
4.3      T_1 ← ∅
4.4      m_1 ← 0
4.5      for i ← 1 to m − 1 do
4.6          j ← i + 1
4.7          if i ∉ T_1 then
4.8              while j ≤ m do
4.9                  if t_i ⊆ t_j then
4.10                     T_1 ← T_1 ∪ {i};      m_1 ← m_1 + 1
4.11                 else
4.12                     if t_j ⊆ t_i then
4.13                         T_1 ← T_1 ∪ {j};      m_1 ← m_1 + 1
4.14                     endif
4.15                 endif
4.16                 j ← j + 1
4.17             endw
4.18         endif
4.19     endfor
4.20     T_2 ← T \ T_1
4.21 end
```

Algoritmo 5: Inclui táxons com características comuns eliminados pelo Algoritmo 4

Input: (T_1, t_i, Z, E, η)
T_1 = conjunto de entrada obtido do Algoritmo 4
t_i = vetores binários contendo as características do táxon i, $\forall i \in T_1$
Z, E = vetores de entrada gerados pelo Algoritmo 8
η = indexador do último vértice incluído pelo Algoritmo 8

Output: (Z, E)
Z, E = vetores que identificam a árvore final $\mathscr{T}$

```
5.1  begin
5.2      for j ← 1 to m do
5.3          h ← T1(j)
5.4          if h > 0 then
5.5              Key ← false
5.6              while not(Key) do
5.7                  ξ ← tj ∩ th
5.8                  Key ← (ξ ≠ 0)
5.9                  if not(Key) then h ← zh
5.10             endw
5.11             η ← η + 1
5.12             ξη ← ξ;          zη ← zh
5.13             ξh ← th \ ξ;     zh ← η
5.14             ξj ← 0;          zj ← η
5.15         endif
5.16     endfor
5.17 end
```

Após a leitura dos dados o processamento é iniciado com a eliminação dos táxons que têm características comuns com o objetivo de eliminar as redudâncias e minimizar o tempo de execução. A construção da filogenia propriamente dita é feita após a certificação de que a instância lida corresponde a uma filogenia perfeita, partindo de um único vértice (raiz da árvore) onde serão inicialmente incluídos aqueles táxons que possuam características disjuntas.

Para cada um dos táxons ainda não considerados, são selecionados, na ordem, aqueles com maior número de características que são incluídos numa aresta já existente ou uma nova a ser criada a partir da raiz de alguma sub-árvore. Ao final, os táxons que foram retirados inicialmente por possuirem características comuns são incluídos em sua devida posição.

A saída do algoritmo será uma "*string*", ou uma cadeia de caracteres contendo a árvore no formato *Newick* [NEW07]; tal forma de representação da árvore filogenética resultante servirá de entrada para desenhá-la utilizando o Algoritmo 2 (página 53).

O Algoritmo 6, descrito a seguir, constrói uma árvore filogenética perfeita.

Algoritmo 6: Reconstrução de Filogenias Perfeitas baseadas em características. A notação ALGOx corresponde uma chamada ao Algoritmo "x"

Input: (m, n, A)
$A =$ matriz $T \times C$ com m táxons e n características
$T = \{1, \ldots, \text{m}\}$ e $C = \{1, \ldots, \text{n}\}$

Output: (S, E) {árvore filogenética final $\mathscr{T}$}
S = *string* no formato padrão *Phylip* correspondente à solução
E = $\{\xi_1, \ldots, \xi_\eta\}$ = vetores ξ_i contendo a identificação das características da aresta i, $\forall i \in \{1, \ldots, \eta\}$

```
6.1  begin
6.2      Key ← ALGO3(m,n,A)
6.3          {Verifica se A_{m×n} corresponde a uma árvore filogenética perfeita}
6.4      if Key then
6.5          {T1, T2} ← ALGO4(m, n, A)          {Elimina táxons com características comuns }
6.6          {T2, Z, E} ← ALGO7(T2)             {Inclui táxons disjuntos na raiz da árvore }
6.7          {Z, E, η} ← ALGO8(T2, Z, E, m)     {Inclui demais táxons na árvore }
6.8          {Z, E} ← ALGO5(T1, Z, E, η)        {Inclui táxons retirados em A2 }
6.9          S ← ALGO9(Z)                       {Gera string a partir do vetor Z }
6.10         T ← ALGO2(S, E)                    {Desenha a árvore filogenética final }
6.11     else
6.12         {Características Incongruentes → Inferir AF }
6.13     endif
6.14 end
```

Os três algoritmos seguintes (7, 8 e 9) são subalgoritmos do anterior (6) e o Algoritmo 10 é

subalgoritmo do Algoritmo 5.

Algoritmo 7: Inclui táxons disjuntos na raiz da árvore

Input: (T_2, t_i)
T_2 = conjunto de entrada obtido do Algoritmo 4
t_i = vetores binários contendo as características do táxon i, $\forall i \in T_2$

Output: (T_2, Z, E)
T_1 = conjunto de táxons inseridos na origem
T_2 = conjunto de entrada alterado sem os táxons inseridos na origem
Z = vetor inteiro que identifica os vértices incluídos na raiz da árvore $\mathscr{T}$
$E = \{ \xi_1, \xi_2, \ldots \xi_\eta \}$ vetores binários que identificam as arestas incluídas em $\mathscr{T}$

```
7.1  begin
7.2      η ← 0
7.3      T1 ← 0
7.4      repeat
7.5          k ← {i, max||ti||1, ∀i ∈ T2}
7.6          for j ← 1 to η do
7.7              l ← T1(j)
7.8              if tk ∩ tl ≠ ∅ then exit
7.9          endfor
7.10         if j > η then
7.11             η ← η + 1
7.12             T1(η) ← k
7.13             ξk ← tk
7.14             T2 ← T2 \ {k}
7.15             zk ← 0
7.16         endif
7.17     until (T2 = ∅);
7.18 end
```

5.5 Procedimentos para execução do Algoritmo de Reconstrução

5.5.1 Passos para construir uma filogenia perfeita

1. Entrada de Dados (ler uma instância do problema com m táxons);

2. Verifica se a instância corresponde a uma filogenia perfeita;

3. Se o item-2 é “verdadeiro” continua, senão, “fim de execução”;

4. Elimina m' táxons com características comuns a outro;

Algoritmo 8: Inclui táxons simples na estrutura geral da árvore

Input: (T_2, t_i, Z, E, m)
T_2 = conjunto de entrada obtido do Algoritmo 7
t_i = vetores binários contendo as características do táxon i, $\forall i \in T_2$
Z = vetor inteiro que identifica os vértices da árvore $\mathscr{T}$
E = conjunto de vetores binários que identificam o conteúdo das arestas de $\mathscr{T}$
m = número de táxons da matriz original T

Output: (Z, E, η)
Z, E = vetores de entrada atualizados
η = indexador do último vértice incluído em $\mathscr{T}$

```
8.1  begin
8.2      η ← m
8.3      while T2 ≠ ∅ do
8.4          select (j from T2) and (k from E) such that (tj ∩ ξk) ≠ 0
8.5          η ← η + 1
8.6          ξη ← tj ∩ ξk;     zη ← zk
8.7          ξj ← tj \ ξη;     zj ← η
8.8          E ← A8(E, Z, j)                    {Elimina características repetidas}
8.9          ξk ← ξk \ ξη;     zk ← η
8.10         T2 ← T2 \ {j}
8.11     endw
8.12 end
```

5. Identifica m'' táxons "disjuntos entre si";

6. Inclui estes táxons diretemanente na raiz;

7. Inclui, um a um, os $(m - m' - m'')$ demais táxons;

8. Inclui m' táxons com características comuns eliminados no item-4

9. Gera a *string* correspondente à árvore final no formato *Newick*

5.5.2 Estruturas de dados de Entrada

1. $\mathscr{A}$ = matriz com m = | $\mathscr{T}$ | táxons e n = | $\mathscr{C}$ | características, com $m_{ij} \in \{0, 1\}$;

2. $\mathscr{T}$={1, . . . ,m} - conjunto de táxons;

3. $\mathscr{C}$={1, . . . ,n} - conjunto de características;

4. $t_i = \mathscr{A}\,[i, 1 : n]$ - vetor binário indicando as características do táxon i, $\forall i \in \mathscr{T}$;

Algoritmo 9: Gera *string* no formato padrão a partir do vetor Z obtido pelo Algoritmo 6

Input: (Z, m, η)
$Z =$ vetor inteiro que identifica os vértices da árvore $\mathscr{T}$
$m =$ número de táxons da matriz original T
$\eta =$ número de vértice da árvore $\mathscr{T}$

Output: (S)
S *string* que representa a árvore $\mathscr{T}$
w tabela dinâmica contendo "ponteiros" de cada vértice da árvore $\mathscr{T}$

```
9.1  begin
9.2      w(i, 0:2) ← [2, −1, −2]    {inicializa matriz dinâmica, ∀i ∈ {0,...,η}}
9.3      for i ← 1 to η do
9.4          w(z_i, 0) ← w(z_i, 0) + 1
9.5          k ← w(z_i, 0)
9.6          w(z_i, k) ← w(z_i, k − 1)
9.7          w(z_i, k − 1) ← i
9.8      endfor
9.9      for j ← 0 to m do r(j) ← w(0, j)
9.10     while ∃ r(j) > m, ∀j ← 1 : r(0) do
9.11         k ← w(r(j), 0);    r(0) ← r(0) + k
9.12         r(j + k : r(0)) ← r(j + 1 : r(0) − k)
9.13         r(j : j + k − 1) ← w(r(j), 1 : k)
9.14     endw
9.15     for i ← 1 to r(0) do
9.16         if r(i)=-1 then s(i) ← "("
9.17         if r(i)=-2 then s(i) ← ")" else s(i) ← r(i)
9.18     endfor
9.19 end
```

Algoritmo 10: Atualiza aresta ξ_j eliminando características repetidas

Input: $(E,\ Z,\ j)$
$E =$ conjunto de vetores binários que identificam o conteúdo das arestas de $\mathscr{T}$
$Z =$ vetor inteiro que identifica os vértices da árvore $\mathscr{T}$
$j =$ aresta $\xi_j \in E$ a ser atualizada

Output: (E)
$E =$ conjunto de arestas de $\mathscr{T}$ com aresta ξ_j atualizada

```
10.1 begin
10.2     k ← z_j
10.3     while k ≠ 0 do
10.4         ξ_j ← ξ_j \ ξ_k
10.5         k ← z_k
10.6     endw
10.7 end
```

5. $c_j = \mathscr{A}[1:m, j]$ - vetor binário indicando quais táxons possuem a característica j, $\forall j \in \mathscr{C}$.

5.5.3 Estruturas de dados de Saída

1. $E = \{\xi_1, \ldots, \xi_\eta\}$ - conjunto que indica o conteúdo das arestas da árvore final;

2. $Z = \{z_1, \ldots, z_\eta\}$ - vetor contendo a origem de cada aresta;

3. S= cadeia de caracteres contendo a estrutura de árvore no formato *Phylip*.

5.6 Simulação do Algoritmo de Reconstrução de Filogenia Perfeita

5.6.1 Exemplo de Entrada

Tabela 5.3 Matriz binária com indicação de $m = 7$ táxons e $n = 12$ características

$A =$

	$\mathbf{c_1}$	$\mathbf{c_2}$	$\mathbf{c_3}$	$\mathbf{c_4}$	$\mathbf{c_5}$	$\mathbf{c_6}$	$\mathbf{c_7}$	$\mathbf{c_8}$	$\mathbf{c_9}$	$\mathbf{c_{10}}$	$\mathbf{c_{11}}$	$\mathbf{c_{12}}$	*Taxon*
$\mathbf{t_1} \rightarrow$	0	0	0	0	0	0	0	1	0	0	0	0	*A*
$\mathbf{t_2} \rightarrow$	0	1	0	0	0	1	0	0	0	0	0	0	*B*
$\mathbf{t_3} \rightarrow$	1	0	0	1	0	0	0	0	0	0	0	0	*C*
$\mathbf{t_4} \rightarrow$	1	0	1	1	1	0	0	0	0	0	0	0	*D*
$\mathbf{t_5} \rightarrow$	1	0	0	0	0	0	0	0	1	1	0	0	*E*
$\mathbf{t_6} \rightarrow$	1	0	0	0	0	0	1	0	0	1	1	1	*F*
$\mathbf{t_7} \rightarrow$	1	0	0	0	0	0	1	0	0	1	1	0	*G*

5.6.2 Passos de Execução do Algoritmo de Reconstrução

1. *Verifica se a instância corresponde a uma filogenia perfeita* (Algoritmo 3)

   Para o exemplo das Tabelas 5.3 / 5.4 temos a entrada:

Tabela 5.4 Formato do arquivo de entrada para os testes dos algoritmos; a primeira linha contém: os valores de *m* (sete táxons) e *n* (doze caracteres) e as linhas seguintes contêm: o nome do táxon, a quantidade de características que ele possui e quais são estas características (exemplo correspondente à Tabela 5.3)

7	12					
A	1	8				
B	2	2	6			
C	2	1	4			
D	4	1	3	4	5	
E	3	1	9	10		
F	5	1	7	10	11	12
G	4	1	7	10	11	

Conjunto de táxons: $\mathscr{T}$={1, 2, 3, 4, 5, 6, 7}

Conjunto de características: $\mathscr{C}$={1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12}

Vetores binários: $c_j, \forall j \in \mathscr{C}$

De acordo com o Lema 1 (página 67) esta instância corresponde a uma filogenia perfeita (Key = *true*) pois $(c_k \cap c_j = \varnothing)$ ou $(c_k \subset c_j)$ ou $(c_k \supset c_j)$ $\forall k \neq j \in \mathscr{C}$.

2. *Elimina táxons com características comuns* (Algoritmo 4)

Entrada: T = $\mathscr{T}$={1, 2, 3, 4, 5, 6, 7}

Táxons com características comuns: $C \subseteq D$ e $G \subseteq F$.

Ao eliminar os táxons C(t_3) e G(t_7), definem-se os conjuntos:

Saída: $T_1 = \{3,7\}$ (táxons excluídos) e $T_2 = \{1,2,4,5,6\}$ (táxons restantes);

Para efeito de implementação estes conjuntos foram representados como vetores:

Vetores	1	2	3	4	5	6	7
T_1	0	0	4	0	0	0	6
T_2	1	1	0	1	1	1	0

Os valores $T_1(3) = 4$ e $T_1(7) = 6$ indicam que $t_3 \subseteq t_4$ e $t_7 \subseteq t_6$; essa informação será útil para o Algoritmo 5.

O valor $T_2(j) = 1$ indica que o táxon "j" ainda deve ser analisado pelo Algoritmo 7 e $T_2(j) = 0$ correponde ao respectivo táxon de T_1.

3. *Inclui táxons disjuntos na raiz da árvore* (Algoritmo 7)

Entrada: T_2 ={1, 2, 4, 5, 6 }

Táxons disjuntos entre si: A (t_1), B(t_2) e F(t_6) eliminados de T_2 ; (Obs.:a seleção dos táxons disjuntos é feita por ordem decrescente do número de suas características).

Saída: Vetores T_2 ={4, 5 } e Z (apontadores) e conjunto das arestas $E = \{\xi_1, \xi_2, \xi_6\}$

Vetores	1	2	3	4	5	6	7
T_2	0	0	0	1	1	0	0
Z	0	0	-1	-1	-1	0	-1

Arestas	1	2	3	4	5	6	7	8	9	10	11	12
ξ_1	0	0	0	0	0	0	0	1	0	0	0	0
ξ_2	0	1	0	0	0	1	0	0	0	0	0	0
ξ_6	1	0	0	0	0	0	1	0	0	1	1	1

O valor $Z(j) = 0$ indica que o táxon j foi incluído na raiz da árvore (ponteiro para o vértice "0") e $Z(j) = -1$ indica que o táxon "j" ainda não foi incluído. A Figura 5.4 mostra a árvore parcial; daqui em diante, após a execução de cada algoritmo será mostrada graficamente a evolução da árvore durante sua construção passo a passo.

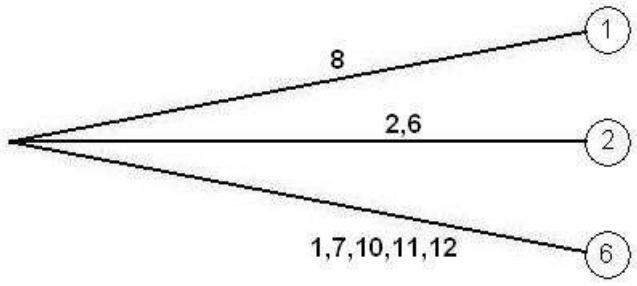


Figura 5.4 Árvore parcial da simulação após execução do Algoritmo 7; as identificações nas arestas correspondem às características de cada táxon incluído (ver Tabela 5.4)

4. *Inclui táxons simples na árvore* (Algoritmo 8)

Entrada: $T_2 = \{4,5\};\ Z = (0,0,-1,-1,-1,0,-1);\ E = \{\xi_1, \xi_2, \xi_6\};\ \eta = 7$

(a) $T_2 = \{4,5\} \neq \varnothing$

$t_4 \cap t_6 \neq \varnothing \rightarrow j = 4\ ,\ k = 6 \quad e \quad \eta = 8$

$\xi_8 \leftarrow t_4 \cap \xi_6 = \{1,3,4,5\} \cap \{1,7,10,11,12\} = \{1\}; \quad z_8 \leftarrow z_6 = 0$

$\xi_4 \leftarrow t_4 \setminus \xi_8 = \{1,3,4,5\} \setminus \{1\} = \{3,4,5\}; \quad z_4 \leftarrow 8$

$E \leftarrow A8(E,Z,4) \rightarrow \xi_4 = \{3,4,5\}$

$\xi_6 \leftarrow \xi_6 \setminus \xi_8 = \{1,7,10,11,12\} \setminus \{1\} = \{7,10,11,12\}; \quad z_6 \leftarrow 8$

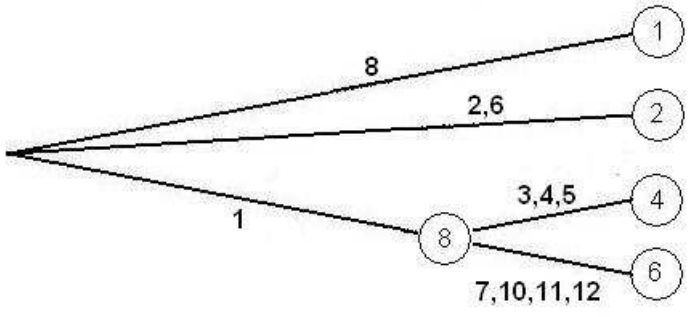


Figura 5.5 Saída do Algoritmo 8 após a inclusão do táxon "4"

(b) $T_2 = \{5\} \neq \varnothing$

$t_5 \cap t_6 \neq \varnothing \rightarrow j = 5\ ,\ k = 6 \quad e \quad \eta = 9$

$\xi_9 \leftarrow t_5 \cap \xi_6 = \{1,9,10\} \cap \{7,10,11,12\} = \{10\}; \quad z_9 \leftarrow z_6 = 0$

$\xi_5 \leftarrow t_5 \setminus \xi_9 = \{1,9,10\} \setminus \{10\} = \{1,9\}; \quad z_5 \leftarrow 9$

$E \leftarrow A8(E,Z,5) \rightarrow \xi_5 = \{9\}$

$\xi_6 \leftarrow \xi_6 \setminus \xi_9 = \{7,10,11,12\} \setminus \{10\} = \{7,11,12\}; \quad z_6 \leftarrow 9$

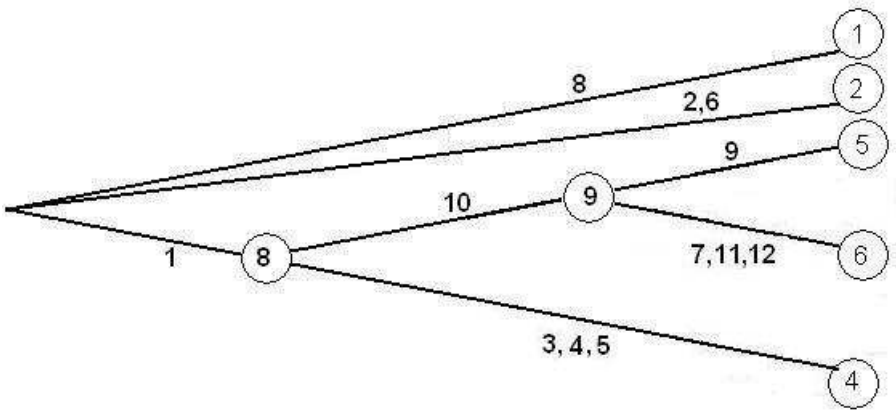


Figura 5.6 Saída do Algoritmo 8 após a inclusão do táxon "5"

(c) $T_2 = \varnothing \rightarrow$ {fim de execução do Algoritmo 8}

Saída: Z (apontadores) e conjunto das arestas $E = \{\xi_1, \xi_2, \xi_4, \xi_5, \xi_6, \xi_8, \xi_9\}$

Vetor	1	2	3	4	5	6	7	8	9
Z	0	0	-1	8	9	9	-1	0	8

Arestas	1	2	3	4	5	6	7	8	9	10	11	12
ξ_1	0	0	0	0	0	0	0	1	0	0	0	0
ξ_2	0	1	0	0	0	1	0	0	0	0	0	0
ξ_4	0	0	1	1	1	0	0	0	0	0	0	0
ξ_5	0	0	0	0	0	0	0	0	0	0	0	1
ξ_6	0	1	0	0	0	1	0	0	0	0	1	1
ξ_8	1	0	0	0	0	0	0	0	0	0	0	0
ξ_9	0	0	0	0	0	0	0	0	0	1	0	0

5. *Inclui táxons com características comuns excluídos pelo Algoritmo 4* (Algoritmo 5)

   Entrada:

   $\eta = 9$

   $T_1 = \{3,7\}$ $Z = (0,0,-1,8,9,9,-1,0,8)$ $E = \{\xi_1, \xi_2, \xi_4, \xi_5, \xi_6, \xi_9, \xi_{10}\}$

   (a) $j = 3 \rightarrow h = 4 > 0$

   $\xi = t_3 \cap t_4 = \{1,4\} \cap \{3,4,5\} = \{4\} \neq \varnothing \rightarrow \eta = 10$

   $\xi_{10} \leftarrow \xi = \{4\}; \quad z_{10} \leftarrow z_4 = 8$

   $\xi_4 \leftarrow t_4 \setminus \xi = \{3,4,5\} \setminus \{4\} = \{3,5\}; \quad z_4 \leftarrow 10$

   $\xi_3 \leftarrow 0; \quad z_3 \leftarrow 10$

   (b) $j = 7 \rightarrow h = 6 > 0$

   $\xi = t_7 \cap t_6 = \{1,7,10,11\} \cap \{7,11,12\} = \{7,11\} \neq \varnothing \rightarrow \eta = 11$

   $\xi_{11} \leftarrow \xi = \{7,11\}; \quad z_{11} \leftarrow z_6 = 9$

   $\xi_6 \leftarrow t_6 \setminus \xi = \{7,11,12\} \setminus \{7,11\} = \{12\}; \quad z_6 \leftarrow 11$

   $\xi_7 \leftarrow 0; \quad z_7 \leftarrow 11$

6. *Gera string no formato padrão Newick* (Algoritmo 9)

   Entrada:

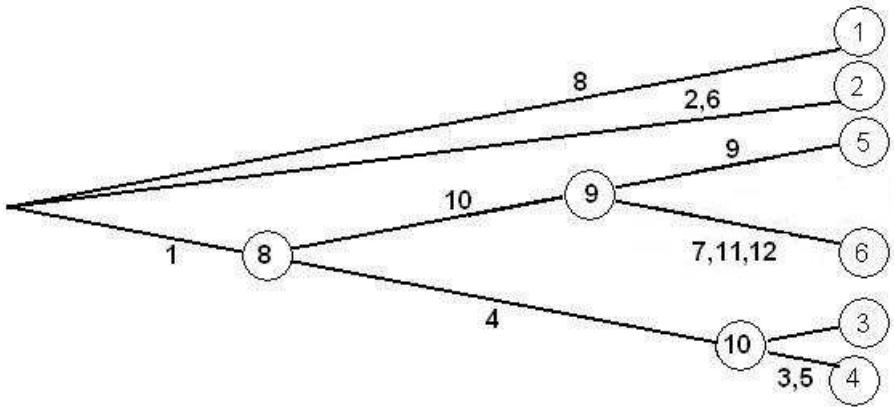


Figura 5.7 Saída do Algoritmo 5 após a inclusão do táxon "3"

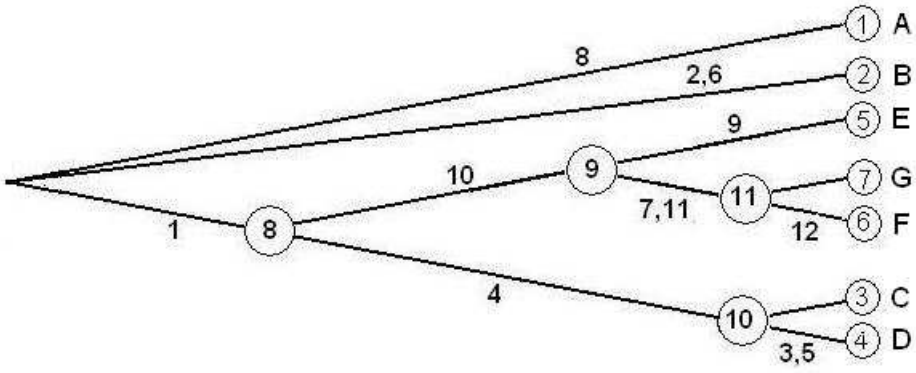


Figura 5.8 Saída do Algoritmo 5 após a inclusão do táxon "7"

Vetor	1	2	3	4	5	6	7	8	9	10	$\eta = 11$
Z	0	0	10	10	9	11	11	0	8	8	9

A execução deste algoritmo gera a seguinte a tabela dinâmica mostrada na Figura 5.9 e árvore final na Figura 5.10.

Nó	Ponteiros
0	(1 2 8)
1	( )
2	( )
3	( )
4	( )
5	( )
6	( )
7	( )
8	(9 10)
9	(5 11)
10	(3 4)
11	(6 7)

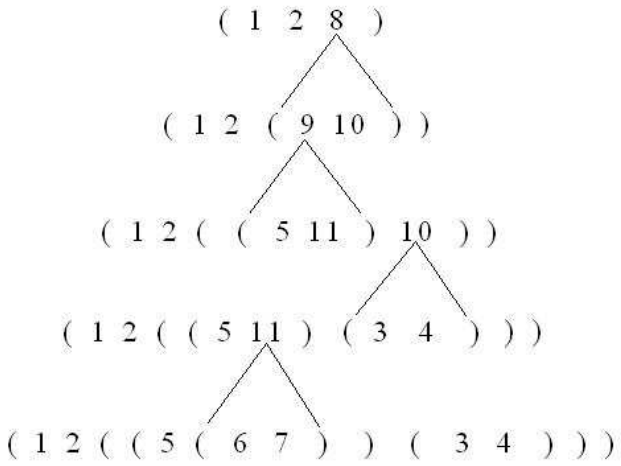


Figura 5.9 Tabela Dinâmica auxiliar usada pelo Algoritmo 9 para gerar a *string* que define a árvore final; esta será a entrada para o Algoritmo 2

Saída:

String $S = (1,2,((5,(6,7)),(3,4))) = (A,B,((E,(F,G)),(C,D)))$;

Características	1	2	3	4	5	6	7	8	9	10	11	12
Arestas (ξ_i)	8	2	4	10	4	2	11	1	5	9	11	6

Alguns algoritmos apresentados neste capítulo estão simulados aqui de forma indireta, visto que o Algoritmo 6 é o algoritmo principal que chama os demais; o Algoritmo 10 é chamado pelo algoritmo 5 e o 1 pelo 2 que desenha a árvore filogenética final representada na Figura 5.10. Para validação de todos estes algoritmos foram feitos outros testes computacionais cujos resultados estão mostrados na seção seguinte.

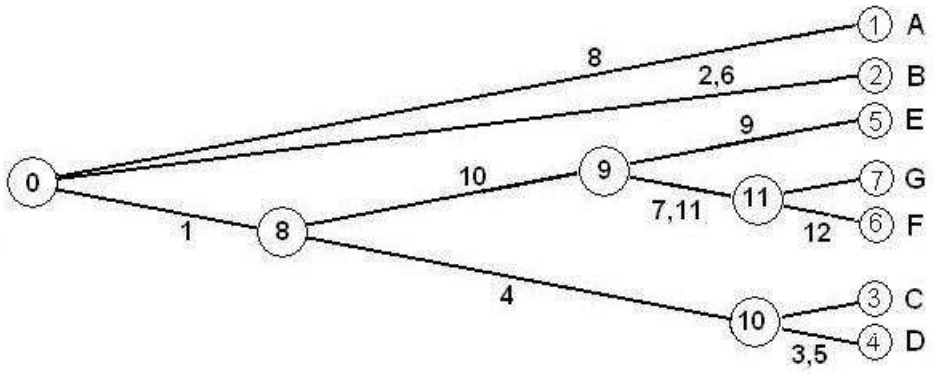


Figura 5.10 Árvore final da simulação gerada pelo Algoritmo 2

5.7 Experimentos Computacionais I

5.7.1 Modelo das Instâncias

Conforme visto na seção anterior, o modelo de entrada para nossos algoritmos é aquele apresentado na Tabela 5.4, página 88, correspondente a uma matriz binária; nos nossos testes optamos por não mostrar essas matrizes por questão de espaço e ilegibilidade; assim, as colunas iniciais da Tabela 5.5 será a forma utilizada para apresentação das instâncias.

Tabela 5.5 Modelo de uma instância e matriz binária correspondente

4	5				
t_1	2	1	3		
t_2	1	4			
t_3	4	1	2	3	5
t_4	2	1	4		

$\rightarrowtail$ Matriz Binária $\rightarrowtail$

Táxon	c_1	c_2	c_3	c_4	c_5
Alpha	1	0	1	0	0
Beta	0	0	0	1	0
Gamma	1	1	1	0	1
Delta	1	0	0	1	0

5.7.2 Resultados dos Testes Computacionais

1. Instância: m=8, n=10 $\longmapsto$ Figura 5.11

2. Instância: m=12, n=15 $\longmapsto$ Figura 5.12

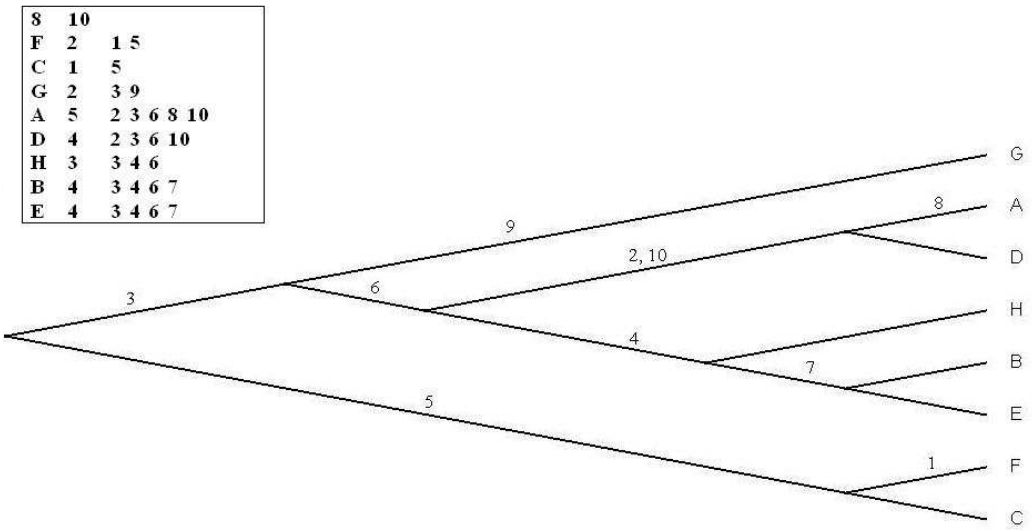


Figura 5.11 Árvore filogenética com m=8 táxons e n=10 características

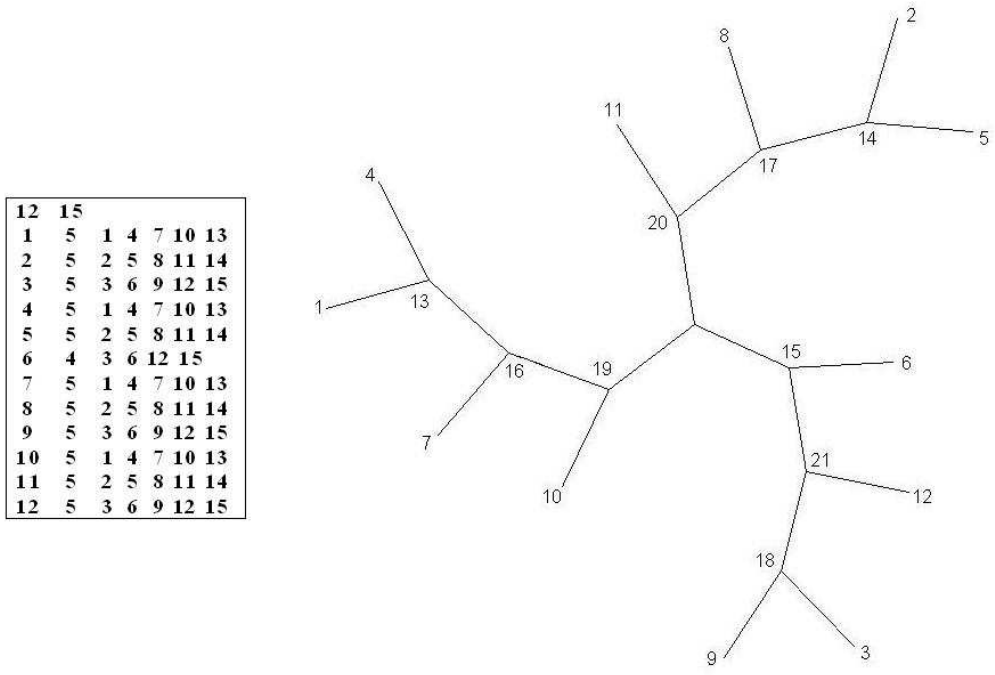


Figura 5.12 Árvore filogenética sem raiz com m=12 táxons e n=15 características, sem identificação do conteúdo das arestas

3. Instância: m=23, n=30 $\longmapsto$ Figura 5.13

23	30	
1	14	2 5 6 9 13 14 15 17 18 19 21 23 25 27
2	10	1 4 5 6 8 9 15 21 25 28
3	6	8 13 18 19 26 28
4	8	7 10 12 14 15 20 22 30
5	12	2 3 6 7 9 12 13 14 20 21 26 30
6	8	2 17 20 22 25 27 28 30
7	11	5 7 8 10 13 18 19 20 21 25 29
8	9	1 8 10 14 15 19 24 28 29
9	6	2 4 5 13 22 24
10	14	1 4 5 7 10 14 15 18 23 24 26 27 28 29
11	12	2 6 7 8 9 17 20 22 23 25 28 29
12	10	1 6 11 12 15 18 21 22 24 27
13	12	2 4 6 7 11 13 18 21 22 25 26 28
14	12	2 5 6 7 8 9 18 23 25 26 27 28
15	13	1 3 5 8 9 12 13 15 17 20 24 26 27
16	8	4 5 6 7 15 16 23 27
17	12	4 6 7 8 9 10 12 13 16 23 26 28
18	11	1 3 6 8 9 10 17 21 23 24 30
19	11	1 2 3 7 8 10 11 13 16 24 25
20	10	1 2 4 8 15 19 25 27 28 29
21	8	3 6 7 11 15 22 24 27
22	7	5 10 11 14 17 24 26
23	9	3 6 7 9 18 20 21 23 24

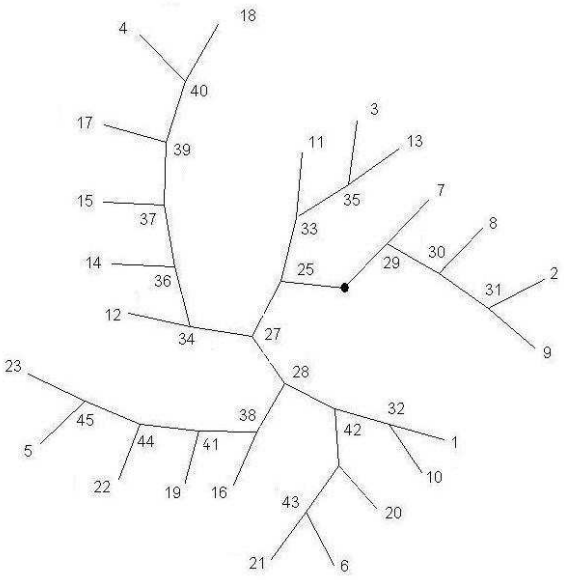


Figura 5.13 Árvore filogenética sem raiz com m=23 táxons e n=30 características, sem identificação do conteúdo das arestas

4. Instância: m=35, n=55 $\longmapsto$ Figuras 5.14 e 5.15

5. Instância: m=12 a partir de uma matriz de distâncias $\longmapsto$ Figuras 5.16 e 5.17

Optamos por desenvolver um algoritmo próprio (Algoritmo 2) [VGM$^+$06]) para desenhar as árvores filogenéticas resultantes da solução do algoritmo de reconstrução de filogenias.

Existem, entretanto, diversas ferramentas desenvolvidas para desenhar árvores filogenéticas como Spectrum [Cha98], PhyloDraw [CJC00], DrawTree [Fel93], TreeView [Pag01], entre outras; todas, porém, com a necessidade de instalação na máquina do usuário onde requerem recursos computacionais específicos.

Também existe o servidor Web PHYml [GGLD05] que utiliza o método de máxima verossimilhança [BGP05, KNT$^+$05, SLMW02, SHB$^+$01] para a inferência de árvores filogenéticas.

35	55	
Pyrobaculum	1	1
Aeropyrum	1	2
Sulfolobus	2	3 53
Rhodobacter	3	3 54 6
Caulobacter	3	3 54 7
Borrelia	3	3 54 8
Treponema	3	3 54 9
Thermoplasma	4	3 5 55 4
Chlorobium	5	3 5 55 10 11
Chamydophila	5	3 5 55 10 12
Pyrococus	3	3 5 13
Methanopyrum	4	3 5 14 15
Methanote	4	3 5 14 16
Methanocaldococ	4	3 5 14 17
Halobacterium	5	3 5 14 18 21
Archaeoglobus	6	3 5 14 18 22 23
Methanosarona	6	3 5 14 18 22 24
Bifidobacterium	6	3 5 14 19 25 26
Corynebacterium	6	3 5 14 19 25 27
Leptospira	6	3 5 14 19 20 28
Buchnera	8	3 5 14 19 20 28 29 30
Wigglesworthia	8	3 5 14 19 20 28 29 31
Pseudomonas	8	3 5 14 19 20 28 32 33
Salmonella	8	3 5 14 19 20 28 32 34
Vibrio	8	3 5 14 19 20 28 32 35
Yersinia	10	3 5 14 19 20 28 32 35 37 38
Eshcerichia	10	3 5 14 19 20 28 32 35 51 39
Shewanella	10	3 5 14 19 20 28 32 35 51 52
Haemophuilus	9	3 5 14 19 20 28 32 41 42
Pasteurella	9	3 5 14 19 20 28 32 41 43
Neisseria	8	3 5 14 19 20 28 44 46
Ralstonia	8	3 5 14 19 20 28 44 47
Xanthomanas	8	3 5 14 19 20 28 45 48
Xylella	8	3 5 14 19 20 28 45 49
Deinococcus	6	3 5 14 19 20 50

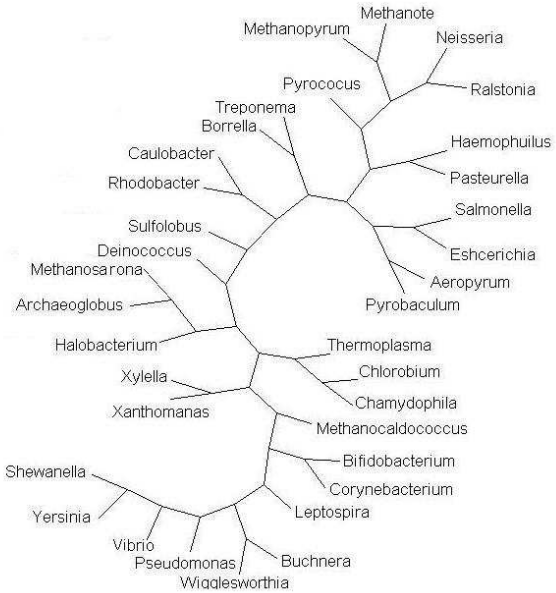


Figura 5.14 Filogenia do *Methionyl-tRNA synthetases*, adaptação parcial com 35 táxons – fonte [SWB$^+$99]

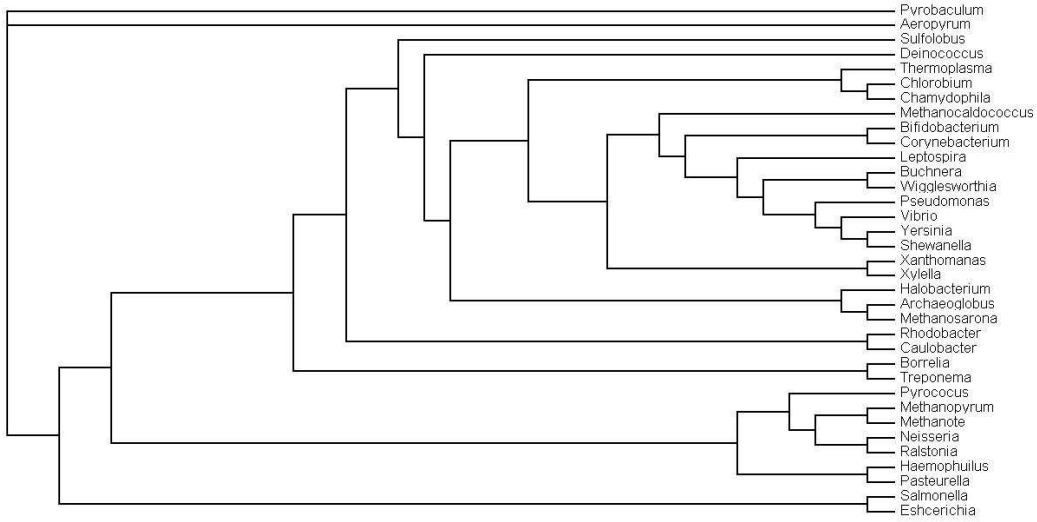


Figura 5.15 Filograma representativo da Filogenia da Figura 5.14, adaptado de [SWB$^+$99]

```
Teste Algoritmo A.10
Pan          0.00000 0.06270 0.29727 1.05443 0.96851 1.33148 1.57404 1.81665 \
             1.78387 2.04425 2.23002 2.92153
Homo         0.00000 0.30207 1.09605 0.99060 1.30331 1.51684 1.86601 1.83256 \
             2.05041 2.20085 2.93766
Rattus       0.00000 1.15873 1.06629 1.49656 1.58147 1.89184 1.88458 2.16722 \
             2.45136 3.21539
Cobitis      0.00000 0.83137 1.93208 2.24523 2.34713 2.30369 1.87563 2.29752 \
             3.15780
Oreochromi   0.00000 2.12354 2.04545 2.97511 2.90259 2.61552 2.71169 4.53111
Drosophila   0.00000 0.91440 2.08824 2.10987 2.18498 1.94142 2.40033
Melipona     0.00000 2.39180 2.36061 2.08709 2.57170 2.42471
Orthopsilo   0.00000 0.03635 1.03402 1.19520 3.11362
Candida      0.00000 1.00052 1.18476 3.07453
Yarrowia     0.00000 1.37538 2.82985
Marchantia   0.00000 3.05968
Caenorhabd   0.00000
```

```
((((Yarrowia:0:57;(Orthopsilosis:0:03;Candida:0:01):0:43):
0:20;Marchantia:0:52):0:35;Caenorhabditis:1:71):0:23;
(Drosophila:0:30;Melipona:0:61):0:54;((Rattus:0:13;(Pan:
0:03;Homo:0:03):0:10):0:15;(Cobitis:0:46;Oreochromis:
0:27):0:33):0:30);
```

Figura 5.16 a) Matriz de distâncias para a Filogenia da Figura 5.17, instância gerada pela ferramenta PhyloTree [VBB+06] para o gene "ATP6 protein" (selecionando estes doze táxons). b) *String* correspondente gerada pelo Algoritmo 9

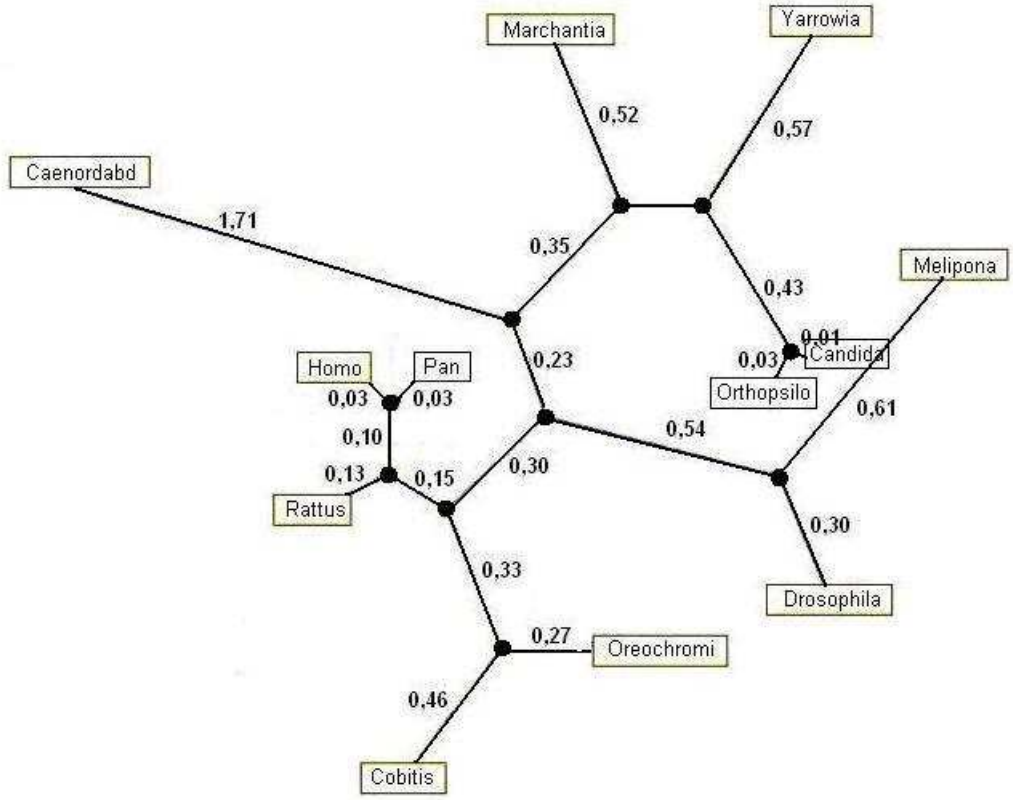


Figura 5.17 Árvore filogenética com doze táxons gerada pelo Algoritmo 2 a partir da matriz de distâncias apresentada na Figura 5.16. Os identificadores das espécies foram representados pelos dez primeiros caracteres.

5.7.3 Estudo de caso em que a Filogenia não é Perfeita

Os experimentos a seguir têm por objetivo apresentar um caso em que a filogenia não é perfeita. A Figura 5.18 contém uma instância com oito táxons e doze características; o resultado que aparece na figura é a saída do Algoritmo 3 que indica que a filogenia não é perfeita para a matriz dada; isto porque não é satisfeita o critério do Lema-1, ou seja: $\exists k \neq j \in \{1,\ldots,n\}$ tal que $(c_k \cap c_j \neq \varnothing)$ e $(c_k \not\subseteq c_j)$ e $(c_k \not\supseteq c_j)$; neste caso diz-se que existe uma "incongruência" entre as colunas k e j.

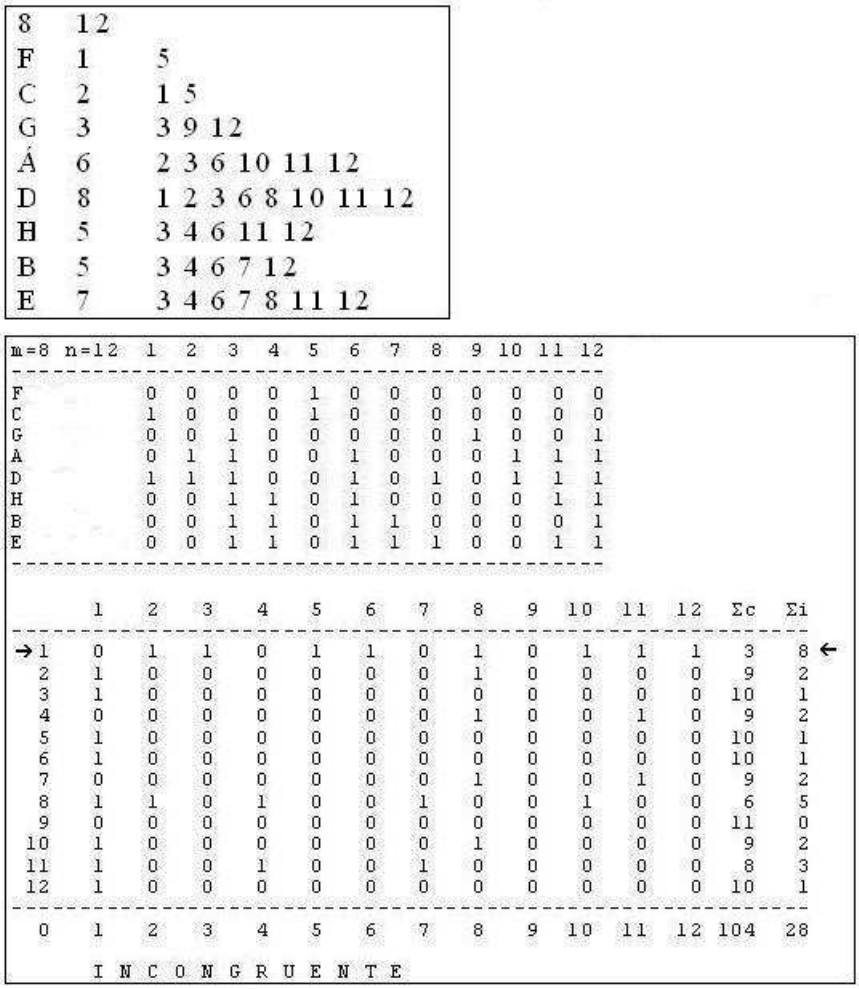
As duas últimas colunas da matriz apresentada na Figura 5.18 mostra $\sum c$ (número de congruências) e $\sum i$ (número de incongruências); com base nessas informações, e usando um critério "guloso", eliminamos a característica "1" que é aquela com maior valor de $\sum i$. Executando novamente o Algoritmo 3 para a instância dada sem a característica "1" obtivemos os resultados apresentados na Figura 5.19. Repetindo o processo, eliminamos seguidamente a característica "8", com resultados na Figura 5.20 e após eliminar a característica "11" foi possível transformar a instância dada numa Filogenia Perfeita com resultados mostrados na Figura 5.21. Isso no entanto, não pode ser generalizado, pois conforme citado anteriormente trata-se de um método guloso, em geral não eficaz.

Este critério é conhecido como "critério da compatibilidade" o qual Day & Sankoff [DS86] mostram que não são conhecidos algoritmos polinomiais para resolvê-lo de forma exata; no caso, este é um problema de otimização que consiste em encontrar o número máximo de características que admitem uma filogenia perfeita, ou seja, excluindo o menor número de características possíveis.

A Figura 5.22 mostra a forma final da árvore filogenética considerando primeiramente somente as características da instância reduzida e incluindo aquelas que foram retiradas de acordo com os critérios de convergência e reversão, discutido na Seção 2.7.

Uma das técnicas para inferir filogenias, tratada no capítulo seguinte, foi utilizada para este exemplo obtendo como resultado a árvore da Figura 5.22.

Para que seja possível utilizar os algoritmos descritos neste capítulo para obter a forma da árvore filogenética inferida com indicação nas arestas das características, sugerimos que seja



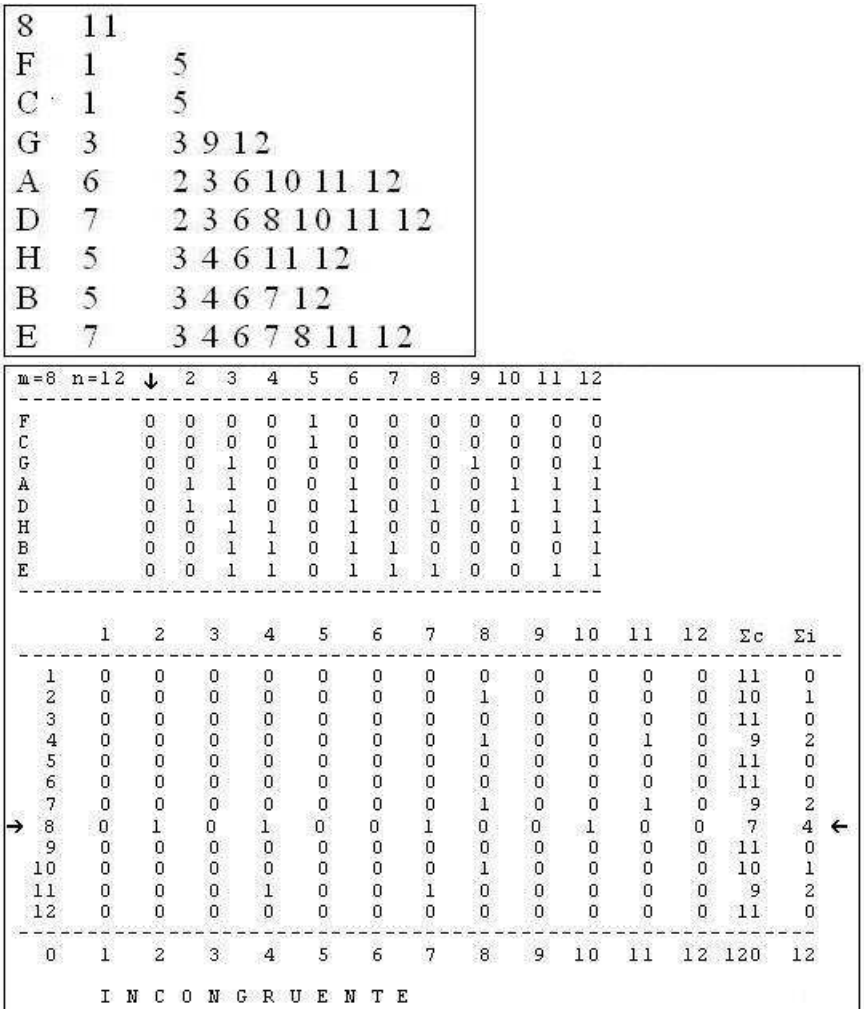
```
8   12
F   1     5
C   2     1 5
G   3     3 9 12
Á   6     2 3 6 10 11 12
D   8     1 2 3 6 8 10 11 12
H   5     3 4 6 11 12
B   5     3 4 6 7 12
E   7     3 4 6 7 8 11 12
```

```
m=8 n=12   1  2  3  4  5  6  7  8  9 10 11 12
-----------------------------------------------
F          0  0  0  0  1  0  0  0  0  0  0  0
C          1  0  0  0  1  0  0  0  0  0  0  0
G          0  0  1  0  0  0  0  0  1  0  0  1
A          0  1  1  0  0  1  0  0  0  1  1  1
D          1  1  1  0  0  1  0  1  0  1  1  1
H          0  0  1  1  0  1  0  0  0  0  1  1
B          0  0  1  1  0  1  1  0  0  0  0  1
E          0  0  1  1  0  1  1  1  0  0  1  1
-----------------------------------------------

      1   2   3   4   5   6   7   8   9  10  11  12  Σc  Σi
-------------------------------------------------------------
→1    0   1   1   0   1   1   0   1   0   1   1   1   3   8 ←
  2   1   0   0   0   0   0   0   1   0   0   0   0   9   2
  3   1   0   0   0   0   0   0   0   0   0   0   0  10   1
  4   0   0   0   0   0   0   0   1   0   0   1   0   9   2
  5   1   0   0   0   0   0   0   0   0   0   0   0  10   1
  6   1   0   0   0   0   0   0   0   0   0   0   0  10   1
  7   0   0   0   0   0   0   0   1   0   0   1   0   9   2
  8   1   1   0   1   0   0   1   0   0   1   0   0   6   5
  9   0   0   0   0   0   0   0   0   0   0   0   0  11   0
 10   1   0   0   0   0   0   0   1   0   0   0   0   9   2
 11   1   0   0   1   0   0   1   0   0   0   0   0   8   3
 12   1   0   0   0   0   0   0   0   0   0   0   0  10   1
-------------------------------------------------------------
  0   1   2   3   4   5   6   7   8   9  10  11  12 104  28

      I N C O N G R U E N T E
```

Figura 5.18 Instância para o caso de uma Filogenia não Perfeita com oito táxons e doze características; as duas últimas colunas da matriz de saída ($\sum c$ e $\sum i$) indicam, respectivamente para cada característica, o número de congruências e de incongruências com as demais características



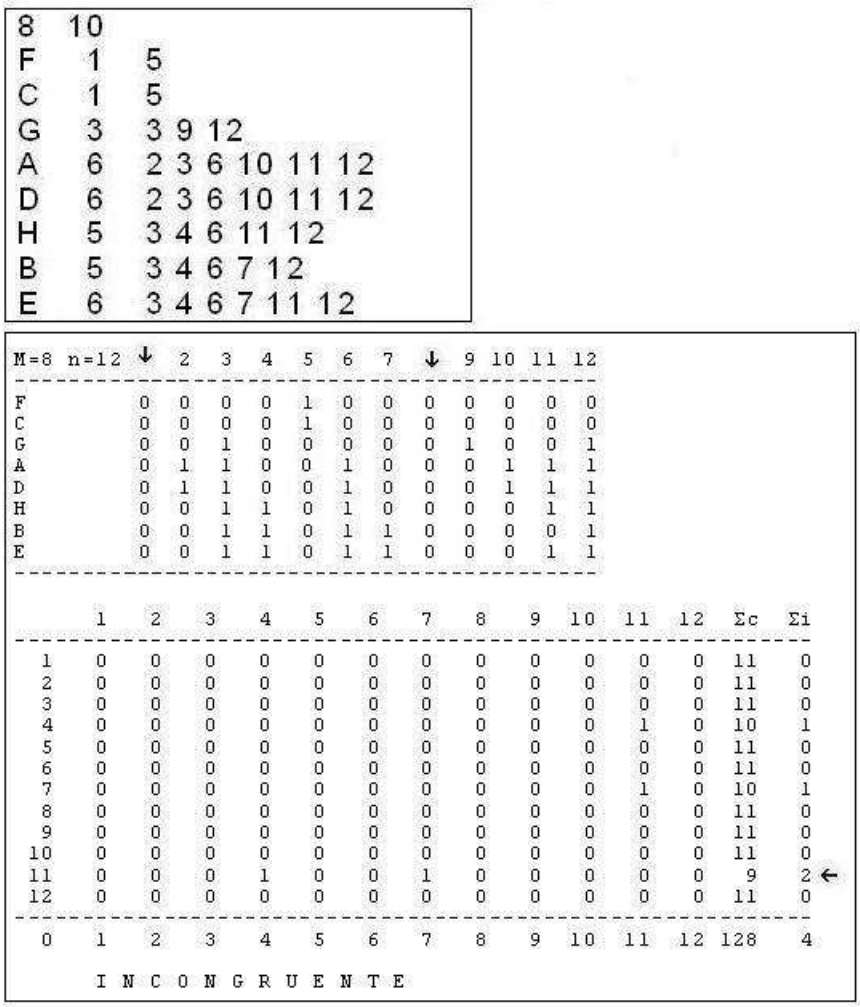
```
8   11
F   1     5
C   1     5
G   3     3 9 12
A   6     2 3 6 10 11 12
D   7     2 3 6 8 10 11 12
H   5     3 4 6 11 12
B   5     3 4 6 7 12
E   7     3 4 6 7 8 11 12
```

m=8 n=12 ↓	2	3	4	5	6	7	8	9	10	11	12
F	0	0	0	0	1	0	0	0	0	0	0	0
C	0	0	0	0	1	0	0	0	0	0	0	0
G	0	0	1	0	0	0	0	0	1	0	0	1
A	0	1	1	0	0	1	0	0	0	1	1	1
D	0	1	1	0	0	1	0	1	0	1	1	1
H	0	0	1	1	0	1	0	0	0	0	1	1
B	0	0	1	1	0	1	1	0	0	0	0	1
E	0	0	1	1	0	1	1	1	0	0	1	1

	1	2	3	4	5	6	7	8	9	10	11	12	Σc	Σi	
1	0	0	0	0	0	0	0	0	0	0	0	0	11	0	
2	0	0	0	0	0	0	0	1	0	0	0	0	10	1	
3	0	0	0	0	0	0	0	0	0	0	0	0	11	0	
4	0	0	0	0	0	0	0	1	0	0	1	0	9	2	
5	0	0	0	0	0	0	0	0	0	0	0	0	11	0	
6	0	0	0	0	0	0	0	0	0	0	0	0	11	0	
7	0	0	0	0	0	0	0	1	0	0	1	0	9	2	
→ 8	0	1	0	1	0	0	1	0	0	1	0	0	7	4	←
9	0	0	0	0	0	0	0	0	0	0	0	0	11	0	
10	0	0	0	0	0	0	0	1	0	0	0	0	10	1	
11	0	0	0	1	0	0	1	0	0	0	0	0	9	2	
12	0	0	0	0	0	0	0	0	0	0	0	0	11	0	
0	1	2	3	4	5	6	7	8	9	10	11	12	120	12	

I N C O N G R U E N T E

Figura 5.19 Instância para o caso de uma Filogenia não Perfeita obtida a partir da Figura 5.18 excluindo a característica “1” (oito táxons e onze características)



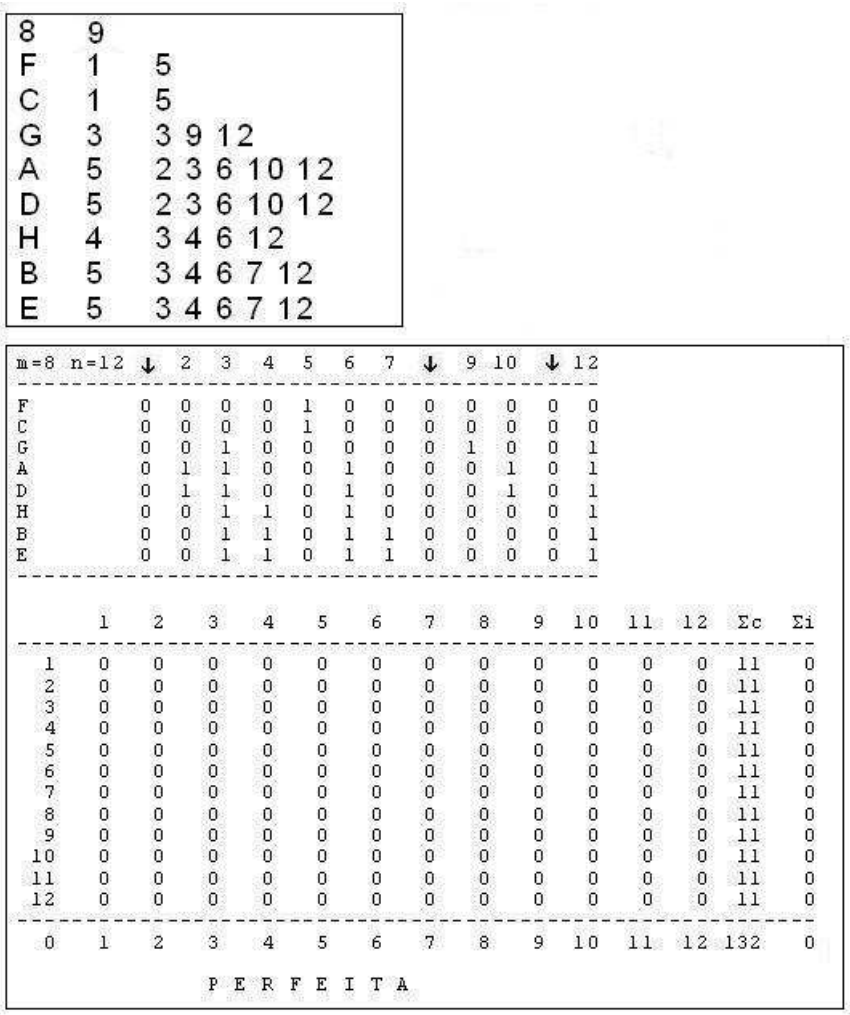
```
8  10
F   1   5
C   1   5
G   3   3 9 12
A   6   2 3 6 10 11 12
D   6   2 3 6 10 11 12
H   5   3 4 6 11 12
B   5   3 4 6 7 12
E   6   3 4 6 7 11 12
```

M=8 n=12	↓ 2	3	4	5	6	7	↓ 9	10	11	12
F	0	0	0	1	0	0	0	0	0	0
C	0	0	0	1	0	0	0	0	0	0
G	0	0	1	0	0	0	0	1	0	1
A	0	1	1	0	0	1	0	0	1	1
D	0	1	1	0	0	1	0	0	1	1
H	0	0	1	1	0	1	0	0	0	1
B	0	0	1	1	0	1	1	0	0	1
E	0	0	1	1	0	1	1	0	0	1

	1	2	3	4	5	6	7	8	9	10	11	12	Σc	Σi	
1	0	0	0	0	0	0	0	0	0	0	0	0	11	0	
2	0	0	0	0	0	0	0	0	0	0	0	0	11	0	
3	0	0	0	0	0	0	0	0	0	0	0	0	11	0	
4	0	0	0	0	0	0	0	0	0	0	1	0	10	1	
5	0	0	0	0	0	0	0	0	0	0	0	0	11	0	
6	0	0	0	0	0	0	0	0	0	0	0	0	11	0	
7	0	0	0	0	0	0	0	0	0	0	1	0	10	1	
8	0	0	0	0	0	0	0	0	0	0	0	0	11	0	
9	0	0	0	0	0	0	0	0	0	0	0	0	11	0	
10	0	0	0	0	0	0	0	0	0	0	0	0	11	0	
11	0	0	0	1	0	0	1	0	0	0	0	0	9	2	←
12	0	0	0	0	0	0	0	0	0	0	0	0	11	0	
0	1	2	3	4	5	6	7	8	9	10	11	12	128	4	

I N C O N G R U E N T E

Figura 5.20 Instância para o caso de uma Filogenia não Perfeita obtida a partir da Figura 5.19 excluindo a característica “8” (oito táxons e dez características)



```
8   9
F   1   5
C   1   5
G   3   3 9 12
A   5   2 3 6 10 12
D   5   2 3 6 10 12
H   4   3 4 6 12
B   5   3 4 6 7 12
E   5   3 4 6 7 12
```

m=8 n=12	↓ 2	3	4	5	6	7	↓ 9	10	↓ 12
F	0	0	0	1	0	0	0	0	0	0
C	0	0	0	1	0	0	0	0	0	0
G	0	0	1	0	0	0	0	1	0	1
A	0	1	1	0	0	1	0	0	1	0	1
D	0	1	1	0	0	1	0	0	1	0	1
H	0	0	1	1	0	1	0	0	0	0	1
B	0	0	1	1	0	1	1	0	0	0	1
E	0	0	1	1	0	1	1	0	0	0	1

	1	2	3	4	5	6	7	8	9	10	11	12	Σc	Σi
1	0	0	0	0	0	0	0	0	0	0	0	0	11	0
2	0	0	0	0	0	0	0	0	0	0	0	0	11	0
3	0	0	0	0	0	0	0	0	0	0	0	0	11	0
4	0	0	0	0	0	0	0	0	0	0	0	0	11	0
5	0	0	0	0	0	0	0	0	0	0	0	0	11	0
6	0	0	0	0	0	0	0	0	0	0	0	0	11	0
7	0	0	0	0	0	0	0	0	0	0	0	0	11	0
8	0	0	0	0	0	0	0	0	0	0	0	0	11	0
9	0	0	0	0	0	0	0	0	0	0	0	0	11	0
10	0	0	0	0	0	0	0	0	0	0	0	0	11	0
11	0	0	0	0	0	0	0	0	0	0	0	0	11	0
12	0	0	0	0	0	0	0	0	0	0	0	0	11	0
0	1	2	3	4	5	6	7	8	9	10	11	12	132	0

P E R F E I T A

Figura 5.21 Instância obtida a partir da Figura 5.20 excluindo a característica “11” (oito táxons e nove características) - transformada em Filogenia Perfeita

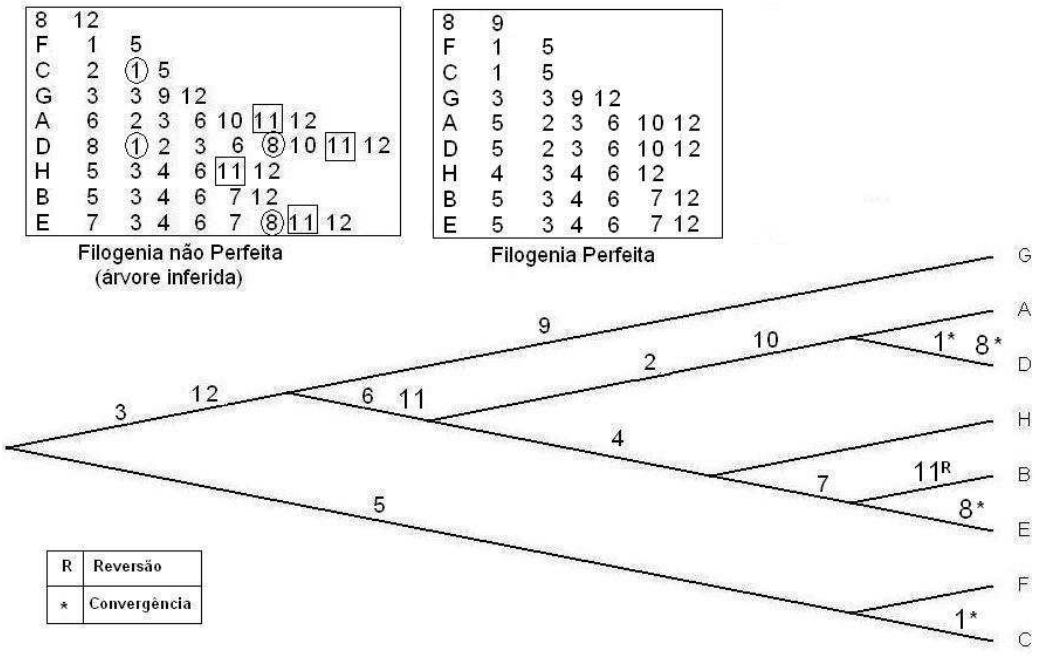


Figura 5.22 Árvore final inferida com duas convergências e uma reversão; exemplo adapatado de [Amo97]

feito o seguinte procedimento. Para cada característica "repetida" por causa das transformações realizadas pode ser atribuído uma identificação de uma nova característica.

Por exemplo, no caso da árvore da Figura 5.22 a "característica" 11^R seria denominada como 13; um dos 1^* seria a característica 14 (o outro continuaria sendo 1) e finalmente um dos 8^* seria 15 (o outro 8). Assim, a forma desta instância convertida seria a apresentada na Figura 5.23 com sua árvore correspondente na Figura 5.24.

```
8 15
F  1  5
C  2  1  5
G  3  3  9 12
A  6  2  3  6 10 11 12
D  8 14  2  3  6 15 10 11 12
H  5  3  4  6 11 12
B  7  3  4  6  7 11 12 13
E  7  3  4  6  8 11 12
```

```
 m=8 n=15    1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
-------------------------------------------------------
 A           0  1  1  0  0  1  0  0  0  1  1  1  0  0  0
 B           0  0  1  1  0  1  1  0  0  0  1  1  1  0  0
 C           1  0  0  0  1  0  0  0  0  0  0  0  0  0  0
 D           0  1  1  0  0  1  0  0  0  1  1  1  0  1  1
 E           0  0  1  1  0  1  1  0  0  0  1  1  1  0  0
 F           0  0  0  0  1  0  0  0  0  0  0  0  0  0  0
 G           0  0  1  0  0  0  0  0  1  0  0  1  0  0  0
 H           0  0  1  1  0  1  0  0  0  0  1  1  0  0  0
-------------------------------------------------------

     1  2  3  4  5  6  7  8  9 10 11 12 13 14 15  Σc  Σi
-------------------------------------------------------
 1   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  14  0
 2   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  14  0
 3   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  14  0
 4   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  14  0
 5   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  14  0
 6   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  14  0
 7   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  14  0
 8   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  14  0
 9   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  14  0
10   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  14  0
11   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  14  0
12   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  14  0
13   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  14  0
14   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  14  0
15   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  14  0
-------------------------------------------------------
 0   1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 210  0

          P E R F E I T A
```

Figura 5.23 Instância convertida após inferência. As características 13, 14 e 15 correspondem respectivamente aos rótulos 11^R, 1^* e 8^*

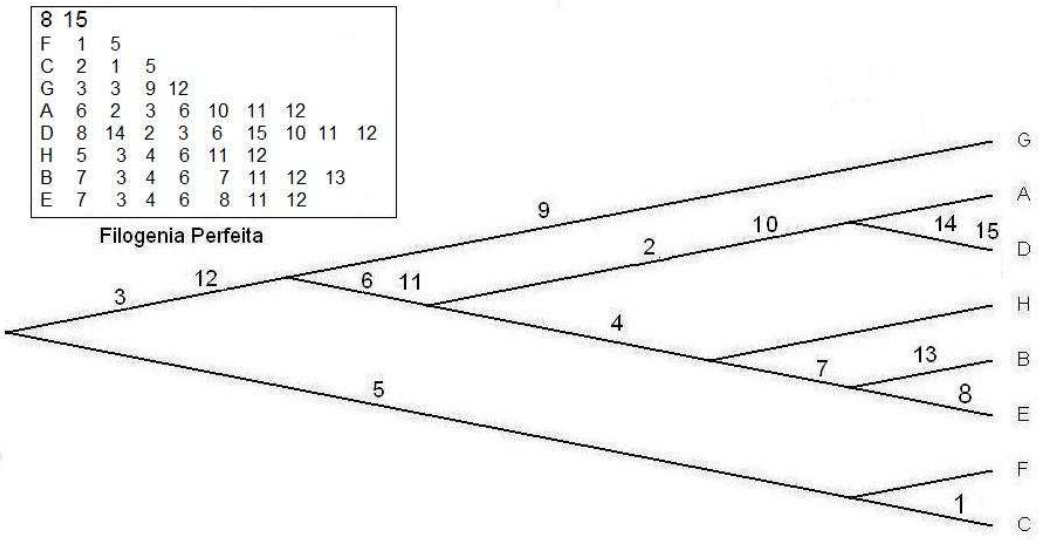


Figura 5.24 Filogenia corresponde à instância da Figura 5.23. Observa-se que é a mesma árvore da Figura 5.22 com a respectiva substutição dos rótulos 11^R, 1^* e 8^*

Daqui em diante, as filogenias serão representadas sem a indicação das características nas arestas pois, conforme foi visto, uma vez que um método determina qual a melhor topologia da árvore para uma dada instância, a atribuição destas em seus ramos pode ser feita de forma simples.

CAPÍTULO 6

Heurística para Inferir Filogenias

6.1 Considerações Iniciais

Como sabemos, um método filogenético consiste basicamente na separação das espécies em grupos correspondentes a subárvores que compartilham propriedades comuns. A formação destes grupos é possível em função das transformações que ocorrem ao longo do processo evolutivo [Ste00].

Um dos critérios adotados, e com bastante aceitação, é o da *parcimônia* [KFHW98, PHY06, SM87] que consiste em inferir [Fel93, SOWH96] que a melhor árvore filogenética para um grupo de táxons é aquela que requer o menor número de transformações evolutivas, correspondendo na natureza ao "caminho" de menor esforço. Portanto, podemos dizer, sob essa análise, que o Problema da Filogenia [AR02, RV05] é um problema de otimização combinatória que consiste em obter, com um número mínimo de passos evolutivos [BFW93, DJS86], uma Árvore Filogenética representativa desse grupo. Utilizando esse critério como função objetivo o problema é classificado como NP-Difícil [FG82, GJ79], conforme análise feita na Seção 3.4.

Para instâncias pequenas são conhecidos alguns algoritmos exatos e, em geral, existem outros aproximados para resolver esse problema [AFB93, AR02, DK87, Gol96, Sta05, SHB+01, WJL96]. O algoritmo apresentado neste capítulo tem como primeira etapa gerar configurações iniciais, algumas delas construídas com base nos dados e outras de forma aleatória, que servirão de base para os passos seguintes que correspondem a recombinações entre estas configurações e criação das vizinhanças diversificadas utilizadas na fase de busca local. Uma rotina final de intensificação com melhoria iterativa é utilizada para tentar obter uma solução ótima do problema [VGFM07].

Uma peculiaridade deste problema, ao adotar a estrutura de dados definida na Seção 6.3, é

que pequenas alterações numa dada configuração podem corresponder a uma grande alteração no valor da função objetivo. A Figura 6.3 da página 118 mostra um exemplo desta afirmação, bem como a forma de representação da estrutura utilizada para armazenamento de árvores filogenéticas.

6.2 Problema da Filogenia sob o Critério da Parcimônia

O problema abordado consiste em encontrar uma filogenia para um conjunto de espécies (chamadas táxons) baseada em um conjunto de características, assim definido:

- Dados:
  - $\mathscr{T}$: conjunto com m táxons a serem analisados, chamados táxons operacionais
  - $\mathscr{C}$: conjunto com n características, cada uma delas com até r estados
  - $\mathscr{R}$: conjunto dos estados: $\{0,1,2,\ldots,(r-1)\}$
  - $\mathscr{M}$: matriz de características $(\mathscr{T} \times \mathscr{C})$; $m_{ij} \in \mathscr{R}$, $\forall i \in \mathscr{T}$,$\forall j \in \mathscr{C}$
- Obter:
  - $\mathscr{A}$: árvore filogenética associada à matriz $\mathscr{M}$ contendo n folhas e todos os nós internos, correspondentes aos táxons hipotéticos, de grau 3

Estudos biológicos justificam [Amo97, Mou01] restringirmos nossa abordagem às árvores binárias, como descrito acima, onde cada característica poderá ter dois ou três estados diferentes. Quando $r = 2$ diremos que a matriz é binária: $m_{ij} = 1$ indica a presença da característica j no táxon i, e $m_{ij} = 0$ caso contrário. Quando $r = 3$, além dos estados 0/1 é considerado um novo estado denotado por "?" que representa uma indeterminação, ou seja, não se tem conhecimento se "j" está presente em "i". A Tabela 6.1 mostra um exemplo de uma matriz de características $\mathscr{M}$.

Para definir que função pretendemos otimizar é necessário introduzir o conceito de parcimônia [KFHW98]. Dada uma árvore viável $\mathscr{A}$ como definido anteriormente, denotamos, para cada vértice da árvore, rótulos correspondentes às características. As folhas das árvores

Tabela 6.1 Matriz $\mathscr{M}$ com $m = 4$ táxons, $n = 6$ características e $r = 3$ estados ($m_{ij} = 1$, quando a característica j está presente no táxon i; $m_{ij} = 0$, quando está ausente e $m_{ij} =$ "?", quando não se tem conhecimento desta presença/ausência, caracterizando uma indeterminação). Como existem três valores iguais a "?" num total de 24 elementos então o percentual de indeterminação desta instância é $12,5\%$

Táxons	Características					
i	c_1	c_2	c_3	c_4	c_5	c_6
1	0	?	0	?	0	?
2	0	1	1	0	1	1
3	0	0	1	1	1	0
4	1	0	0	1	1	0

terão rótulos (táxons) dados pela linha correspondente da matriz $\mathscr{M}$. Os rótulos dos nós internos serão dados pelo consenso entre os dois nós filhos: caso o valor da característica seja idêntico nos filhos, o pai herda este mesmo valor; se os dois filhos possuem características definidas e diferentes, o pai terá o valor indefinido representado por "?" naquela característica e se um dos filhos tem uma característica definida (0/1) e o outro, não ("?") na mesma posição, o pai herda o valor "definido" (0/1) naquela posição. Como exemplo, a Figura 6.1 mostra o cálculo da parcimônia de uma árvore $\mathscr{A}$ associada à matriz da Tabela 6.1.

O custo (parcimônia total) da árvore rotulada será dado pela soma do custo de cada nó (vértice) não terminal. O custo de um vértice é obtido somando para todas as características o número de posições em que o rótulo do pai difere do rótulo do filho (e ambos são diferentes de "?"). Este número pode ser interpretado como o total de ocorrências genéticas necessárias para que a árvore represente o que ocorreu na natureza. Alguns biólogos acreditam que este número deve ser minimizado, ou seja, a natureza é parcimoniosa, e busca o menor número possível de ocorrências de transformações evolutivas, como reversões e convergências [Amo97]. Vamos buscar, portanto, uma árvore filogenética que minimiza o critério da parcimônia, como descrito acima.

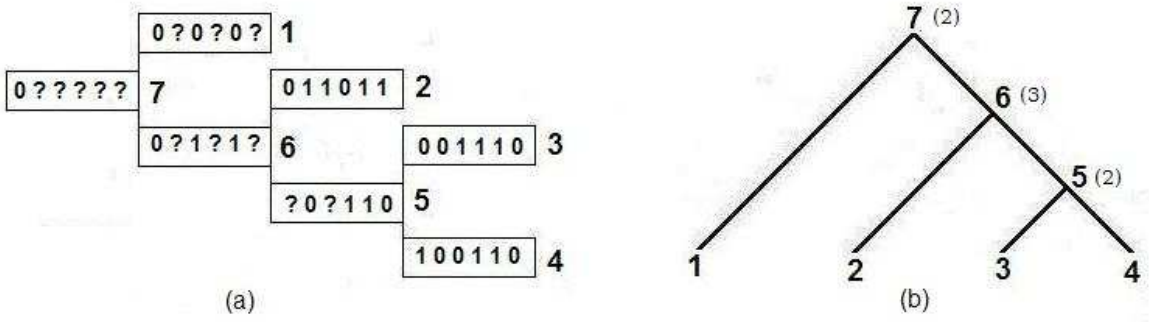


Figura 6.1 a) Exemplo do cálculo da parcimônia de uma das árvores associada à matriz de características da Tabela 6.1. b) relacionamentos considerados para cálculo da parcimônia. Os números entre parênteses indicam a parcimônia de cada nó. O "custo total" da parcimônia desta árvore é igual a 7 que corresponde ao valor da função objetivo (ver subseção 6.4)

6.3 Representação dos Dados

Optamos por representar uma árvore filogenética $\mathscr{A}$ através de vetores indexados pelos vértices. Um vetor $z \in N^{2m-2}$ será interpretado da seguinte forma. As primeiras m posições correspondem aos m táxons originais. Assim, z_i para $i = 1,\ldots,m$ deverá assumir valores entre $m+1$ e $2m-1$. As posições restantes corresponderão aos $m-2$ nós internos (a raiz não é representada). Uma representação válida deve satisfazer para estes índices, além de $m+1 \leq z_i \leq 2m-1$, $z_i > i$. Como a árvore deve ser binária, cada valor de um componente do vetor deverá ocorrer 2 vezes.

A Figura 6.2 mostra algumas topologias de árvores e vetores z correspondentes. Vimos como calcular o número possível de topologias de árvores filogenéticas, em função de m, através da Equação 4.11 da página 66.

A representação que estamos utilizando é certamente redundante, uma vez que diferentes vetores z viáveis podem representar a mesma topologia em função dos valores duplos no vetor z ou de algumas simetrias presentes nas árvores binárias. Por exemplo, $z =< 5,6,5,6,7,7 >$ é redundante a $z^{(d)}$ da Figura 6.2 e $z' =< 7,6,7,6,9,8,8,9 >$ é redundante a $z^{(e)}$ da mesma figura. A Equação 6.1 determina o número de arranjos correspondentes às soluções viáveis geradas a

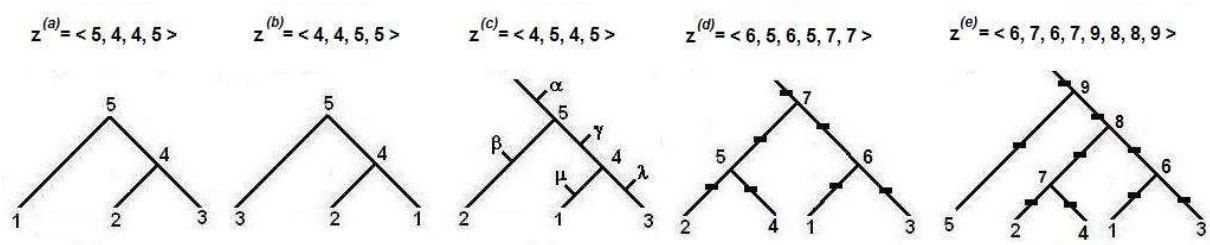


Figura 6.2 Os vetores $z^{(a)}$, $z^{(b)}$ e $z^{(c)}$ representam todas as configurações possíveis com m=3 táxons. Em $z^{(c)}$ estão indicadas as posições em que um quarto táxon poderia ser inserido. $z^{(d)}$ seria uma das 15 configurações possíveis para m=4; a posição β de $z^{(c)}$ foi a escolhida para inserção do táxon operacional "4". $z^{(e)}$ é uma das 105 configurações possíveis para m=5; a posição acima da raiz "7" de $z^{(d)}$ foi a escolhida para inserir o quinto táxon.

partir de uma dada configuração z. Ao comparar essa fórmula com representada pela equação 4.11 observa-se que $\chi'_m > \chi_m$.

$$\chi'_m = \frac{m!(m-1)!}{2^{m-1}} \tag{6.1}$$

O valor de χ'_m da Equação 6.1 pode ser obtido recursivamente através da Fórmula 6.2.

$$\chi'_2 = 1; \qquad \chi'_k = \chi'_{k-1}\frac{k(k-1)}{2}, \quad \forall k \in \{3,\ldots,m\} \tag{6.2}$$

6.3.1 Análise do número de arranjos viáveis

A Tabela 6.2 faz uma comparação entre o número de arranjos calculados pelas Equações 6.1 e 6.2 e o número de árvores filogenéticas dado pela Equação 4.11 da página 66.

Considerando:

1. Modelo da Estrutura de Dados

   $< z_1, z_2, z_3, z_4, \ldots, z_m, z_{m+1}, \ldots, z_{2m-3}, z_{2m-2} >$

2. Restrições

$m+1 \leq z_i \leq 2m-1$ e $z_i > i \quad \forall i \in I = \{1,\ldots,2m-2\}$

3. Propriedade

   Existem $(m-1)$ partições de I na forma $\{p,q\}$ tal que $z_p = z_q$

4. Número de Permutaçoes (com repetição) da estrutura (1) com a propriedade (3)

$$P_m = \frac{(2m-2)!}{2^{m-1}} \tag{6.3}$$

Tabela 6.2 Tabela comparativa entre o número de arranjos possíveis (P), viáveis (χ') e o número de árvores filogenéticas existentes (χ) para m táxons. Os valores 45 e 117 são, respectivamente, o menor e o maior valor de m nas instâncias dadas.

m	P_m	χ'_m	χ_m
2	1	1	1
3	6	3	3
4	90	18	15
5	2.520	180	105
6	113.400	2.700	945
7	7.484.400	56.700	10.395
10	1,25e013	2,57e009	3,45e007
20	9,98e038	5,64e029	8,20e021
45	1,05e121	1,80e097	3,96e066
117	5,00e414	1,62e348	1,47e224

Podemos citar como vantagens do uso desta representação a facilidade de modificar vetores ou combiná-los para obter novas configurações, além da forma rápida de verificar se uma dada configuração é viável ou não (ver Algoritmo 13). Observa-se ainda que as restrições do tipo $z_i > i$ eliminam vários arranjos de z reduzindo o espaço de busca contendo somente soluções viáveis. Mesmo em relação à redundância, podemos obter alguma vantagem durante o processo de diversificação da múltipla vizinhança.

Em nosso algoritmo é utilizado ainda uma estrutura adicional para armazenar, além de cada vértice interno i (rotulados de $m+1$ a $2m-1$), os rótulos de seus filhos direito e esquerdo, $i.right$ e $i.left$, respectivamente. A Tabela 6.3 mostra um exemplo dessa estrutura.

Tabela 6.3 Exemplo do uso de ponteiros em árvores binárias – parte do vetor $z^{(d)}$ da Figura 6.2. Observa-se que $z(i.left) = z(i.right) = i$

i	$z(i)$	$i.left$	$i.right$
5	7	2	4
6	7	1	3
7	—	5	6

6.4 Cálculo da Função Objetivo

Conforme comentado na Seção 6.2 as instâncias contêm algumas características indefinidas, representadas pelo caráter "?". Por ocasião da implementação do algoritmo de cálculo da função custo, este caráter foi substituído pelo dígito "2", ou seja, o conjunto de estados utilizado é $\mathscr{R}$= $\{0,1,2\}$.

Tabela 6.4 Resultado da parcimônia considerando o conjunto de estados $\mathscr{R}$= $\{0,1,2\}$. A função f representa o incremento do custo e t o resultado da operação entre dois estados

f/t	0	1	2
0	0/0	1/2	0/0
1	1/2	0/1	0/1
2	0/0	0/1	0/2

A utilização de $\mathscr{R}$ reduz o número de testes do Algoritmo 11, de modo que se a soma das duas características for igual a "1" então $f = 1$ e $t = 2$, caso contrário, $f = 0$ e t é igual ao menor valor entre os dois operandos.

Considerando que os vetores t_L e t_R contêm as "n" características dos táxons "filhos", respectivamente, à esquerda e à direita do táxon hipotético "pai" (t), então o conteúdo deste vetor $t = z_i$ e do resultado de sua parcimônia são dados pelo Algoritmo 11 de acordo com a definição:

$$z_i(j) = \begin{cases} 2 & se \quad t_L(j) + t_R(j) = 1 \quad \Rightarrow cost\_node(t(j)) = 1 \\ min\{t_L(j), t_R(j)\} & caso\ contrario \quad \Rightarrow cost\_node(t(j)) = 0 \end{cases} \tag{6.4}$$

Algoritmo 11: Calcula custo parcial (um nó interno) pelo critério da parcimônia

```
11.1  function cost_node(n, t_L, t_R, t)
      Input: n, t_L, t_R
        t_L, t_R = vetores (filho esquerdo/direito) com n características pertencentes ao conjunto R = {0,1,2}
      Output: cost_node, t
        cost_node – valor parcial da função objetivo (custo de um nó interno)
        t = vetor (pai) com n características, t(j) ∈ R, ∀j ∈ {1,...,n}
11.2  begin
11.3      cost_node ← 0
11.4      for j ← 1 to n do
11.5          if (t_L(j) + t_R(j))=1 then
11.6              t(j) ← 2
11.7              cost_node ← cost_node + 1
11.8          else
11.9              if t_L(j) < t_R(j) then
11.10                 t(j) ← t_L(j)
11.11             else
11.12                 t(j) ← t_R(j)
11.13             endif
11.14         endif
11.15     endfor
11.16     return(cost_node, t)
11.17 end
```

Ao considerar todos os nós da árvore filogenética representada pelo vetor z e utilizando o Algoritmo 11 obtemos a função objetivo calculada pelo Algoritmo 12 dada pela fórmula:

$$cost(z) = \sum_{i=m+1}^{2m-1} \sum_{j=1}^{n} cost\_node(n, i.left(j), i.right(j), i(j)) \tag{6.5}$$

Teorema 4. *O Algoritmo 12 tem complexidade de tempo* $O(nm)$, $\forall n \geq 3$.

Demonstração. Sejam as constantes c_1 e c_2 relativas, respectivamente, aos tempos de execução dos comandos externos e internos do laço **for**{j} do Algoritmo 11 ($cost\_node$), logo sua função de tempo é dada por $f_1(n) = c_1 + nc_2$.

Sejam as constantes c_3 e c_4 relativas, respectivamente, aos tempos de execução dos comandos simples externos e internos do laço **for**{i} do Algoritmo 12 ($cost$) que é executado $(nos - (m+1) + 1) = (2m - 2 - m - 1 + 1) = (m-2)$ *vezes*, logo sua função de tempo é dada por $f_2(m) = c_3 + (m-2)(c_4 + f_1(n)) = c_3 + (m-2)c_4 + (m-2)(c_1 + nc_2) = c_5 + mc_6 + mnc_2$, portanto a ordem de complexidade do Algoritmo 12 é $\mathcal{O}(mn)$.

□

Algoritmo 12: Calcula custo total de uma árvore representada por z pelo critério da parcimônia. Se o limite superior for atingido o processo é interrompido retornando $cost = \infty$. A configuração z tem a estrutura mostrada na Seção 6.3 e M é uma matriz de ordem $(2m-1) \times n$ com as linhas de 1 a m iguais a da matriz de entrada $\mathscr{M}$

```
12.1  function cost(m, n, z, nos, upperbound, M)
12.2  begin
12.3      cost ← 0
12.4      nos ← 2m − 2
12.5      for i ← m + 1 to nos + 1 do
12.6          seek(z, i.left, i.right) tal que z(i.left) = z(i.right) = i
12.7          t_L ← M(i.left, 1 : n);
12.8          t_R ← M(i.right, 1 : n)
12.9          cost ← cost + cost_node(n, t_L, t_R, t)
12.10         M(i, 1 : n) ← t(1 : n)
12.11         if cost ≥ upperbound then return(∞)
12.12     endfor
12.13     return(cost)
12.14 end
```

6.5 Descrição da Heurística

6.5.1 Introdução

A necessidade de partidas múltiplas e geração simultânea de diversas vizinhanças é justificada pelas características das configurações e estrutura de dados utilizada: pequenas alterações numa configuração podem acarretar em uma grande diferença para melhor ou pior no valor da função considerada. A Figura 6.3 mostra duas situações para a instância *xiii* da Tabela 6.6 da página 132.

Como diversas configurações distintas podem possuir o mesmo valor de parcimônia, foi utilizada uma tabela dinâmica $\mathscr{P}$ para armazenar todos os resultados inteiros encontrados dentro de uma faixa variável $\mathscr{F}$= [*upperbound*, *max-cost*], onde inicialmente esses parâmetros são respectivamente o menor e o maior valor da função objetivo na fase de geração das configurações iniciais. Observe que o limite inicial de $\mathscr{F}$ é constantemente alterado e a variável $\mathscr{P}$(*ind-cost*) contém o vetor de configuração z tal que $cost(z) = ind\text{-}cost$.

Teorema 5. *Para m táxons e n características tem-se:* $n \leq f(S) \leq \frac{mn}{2}$.

Demonstração. Parte I:

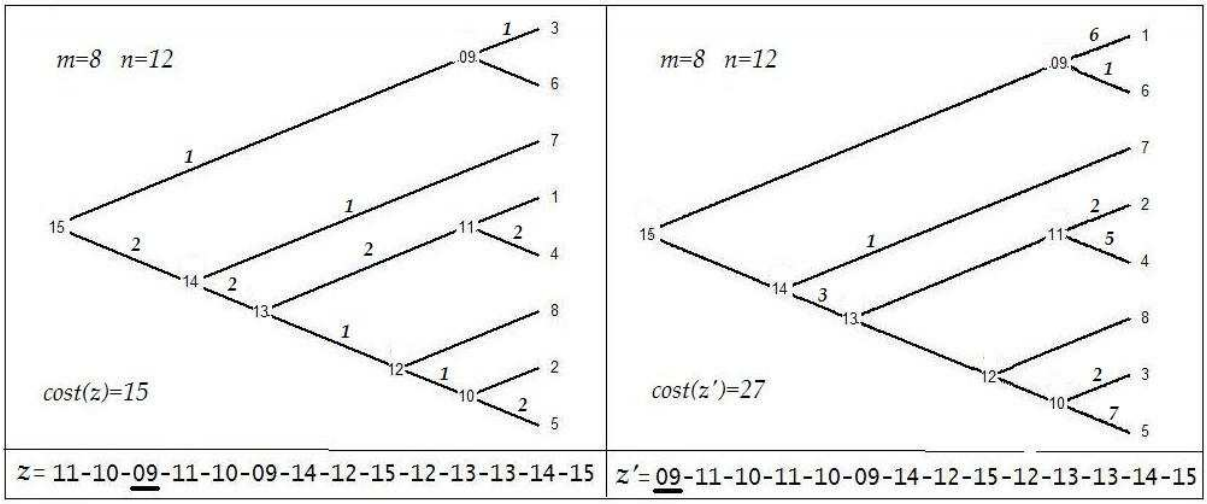


Figura 6.3 Duas configurações vizinhas, onde a segunda (z') é obtida a partir da primeira (z) por uma pequena modificação (deslocamento do elemento da posição 3 para 1 do vetor), com um significativo acréscimo (80%) do valor da função objetivo. As folhas de 1 a 8 correspondem aos táxons analisados ou táxons operacionais, os nós intermediários de 9 a 14 são os ancestrais "mais recentes" de cada subárvore ou táxons hipotéticos e a raiz 15 é o ancestral comum a todas espécies e grupos de espécies representados; os valores nas arestas representam o custo (número de transformações do tipo $0 \leftrightarrow 1$) entre os nós

- $f(S) \geq n$ $(f(S)$ é a função $cost)$

  $f(S) = n$ se $\mathscr{S}$ corresponde a uma filogenia perfeita. Setubal e Meidanis [SM97] definem e mostram que: "Uma matriz binária $\mathscr{M}$ admite uma Filogenia Perfeita se, e somente se, para cada par de características distintas (colunas $i \neq j$), $\forall\{i,j\} \subseteq \{1,\ldots,n\}$, os conjuntos $\beta_i = \mathscr{M}[1:m,i\,]$ e $\beta_j = \mathscr{M}[1:m,j\,]$ são disjuntos, $(\beta_i \cap \beta_j) = \varnothing$, ou um deles contém o outro, $(\beta_i \subseteq \beta_j)$ ou $(\beta_j \subseteq \beta_i)$"

  Sem perda de generalidade podemos considerar a Tabela 6.5 para simular os casos:

Tabela 6.5 Seleção de características para simular Filogenia Perfeita/Inferida

Táxons	i	j	k	l
A	0	1	1	0
B	1	0	1	1
C	0	0	0	1

  i) $\beta_i \cap \beta_j = \emptyset \Rightarrow cost\_node(2,A,B) = 2\ e\ t_{AB} = (2,2)$ {filogenia perfeita}

  $cost\_node(2,t_{AB},C) = 0 \Rightarrow f(S) = 2 = n$ e $t_{ABC} = (0,0)$.

  ii) $\beta_j \subseteq \beta_k \Rightarrow cost\_node(2,A,B) = 1\ e\ t_{AB} = (2,1)$ {filogenia perfeita}

  $cost\_node(2,t_{AB},C) = 1 \Rightarrow f(S) = n = 2$ e $t_{ABC} = (0,2)$.

  iii) $\beta_k \cap \beta_l \neq \emptyset,\ \beta_k \nsubseteq \beta_l\ e\ \beta_l \nsubseteq \beta_k$ {filogenia inferida} $cost\_node(2,A,B) = 1$

  $e\ t_{AB} = (1,2)\ e\ cost\_node(2,t_{AB},C) = 1\ e\ t_{ABC} = (2,1) \Rightarrow f(S) > n = 2$

  Parte II:

- $f(S) \leq \frac{mn}{2}$

  O pior caso é quando m é par e existem exatamente $\frac{m}{2}$ subárvores no penúltimo nível agrupando cada uma delas dois táxons com todas as n características disjuntas entre si, de modo que, denominando o conjunto de táxons $\mathscr{T}$=$\{t_1, t_2, t_3, \ldots t_{m-1}, t_m\}$ teríamos $cost\_node(n,t_{2i-1},t_{2i}) = n,\ ,\forall i \in \{1,\ldots,\frac{m}{2}\}\ e\ t = (2,2,\ldots,2) \Rightarrow f_{max}(S) = \frac{mn}{2}$. Se m é ímpar, pode-se incluir um táxon fictício t_{m+1} nulo e o táxon hipotético da subárvore com

filhos t_m *e* t_{m+1} irá possuir somente elementos iguais a "2" ou "0" não incrementando o custo final da função objetivo.

□

6.5.2 Deslocamentos de Blocos e Reconexões

A Figura 6.4 contém exemplos de alguns tipos de "movimentação de blocos" numa árvore filogenética. Os vetores representativos de cada subárvore são também mostrados para indicar qual a forma de deslocamento necessário para reconectá-los de modo que a nova configuração seja viável.

Procedimentos semelhantes podem ser encontrados em Rodrigues [Rod03] onde são apresentados todos os tipos possíveis de transferências, chamadas canônicas, de subárvores numa árvore binária.

A ocorrência desse movimentos são freqüentes durante o processamento do Algoritmo 16; alguns deles por recombinações entre duas configuraçoes apresentadas na Figura 6.5, outros na fase de melhoria iterativa, quando os táxons são trocados dois a dois sempre que possível (desconsiderando-se as repetições e verificando sua viabilidade). Por exemplo, o tipo da movimentação $1 \leftrightarrow 2$ é obtida na forma do descendente z_9 da Figura 6.5; $3 \leftrightarrow 4$ do z_{10}; $5 \leftrightarrow 6$, $7 \leftrightarrow 8$ e $9 \leftrightarrow 10$ do z_{11} e $11 \leftrightarrow 12$ do z_{12}. Outras trocas são processadas pela rotina final de melhoria iterativa quando repetições sucessivas atingirão outros deslocamentos e reconexões, alguns deles não apresentados nesta figura.

6.5.3 Recombinações Múltiplas

A Figura 6.5 mostra os 12 tipos de configurações geradas (descendentes) a partir de todas as combinações duas a duas das configurações iniciais (pais). Os pontos de corte indicados nos vetores z_i e z_j são gerados de forma aleatória em cada caso. Como se observa no Algoritmo 16 o melhor de todos estes descendentes é selecionado para a fase seguinte.

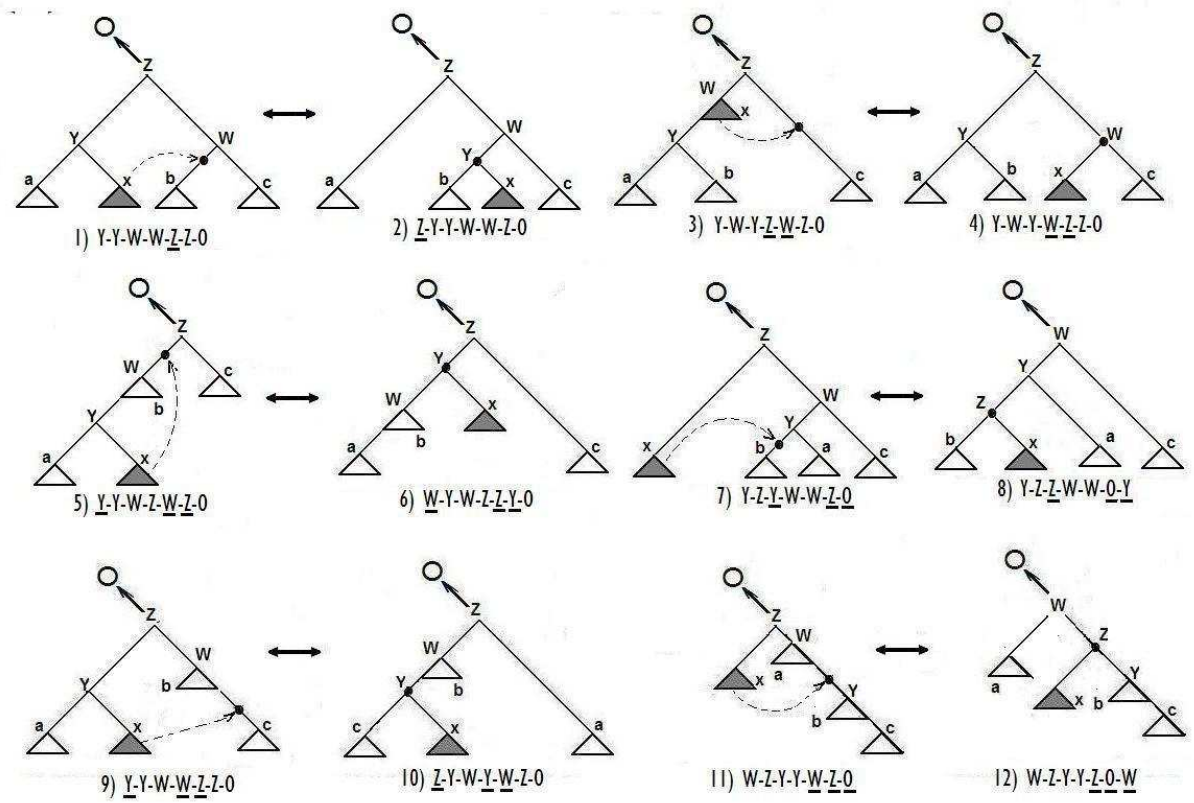


Figura 6.4 Exemplos de movimentação de blocos em árvores filogenéticas. Os rótulos a, x, b e c tanto podem ser táxons operacionais (folhas) como hipotéticos (nós internos). A notação com dupla seta ($\leftrightarrow$) indica que a conversão entre as configurações é reflexiva. Os rótulos abaixo de cada subárvore correspondem as suas representações utilizando a estrutura z (seção 6.3). Os elementos sublinhados são aqueles que devem ser "deslocados" para reproduzir as respectivas conversões

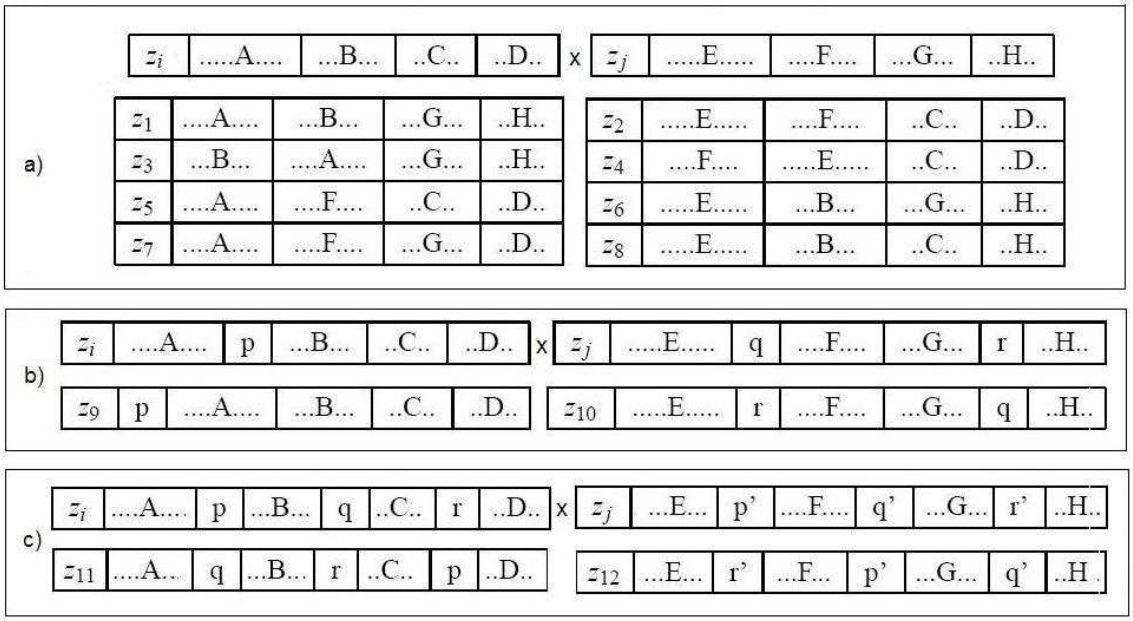


Figura 6.5 Recombinações realizadas nas configurações z_i e z_j para gerar: a) z_1 a z_8 por permuta de blocos; b) z_9 por deslocamento de blocos e c) z_{10}, z_{11} e z_{12} por deslocamento de elementos. As letras maiúsculas A–H representam os blocos (um ou mais elementos) enquanto que as minúsculas p, q e r, representam os elementos (uma única posição do vetor)

6.5.4 Verificando se uma dada configuração é viável

Ao recombinar configurações, ou deslocar e reconectar blocos, é necessário verificar se a configuração resultante desta transformação é viável. O Algoritmo 13 descrito a seguir faz esse teste.

Algoritmo 13: Verifica se uma configuração é viável. Se verdadeiro retorna 0; caso contrário retorna o valor da primeira posição onde Z não é viável

```
13.1  function Viavel(m, nos, Z)
      Input: m, nos, Z
        m – número de táxons
        nos – número de nós da árvore filogenética
        Z – vetor de configurações de tamanho = nos
      Output: Viavel
        Viavel = 0 ⇒ o vetor de configuração Z é viável
        Viavel ≠ 0 ⇒ Z não é viável
      Data: W
        W - vetor de trabalho
13.2  begin
13.3      for i ← nos − 1 downto m + 1 do
13.4          if Z(i) ≤ i then
13.5              return(i)
13.6          endif
13.7      endfor
13.8      W ← 0
13.9      for i ← 1 to nos do
13.10         W_Z(i) ← W_Z(i) + 1
13.11     endfor
13.12     for i ← m + 1 to nos + 1 do
13.13         if W_i ≠ 2 then
13.14             return(i)
13.15         endif
13.16     endfor
13.17     return(0)
13.18 end
```

6.6 Procedimento para Inferir uma Árvore Filogenética

O Algoritmo proposto, descrito a seguir na subseção 6.6.2, contém uma fase inicial de geração das configurações que utiliza uma função randômica com distribuição uniforme no intervalo

$[1, 2^{31} - 1]$ desenvolvida por Park & Miller [PM88]; toda a aleatorização necessária em qualquer rotina dos algoritmos utiliza esta função.

A Figura 6.6 mostra uma simulação do uso desta função de randomização para dez execuções na faixa de inteiros $[1,99]$. Observa-se uma uniformidade na distribuição dos números aleatórios nessa faixa. Para qualquer algoritmo que usa aleatorização essa regularidade é importante pois tende a minimizar a geração de configurações redundantes.

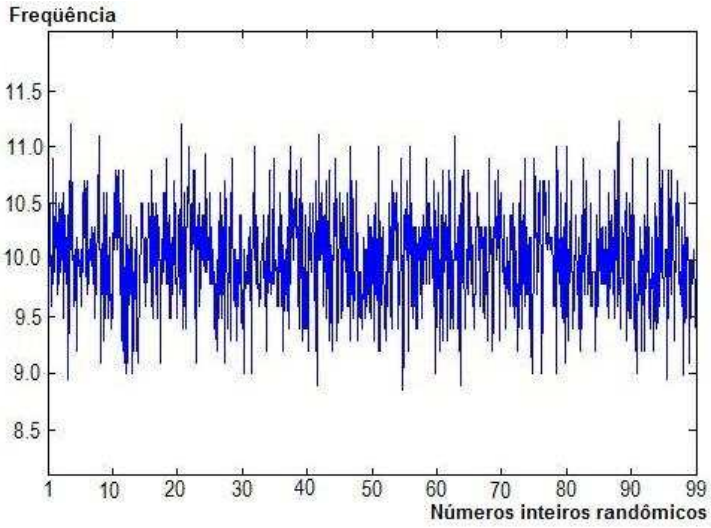


Figura 6.6 Resultados dos testes da função de randomização para dez execuções. Observe que a faixa $[1,99]$ tem para cada elemento uma freqüência média calculada entre $[9,11]$, concentrando este valor em torno de uma aproximação de 5% do valor ideal que seria 10 vezes

6.6.1 Gerando Dados Iniciais

A população inicial gerada, de tamanho "$npop$" (proporcional a m e indicado na Tabela 6.6), contém até 60% das configurações geradas com base nos dados, o restante é construído de forma aletória; uma análise prévia das características dos táxons é feita para classificá-los de modo que táxons "semelhantes" sejam nós terminais vizinhos na árvore correspondente.

Os Algoritmos 14 e 15 apresentados a seguir geram uma população inicial para o heurística proposta.

Algoritmo 14: Gera Configurações iniciais para um Problema de Filogenia a ser tratado pela Heurística de Busca Local proposta – corresponde à rotina Gera_Pop do módulo principal

Input: $m, n, \mathscr{M}, \mathscr{N}$
$\mathscr{M}$ – Matriz com m táxons e n características em $R = \{0,1,2\}$
$\mathscr{N}$ – tamanho da população

Output: $P, upperbound, maxcost$
P - matriz com $\mathscr{N}$ configurações $upperbound$ e $maxcost$ - menor e maior custo da população

Data: $C, nos, Nger, key, Z$
C - matriz para armazenar cada par de táxons e total de posições que não geram custo
nos - número de nós da árvore $Nger$ - contador para o número de configurações geradas
key - controle para o teste de parada Z - vetor de configuração

```
14.1  begin
14.2      nos ← 2m − 2;      Nger ← 0
14.3      for i ← 1 to m − 1 do
14.4          for j ← i + 1 to m do
14.5              Nger ← Nger + 1      C_{Nger,3} ← 0
14.6              for k ← 1; to n do
14.7                  if (M(i,k) + M(j,k) ≠ 1) then
14.8                      C_{Nger,1} ← i;   C_{Nger,2} ← j
14.9                      C_{Nger,3} ← C_{Nger,3} + 1
14.10                 endif
14.11             endfor
14.12         endfor
14.13     endfor
14.14     if Nger > 𝒩 then Nger ← 𝒩
14.15     OrdenaDecrescente (C)   { chave = C_{i,3} },   ∀i = 1...Nger
14.16     key ← .true.;   j ← m;   k ← 0
14.17     upperbound ← huge   maxcost ← 0
14.18     while key do
14.19         key ← .false.;   Z ← 0
14.20         for i ← 1 to Nger do
14.21             if C(i,3) ≠ 0 then
14.22                 key ← .true.;   j ← j + 1
14.23                 Z(C_{i,1}) ← j;   Z(C_{i,2}) ← j
14.24                 i ← i + 1;   C_{i,3} ← 0
14.25             endif
14.26         endfor
14.27         f ← cost(m, n, z, nos, huge, ℳ)
14.28         Complementa (Z);      Atualiza (uppercost, maxcost, f)
14.29         k ← k + 1;   P(k) ← {Z, f}
14.30     endw
14.31     while k < 𝒩 do
14.32         Gera_Rand (Z);      f ← cost(m, n, z, nos, huge, ℳ)
14.33         Atualiza (uppercost, maxcost, f)
14.34         k ← k + 1;   P(k) ← {Z, f}
14.35     endw
14.36 end
```

Algoritmo 15: Gera Configurações Aleatórias para m Problema de Filogenia a ser tratado pela Heurística de Busca Local proposta – corresponde à rotina Gera_Rand do módulo Gera_Pop. A função rnd(a,b) gera uniformemente valores inteiros no intervalo [a,b] [PM88]

15.1 **subroutine** *Gera_Rnd*(m, nos, Z)

Input: $m, n, \mathscr{M}$
$\mathscr{M}$ – Matriz com m táxons e n características pertencentes ao conjunto $R = \{0,1,2\}$

Output: Z
Z - vetor de configuração

Data: W
W - vetor de trabalho

15.2 **begin**
15.3 $Z \leftarrow 0$; $W \leftarrow 0$
15.4 **for** $i \leftarrow nos-1$ **downto** $m+1$ **do**
15.5 $k \leftarrow rnd(i, nos+1)$
15.6 **if** $W(k) < 2$ **then**
15.7 $W(k) \leftarrow W(k)+1$; $Z(i) \leftarrow k$
15.8 **endif**
15.9 **endfor**
15.10 **for** $i \leftarrow 1$ **to** m **do**
15.11 $k \leftarrow rnd(m+1, nos+1)$
15.12 **if** $W(k) < 2$ **then**
15.13 $W(k) \leftarrow W(k)+1$; $Z(i) \leftarrow k$
15.14 **endif**
15.15 **endfor**
15.16 Complementa (Z)
15.17 return(Z)
15.18 **end**

6.6.2 Algoritmo Proposto

No Algoritmo 16, descrito a seguir, todos elementos da população gerada são recombinados dois a dois obtendo uma vizinhança variável de 12 configurações distintas utilizando os procedimentos descritos na subseção 6.5.3; o processo de busca local seleciona a "melhor" configuração destas vizinhanças para a etapa final de intensificação com melhoria iterativa.

6.6.3 Complexidade do Algoritmo

Teorema 6. *O Algoritmo 16 tem complexidade de tempo $O(\mathscr{K} m^3 n)$, onde $\mathscr{K}$ é o número de iterações para a etapa de melhoria.*

Demonstração. De acordo com o Teorema 4 a função *cost* tem complexidade $\mathscr{O}$(mn). A rotina *Generate_Pop* que gera a população inicial tem complexidade $\mathscr{O}(m^3 n)$ pois combina todos os m táxons dois a dois e calcula seus custos. Considerando que as constantes relativas ao tempo de execução dos comandos simples estão representadas pela letra c é fácil verificar que existem constantes reais positivas α, β e γ tais que, se $f_1(\mathscr{N},m,n)$ e $f_2(\mathscr{K},m,n)$ são funções que computam o tempo de processamento, respectivamente, dos laços externos *for*{i} e *for*{k} então:

$$f_1(\mathscr{N},m,n) = \mathscr{N}\{c + \frac{(\mathscr{N}-1)}{2}[c + 12(c+2mn)]\} \leq \beta \mathscr{N}^2 mn$$

$$f_2(\mathscr{K},m,n) = \mathscr{K}\{c + \alpha[c + (2m-3)*(m-2)(c+mn)]\} \leq \gamma \mathscr{K} m^3 n$$

logo, a função de complexidade de tempo do Algoritmo 16 é dado por:

$$f(\mathscr{N},\mathscr{K},m,n) \leq c + cm^3 n + \beta \mathscr{N}^2 mn + \gamma \mathscr{K} m^3 n \leq \alpha \mathscr{K} m^3 n \Rightarrow \mathscr{O}(\mathscr{K} m^3 n)$$

□

Algoritmo 16: Módulo Principal do algoritmo proposto

Input: $m, n, \mathscr{M}, \mathscr{K}, \mathscr{N}$
$\mathscr{M}$ – Matriz com m táxons e n características pertencentes ao conjunto $R = \{0,1,2\}$
$\mathscr{K}$ – número de iterações para melhoria $\mathscr{N}$ – tamanho da população

Output: z^*, s^*
s^* – "melhor" solução obtida
z^* – configuração correspondente à solução s^*, ou seja, s^* = $cost(m, n, z^*,,, M)$

Data: P, V - matrizes contendo configurações nv - número de vizinhos selecionados
nos - número de nós da árvore $upperbound$ - melhor resultado atual
$z_1, z_2, , z_3, z_4, z$ - vetores de configurações $c_1, c_2, , c_3, c_4, c$ - custos

16.1 **begin**
16.2 $nos \leftarrow 2m-2$; $nv \leftarrow 0$
16.3 Generate_Pop(P, $\mathscr{N}$, $upperbound$, $maxcost$)
16.4 $s^* \leftarrow upperbound$
16.5 **for** $i \leftarrow 1$ **to** $\mathscr{N}-1$ **do**
16.6 $(z_1, c_1) \leftarrow P(i, 1:n)$
16.7 **for** $j \leftarrow i+1$ **to** $\mathscr{N}$ **do**
16.8 $(z_2, c_2) \leftarrow P(j, 1:n)$; $c \leftarrow \infty$
16.9 **for** $k \leftarrow 1$ **to** 12 **do**
16.10 $(z_3, z_4) \leftarrow crossover\_case\{k\}(z_1, z_2)$
16.11 $c_3 \leftarrow cost(m, n, z_3, nos, upperbound, M)$
16.12 $c_4 \leftarrow cost(m, n, z_4, nos, upperbound, M)$
16.13 $l = \{ind \mid c_{ind} = min(c_1, c_2, c_3, c_4)\}$
16.14 **if** $c > c_l$ **then** $c \leftarrow c_l$; $upperbound \leftarrow c_l$; $z \leftarrow z_l$
16.15 **if** $s^* \geq c_l$ **then** $s^* \leftarrow c_l$; $z^* \leftarrow z_l$; update(P)
16.16 **endfor**
16.17 $nv \leftarrow nv+1$; $V(nv) \leftarrow z$
16.18 **endfor**
16.19 **endfor**
16.20 **if** $nv < \mathscr{K}$ **then** $\mathscr{K} \leftarrow nv$
16.21 **for** $k \leftarrow 1$ **to** $\mathscr{K}$ **do**
16.22 $ind \leftarrow select(1, nv)$; $z \leftarrow (V(ind);)$ $improve \leftarrow 2$
16.23 **while** $improve > 1$ **do**
16.24 $improve \leftarrow 0$
16.25 **for** $i \leftarrow 1$ **to** $nos-2$ **do**
16.26 **for** $j \leftarrow i+1$ **to** $nos-1$ **do**
16.27 **if** $(z(i) \neq z(j))$ *and* $(z(i) > j))$ **then**
16.28 $swap$(z(i),z(j))
16.29 $c \leftarrow cost(m, n, z, nos, upperbound, M)$
16.30 **if** $s^* \geq c$ **then**
16.31 $s^* \leftarrow c$; $z^* \leftarrow z$;
16.32 $improve \leftarrow improve+1$; $update(P)$
16.33 **else**
16.34 $swap$(z(i),z(j))
16.35 **endif**
16.36 **endif**
16.37 **endfor**
16.38 **endfor**
16.39 **endw**
16.40 **endfor**
16.41 **end**

6.7 Experimentos Computacionais II

6.7.1 Recursos Computacionais utilizados

A máquina utilizada foi um Intel/Celerom 1.6Ghz e 512Mb de RAM, um pouco inferior àquela utilizada nos testes comparativos da Tabela 6.6 (Pentium IV, 2Ghz e 512Mb). A maioria das rotinas foram escritas em Fortran PowerStation 4.0 (Microsoft Developer Studio, 1995) e algumas delas na linguagem C (Microsoft Visual C++).

6.7.2 Modelo das Instâncias

Duas classes de instâncias foram testadas, a primeira refere-se às mais utilizadas nos testes de problemas de filogenia baseada em características com resultados em Goloboff [Gol96], Andreatta & Ribeiro [AR02] e Ribeiro & Vianna [RV05], enquanto que a segunda classe contém uma miscelânea de instâncias, quatro delas também encontradas em [RV05], outra corresponde àquela apresentada na Figura 6.3 (página 118) e por último, mais três geradas para verificar como o algoritmo se comporta no caso de filogenia perfeita e qual a variação de tempo de execução quando há modificação somente do percentual de indefinição mantendo o tamanho da instância.

Para cada instância são obtidos, em função de suas características, os parâmetros relativos ao tamanho da população inicial gerada ($Npop$) e seus valores de custo mínimo (Min) e máximo (Max) indicados na Tabela 6.6.

6.7.3 Comparação dos Resultados com Soluções conhecidas

O algoritmo foi executado para as instâncias no mínimo dez vezes cada uma, obtendo resultados próximos ou quase sempre iguais aos melhores apresentados na Tabela 6.6. A Figura 6.7 apresenta a árvore filogenética correspondente à instância i para a qual o algoritmo obteve o melhor resultado. A seguir citamos alguns comentários sobre os resultados obtidos.

- Como esperado, em função da complexidade do algoritmo (ver Teorema 6), seu tempo

de execução cresce proporcionalmente ao tamanho da entrada m (número de táxons) e n (número de características). Isto é comprovado pelos maiores valores absolutos de tempo, indicados pelas instâncias *iii*, *vii* e *viii* da Tabela 6.6 e Figura 6.8.

- Mantido o tamanho da entrada e alterando o percentual de indeterminação das características, o tempo é reduzido a medida que este valor aumenta (ver instâncias *iv*, *xv* e *xvi* da Tabela 6.6).

- Foi observado na fase de testes que o melhor resultado (instância *i*) foi conseguido na fase de melhoria iterativa, quando foi processada a troca de elementos indicada pela forma z_{10} da Figura 6.5.

- Os piores resultados (instâncias *vi*, *ix* e *x*) decorrem do pequeno tamanho da população, porque durante a geração inicial não foi obtida uma grande variedade de configurações iniciais com custos distintos.

- Por fim, verifica-se que, quando a filogenia é perfeita [SM97] o algoritmo é bem rápido (instância *xiv*), pois na geração inicial os táxons com características comuns são dispostos em vértices próximos indicando suas similaridades.

6.7.4 Estratégias de Redução do Tempo de Execução

Uma primeira redução de tempo de execução deve-se à transformação do conjunto de estados do tipo caráter $\mathscr{R}=\{0,1,?\}$ para numérico $\mathscr{R}'=\{0,1,2\}$. Com isso foi possível codificar o Algoritmo 11 usando, no pior caso, somente duas comparações ao invés de no mínimo três, ocasionando uma redução em torno de 33% nas inúmeras avaliações de *cost_node*, considerando que sua função de complexidade de tempo (αmn) é fator da função de complexidade da heurística ($\beta m^3 n$).

Outra redução significativa pode ser observada pelos resultados apresentados na Figura 6.9 que mostra que mais de 90% das avaliações ultrapassam o valor do parâmetro *Min* obtido no processo de geração das configurações inciais. Assim, foi incluído um parâmetro

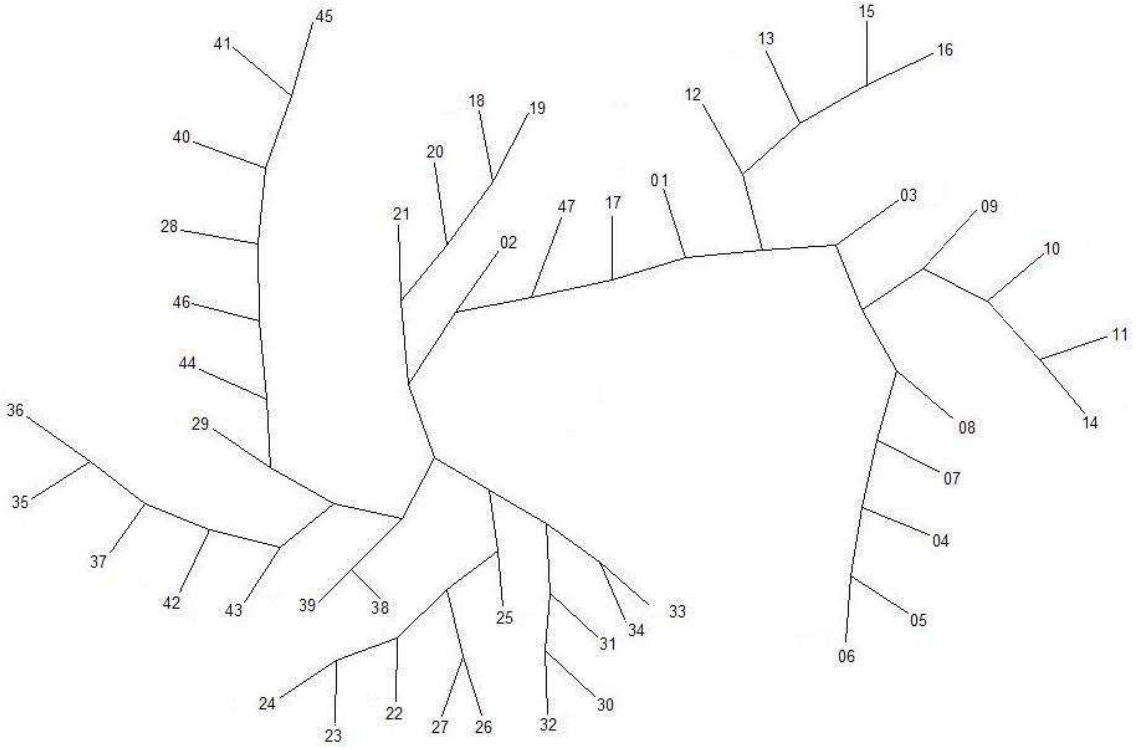


Figura 6.7 Árvore Filogenética correspondente à instância que a heurística obteve o melhor resultado (*i*-GRIS) [RV05]

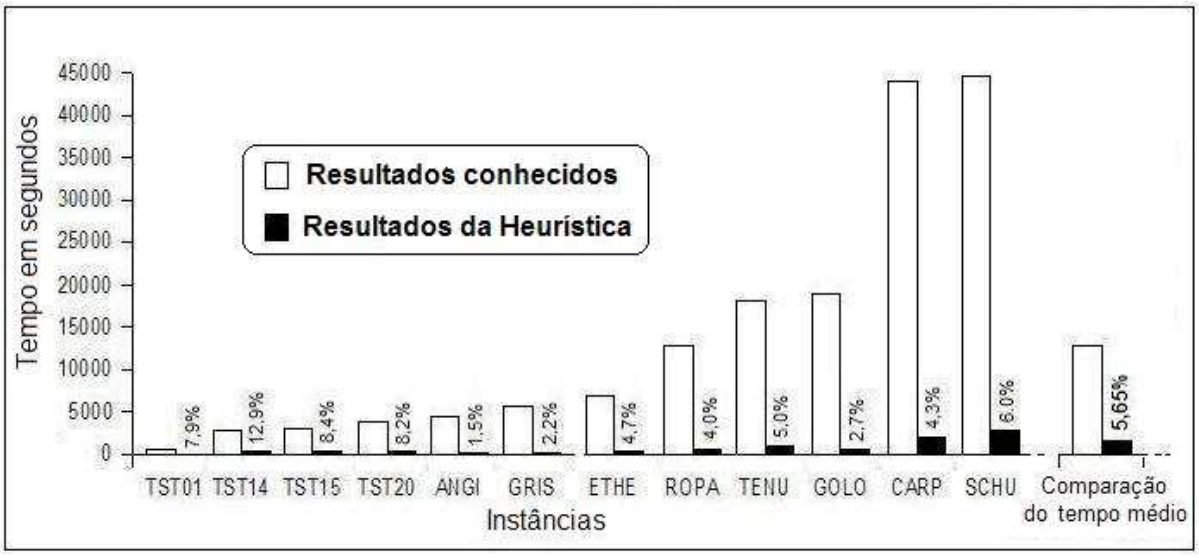


Figura 6.8 Comparação do tempo de execução do algoritmo para as instâncias com resultados conhecidos

Tabela 6.6 Resultados obtidos pelo algoritmo proposto. As instâncias comparativas de *i* a *xii* foram retiradas de [AR02, Gol96, RV05], a instância *xiii* é um exemplo contido em [Amo97, p.85] e as demais foram geradas, onde a *xiv* corresponde a um exemplo de Filogenia Perfeita e as duas últimas são alterações da instância *iv* com inclusão, respectivamente, de 20% e 40% de indeterminações. $Npop$ é o número de configurações iniciais distintas utilizadas como base do processo de busca, distribuídas no intervalo $[Min, Max]$. *Perc* é o percentual de indeterminação da instância

Instâncias Comparativas					Solução Conhecida	Algoritmo Proposto					Aproximação
						População Inicial			Solução		
N^{o}	Referência	m	n	Perc	Melhor	Npop	Min	Max	t(s)	Valor	Gap
i	GRIS	47	93	43	172	113	254	510	128	172	1,0000
ii	ANGI	49	59	9	216	89	303	444	66	217	1,0046
iii	TENU	56	179	9	682	209	1149	1720	952	685	1,0044
iv	ETHE	58	86	0,1	372	160	515	936	338	374	1,0054
v	ROPA	75	82	14	325	151	466	767	533	330	1,0154
vi	GOLO	77	97	20	496	110	712	947	527	504	1,0161
vii	SCHU	113	146	0,9	759	180	1627	2831	2832	760	1,0013
viii	CARP	117	110	16	548	140	1064	1632	1966	550	1,0036
ix	TST01	45	61	20	547	85	649	773	45	559	1,0214
x	TST14	67	99	30	1173	124	1396	1619	416	1205	1,0273
xi	TST15	69	77	40	765	108	956	1131	273	772	1,0092
xii	TST20	75	79	50	673	105	880	1031	347	677	1,0059
Médias					—	131	—	—	702	—	1,0096
Outras Instâncias					Solução Conhecida	Algoritmo Proposto					Aproximação
						População Inicial			Solução		
xiii	Amo97	8	12	0	15	12	16	27	0,1	15	1,0000
xiv	FP00	35	52	0	52	51	67	140	8	52	1,0000
xv	ETHE20	58	86	20	—	141	440	750	234	309	—
xvi	ETHE40	58	86	40	—	104	359	558	152	256	—

(*upperbound*) que corresponde ao melhor resultado "atual" obtido na evolução do processo de busca. O Algoritmo 12 computa, passo a passo, a parcimônia da árvore analisada e quando seu valor parcial ultrapassar esse parâmetro a função é encerrada retornando $cost = \infty$. Simulações mostraram que essa redução gira em média em torno de 60%. Desta forma o valor da redução de tempo total esperado, somente nas avaliações da função objetivo, ultrapassa a 90%. A Figura 6.9 comprova esse desempenho em relação ao tempo de execução.

6.8 Fluxo Geral para construir Filogenias

Os resultados dos experimentos computacionais e outras simulações mostraram que através de deslocamento de blocos foi possível diversificar a busca pela solução ótima do problema para cada instância. Foi observada que a permuta de três elementos de uma configuração traz bons resultados. Como este procedimento é realizado de forma restrita, optamos por usar a programação paralela com o objetivo de expandir a vizinhança da busca local.

No capítulo seguinte mostraremos a estratégia utilizada pelo procedimento de melhoria iterativa a partir do resultado obtido pela heurística aqui apresentada. Desta forma sintetizamos no Algoritmo 17 o fluxo geral de todas as técnicas e estrutura criadas com o objetivo único de resolver o problema da filogenia baseada em características.

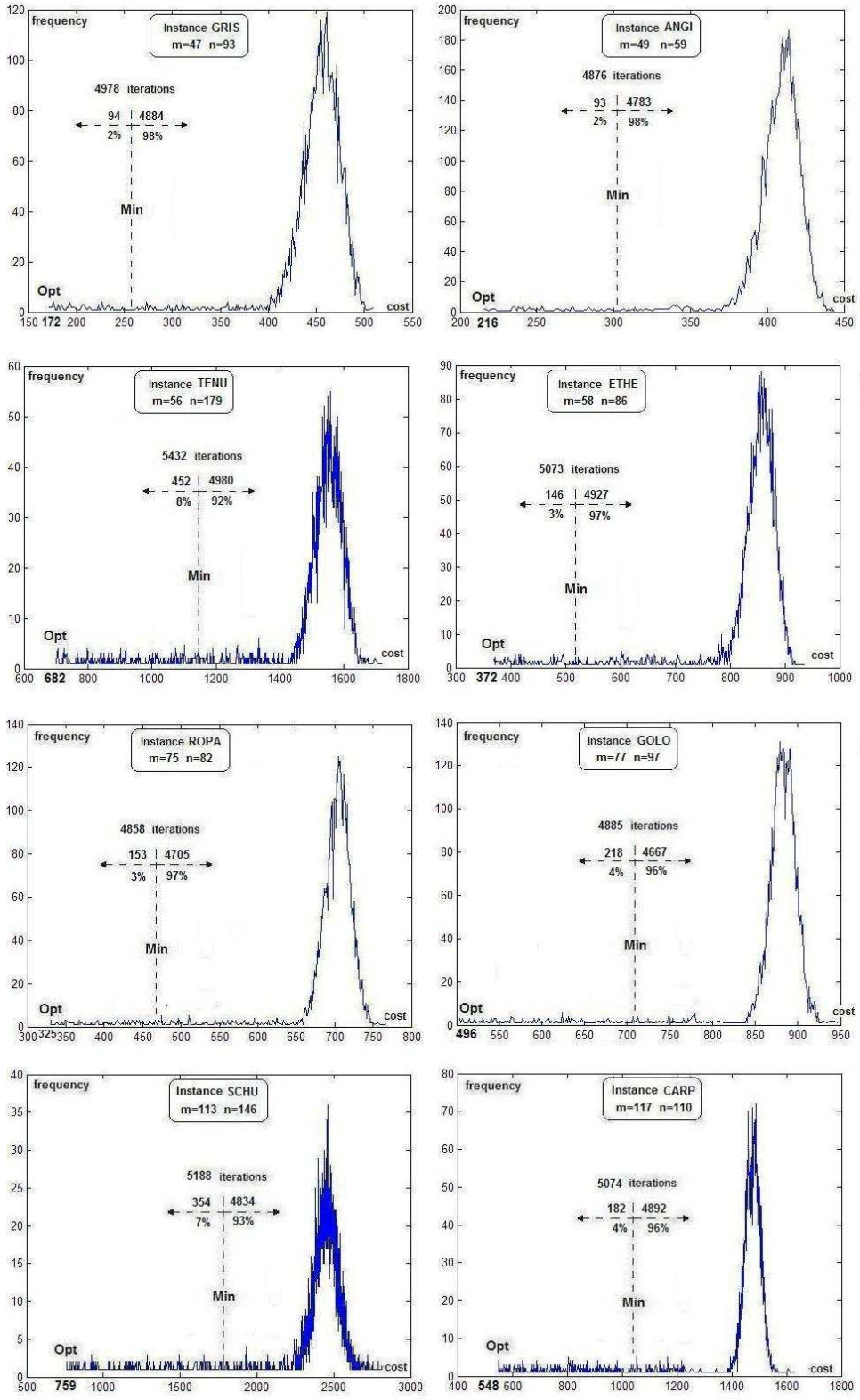


Figura 6.9 Resultados das simulações para as instâncias *i* a *viii* da Tabela 6.6. *Min* é o parâmetro "*upperbound*" da função *cost* do Algoritmo 12 que foi alterado para ∞ com o objetivo de que todos seus valores fossem calculados para permitir analisar a freqüência dos resultados obtidos. *Opt* corresponde à melhor solução conhecida.

Algoritmo 17: Fluxo Geral para geração de Árvores Filogenéticas. Para filogenias perfeitas (decisão do Algoritmo 3), A é construída pelo Algoritmo 6, caso contrário, ela é inferida pelos Algoritmos 16 (Heurística Serial), 21 (programa Mestre) e 22 (programa Escravo)

Input: m, n,M – Matriz com m táxons e n características

Output: A – árvore filogenética correspondente à matriz M

Data: P - matriz contendo configurações da população inicial

Z_i, Z^* - vetores de configuração

17.1 **begin**

17.2 **if** *Filogenia_Perfeita* (M) **then**

17.3 Constrói_Filogenia $(M \longrightarrow Z^*)$

17.4 Desenha $(A \longleftarrow [M\ ,\ Z^*]\ )$

17.5 **else**

17.6 Gera_População_Inicial $(P \longleftarrow M\ )$

17.7 Heurística_Serial $([M\ ,\ P] \longrightarrow Z_i]$

17.8 Processos_Mestre_Escravo $([M\ ,\ Z_i] \longrightarrow Z^*)$

17.9 Modifica $([M\ ,\ Z^*] \longrightarrow M)$

17.10 Desenha $(A \longleftarrow [M\ ,\ Z^*]\ )$

17.11 **endif**

17.12 **end**

CAPÍTULO 7

Paralelização de Busca Local para o Problema da Filogenia

7.1 Computação de Alto Desempenho

7.1.1 Introdução ao Processamento Paralelo

A computação de alto desempenho [Via98, Ble96] tem sido uma ferramenta bastante poderosa para todos os pesquisadores. Técnicas de programação vetorial e paralela têm resolvido grandes desafios computacionais, num tempo cada vez menor, utilizando arquiteturas especiais que integram vários conceitos de *hardware* e *software*.

Desta forma, processos concorrentes podem utilizar multiprocessadores fortemente conectados com uso de memória compartilhada ou distribuída com o intuito de melhorar o desempenho das máquinas e reduzir o tempo de processamento das tarefas.

O termo paralelismo refere-se ao processamento simultâneo existente na execução de programas, sub-rotinas e instruções. Mesmo em sistemas com um único processador há implementada alguma forma de paralelismo em nível de *hardware*.

As aplicações desenvolvidas com esse enfoque utilizam redes de computadores que são voltadas para o processamento "paralelo", onde um "ponto/estação" executa suas tarefas independentemente e transmite seus resultados para um servidor central. Nosso enfoque aqui está relacionado com a utilização simultânea de vários processadores, numa mesma máquina ou em máquinas distintas, para executar um dado programa com redução significativa do tempo de execução se comparado com sua versão serial.

O processamento paralelo sempre é possível quando se observa num programa serial, por vezes chamados seqüencial, duas ou mais seções realizando processos independentes. Diversos trabalhos para o Problema da Filogenia [SLMW02, SHB+01, KNT+05, Jon94] utilizam esta técnica de programação.

7.1.2 Arquiteturas de Sistemas Paralelos

Existem três tipos de arquiteturas que permitem alguma forma de paralelização:

i) Computadores Seqüenciais

Computadores convencionais com um único processador operam com uma seqüência simples de instruções sobre uma seqüência simples de dados sendo por isso classificados como SISD (*Single Instruction, Single Data*). A velocidade de processamento nestas máquinas depende de dois fatores: o tempo necessário para troca de informações entre a memória e a CPU (*Central Processing Unit*) e para a execução das instruções.

Para aumentar a performance destes computadores e introduzir alguma forma de paralelismo são usados os seguintes mecanismos:

- *"interleaving"* da memória - blocos de memória acessados de forma independente;
- *"pipelining"* de instruções - o processador através de unidades funcionais específicas pode executar simultaneamente mais de uma fase de execução de uma instrução (busca, decodificação, cálculo de endereços e execução);
- memória *"cache"* - memória relativamente pequena e extremamente rápida que funciona como um *buffer*[1] da memória primária e será o local onde o processador primeiro busca a instrução a ser executada.

  A Figura 7.1 mostra um esquema representativo destes mecanismos com opção de utilização de um processador simples ou com *"pipelining"* de "n" estágios.

ii) Processamento Vetorial

Arquiteturas paralelas referidas como SIMD (*Single Instruction, Multiple Data*) funcionam com uma única unidade de controle enviando para várias unidades de processamento uma mesma instrução que é executada de forma síncrona, manipulando elementos de um ou mais vetores (ou matrizes); a Figura 7.2 dá uma idéia do funcionamento deste tipo de processamento.

O processador vetorial utiliza registradores vetoriais que permitem num único ciclo de busca e execução da instrução processar várias operações simultaneamente. Podemos ainda

[1] região de memória temporária utilizada para escrita e leitura de dados [Wik06a]

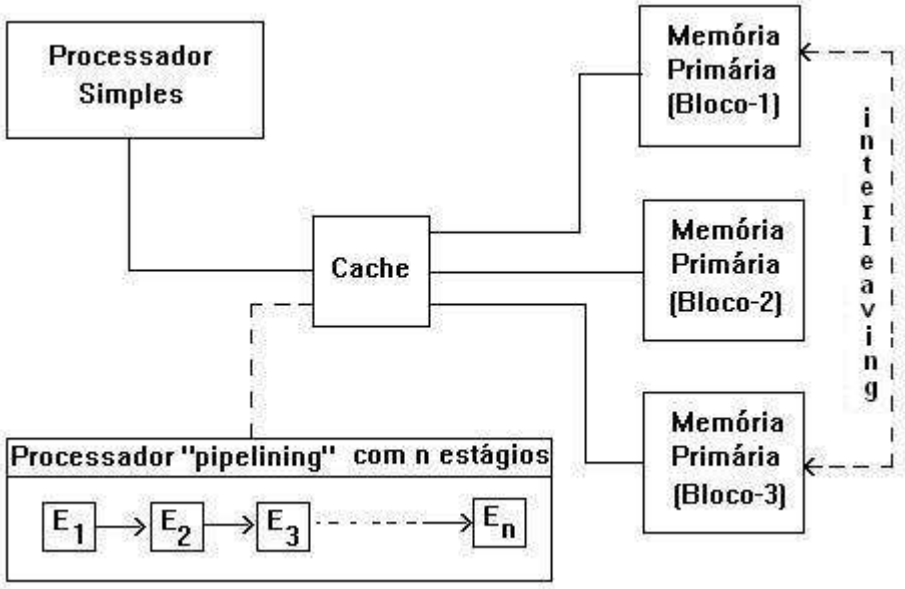


Figura 7.1 Mecanismos de Paralelismo para computadores SISD

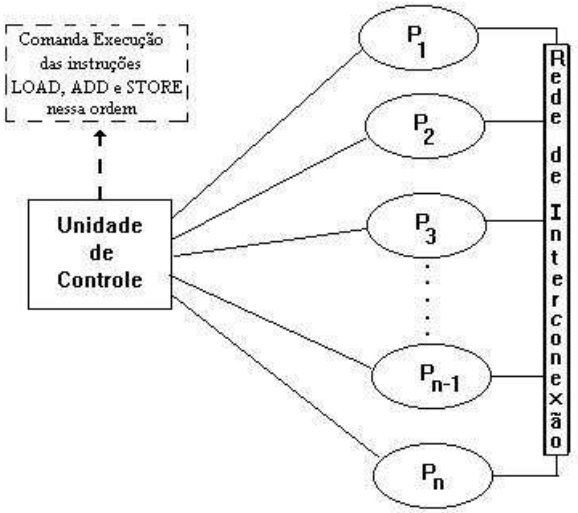


Figura 7.2 Processamento Vetorial - Arquitetura SIMD

acrescentar que esse tipo de processamento, presente em supercomputadores, só traz vantagens em aplicações com elevado grau de vetorização como alguns problemas numéricos, processamento de imagens, estudos meteorológicos etc.

iii) Máquinas Paralelas

Máquinas com múltiplos processadores em que cada uma delas é capaz de executar um programa independente; elas são classificadas como computadores MIMD (*Multiple Instruction, Multiple Data*). Como já citado anteriormente, neste tipo de ambiente só existe ganho real de tempo se as tarefas puderem ser divididas em processos independentes para execução simultânea. A Figura 7.3 da página 143 mostra uma forma de utilização destes tipo de processamento.

7.1.3 Programação Paralela

Uma linguagem para programação paralela [Ble96] deve oferecer mecanismos para compartilhar informações entre processadores. Sendo um programa paralelo composto de dois ou mais conjuntos de instruções a serem executadas concorrentemente deve-se especificar como é feito seu particionamento e quais são os pontos e forma de sincronização e comunicação entre os processos. Com base nestas colocações relacionamos a seguir os seguintes paradigmas:

i) Programação Paralela Explícita x Implícita

A chamada programação paralela explícita requer esforço total do usuário que necessita codificar todas etapas do paralelismo com auxílio de bibliotecas especiais de procedimentos e funções. A tarefa do compilador não é diferente da usada para um programa serial, ou seja, simplesmente ele gera o código executável na forma prevista pelo programador. Os programas que usam rotinas do PVM (*Parallel Virtual Machine*) [Bar94, GBD94], nas linguagens Fortran, *C* e C^{++} [IBM94], enquadram-se nesta categoria. Esta é a técnica utilizada neste trabalho.

Na programação paralela implícita a tarefa do programador é facilitada pois a maior carga de paralelização é feita pelo compilador. Entretanto, o mapeamento de um programa serial em paralelo não garante a geração de um código eficiente, pois esta conversão não é totalmente automática. O compilador particiona o programa serial em blocos e analisa a dependência entre eles, como o controle das estruturas de loops, desvios e chamadas a subrotinas. A linguagem

Fortran "D", por exemplo, pode ser usada para a programação implícita.

ii) Memória Compartilhada x Passagem de Mensagens

No paradigma de programação com uso de memória compartilhada os processos paralelos acessam variáveis comuns; cada processador lê e grava os dados numa única memória, de forma que as linguagens devem tratar dos problemas chamados de "exclusão-mútua" em que, se um determinado processo está alterando o conteúdo de uma variável, um outro somente deve acessá-la após a conclusão da operação que estava sendo realizada.

Na passagem de mensagens cada processo tem suas variáveis locais. Os valores destas variáveis podem ser transferidos entre os processos através de instruções de envio e recebimento de dados. Existem as "*Message-Passing Libraries*" que realizam este procedimento utilizando um protocolo de rede. Esta é a forma aqui utilizada.

iii) Dados x Controle

Programas que computam múltiplos dados podem ser particionados de forma que, cada processo execute a computação parcial desejada. Estes são chamados *"data-parallel programs"* e contêm uma seqüência simples de instruções a serem executadas de forma distinta em função dos dados, ou parâmetros passados para cada processador que participa da paralelização no início de sua execução, conforme utilizamos na subseção 7.2.4.

Em Kumar et al. [KGKG03] encontramos as linguagens *Dataparallel C e C** classificadas nessa categoria. O chamado *"control parallelism"* refere-se a execução simultânea de diferentes conjuntos de instruções aplicados a diferentes conjuntos de dados, dispostos em módulos especiais, cujos resultados parciais são passados de um para outro numa forma predefinida.

7.1.4 Parallel Virtual Machine – PVM

O Projeto PVM [Bar94, GBD94] foi desenvolvido em 1989 pelo *Oak Ridge National Laboratory*. Suas versões são de domínio público, sendo distribuídas livremente e usadas em muitas aplicações científicas em vários centros de pesquisa.

Este sistema dá suporte a aplicações escritas em Fortran, C ou C++, que possam ser paralelizadas através do processo de troca de mensagens e do uso de memória distribuída. A idéia

básica é que uma determinada tarefa seja particionada em outras menores a serem executadas simultaneamente por processadores distintos. A comunicação entre as diversas máquinas envolvidas no processo de paralelização é possível devido à utilização de subrotinas escritas nestas linguagens.

O PVM é um sistema que também permite que uma rede de computadores heterogêneos possa ser usada, sob ambiente Unix, como se fosse uma única máquina agregando todos recursos destes computadores. Os processadores das máquinas configuradas para o processamento paralelo devem estar conectados por hardware específico através do uso de *switches* ou via redes baseadas no protocolo IP (*Internet Protocol*). Os recursos computacionais utilizados em nossos testes (ver subseção 7.3.1) têm essas características.

Na distribuição de tarefas durante a execução de um programa paralelo, o número de processos ou tarefas (NTASKS), a serem executadas, independe do número de processadores (NPROC) utilizados. É claro que se NTASKS < NPROC haverá processador ocioso, o que não é desejável, portanto essa situação em geral não deve ocorrer.

A Figura 7.3 dá uma idéia da forma do processamento paralelo para um fluxo de execução de vários processos. O processador da máquina local (aquela em que o usuário está conectado) necessariamente fará parte da paralelização, ou seja, temos no exemplo quatro máquinas das quais uma delas é responsável tanto pela execução de algumas tarefas disparadas como pelo controle geral do processo, envio e recebimento de dados, sincronização, comunicação e tratamento das informações.

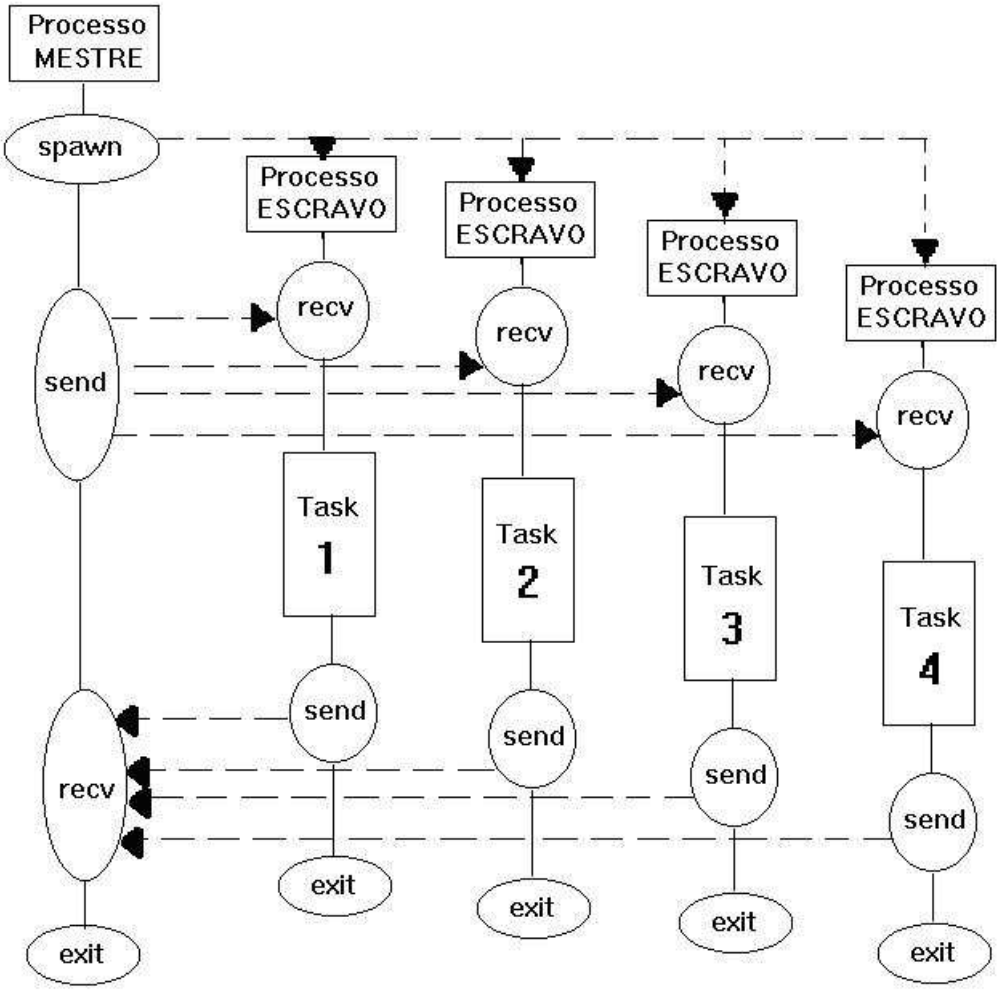


Figura 7.3 Fluxo com quatro processos para o modelo de programação paralela do tipo "Mestre/Escravo" utilizado no desenvolvimento e implementação do algoritmo proposto

7.2 Algoritmo Paralelo para o Problema da Filogenia

7.2.1 Procedimento de Melhoria Iterativa

Observa-se em algumas recombinações descritas na Subseção 6.5.3, e mostradas na Figura 6.5 na página 122, que uma troca simultânea de posições entre dois ou três elementos do vetor de configuração acarreta alterações [Rod03], às vezes simples, outras vezes bruscas, na topologia da árvore e, conseqüentemente em sua parcimônia.

O Algoritmo 16 (página 128) faz todas as combinações com duas permutas, porém aquelas com três permutas são feitas de forma aleatória e eventualmente. Esse último procedimento foi evitado por causa da alta complexidade da heurística, $O(m^3n)$. No caso, se fossem incluídas todos os arranjos possíveis com três elementos a ordem passaria para $O(m^4n)$, tornando seu tempo de execução inadmissível para a maioria das instâncias.

Por exemplo, considerando que a notação Z_{ijk}, para o vetor de configuração Z, indica que as três posições "i, j, k" estão dispostas nessa ordem (da mesma forma para duas posições Z_{ij}), pode-se afirmar que no trecho de algoritmo a seguir, a rotina *Permuta2* (Algoritmo 18) que troca Z_{ij} por Z_{ji} é executada $\frac{n(n-1)}{2}$ vezes, contribuindo com $O(n^2)$ para a complexidade do algoritmo que o contém.

Algoritmo 18: Bloco que permuta dois elementos do vetor de configuração

```
18.1 for i ← 1 to n − 1 do
18.2     for j ← i + 1 to n do
18.3         Permuta2(Z_ij)
18.4     endfor
18.5 endfor
```

De modo similar, a rotina *Permuta3* (Algoritmo 19), que gera todos os 6 arranjos de Z_{ijk} contribuiria com $O(n^3)$ para um dado algoritmo.

O procedimento de melhoria iterativa, proposto nesta seção através da programação paralela, utiliza a rotina *Permuta3* reduzindo sua complexidade para $O(n^2)$. A alteração para atingir

Algoritmo 19: Bloco que permuta três elementos do vetor de configuração

```
19.1 for i ← 1 to n − 2 do
19.2     for j ← i + 1 to n − 1 do
19.3         for k ← j + 1 to n do
19.4             Permuta3(Z_ijk)
19.5         endfor
19.6     endfor
19.7 endfor
```

esse objetivo consiste em modificar o bloco do Algoritmo 19 para a forma do Algoritmo 20.

Algoritmo 20: Rotina de Permuta de três elementos do vetor de configuração com ordem de complexidade quadrática (bloco contido no Programa Escravo)

```
20.1 input(i)
20.2 for j ← i + 1 to n − 1 do
20.3     if i ≠ j then
20.4         for k ← j + 1 to n do
20.5             if (i ≠ j).and.(i ≠ k) then
20.6                 Permuta3(Z_ijk)
20.7             endif
20.8         endfor
20.9     endif
20.10 endfor
```

O procedimento *Permuta3* gera 5 configurações distintas **viáveis** a partir da atual Z_{ijk}, quais sejam: Z_{ikj} , Z_{jik} , Z_{jki} , Z_{kij} , Z_{kji}. De modo que seria correto afirmar que a contribuição para a complexidade do algoritmo que utiliza essa rotina é $5O(n^2)$. Este fator será levado em consideração no cálculo da complexidade do algoritmo proposto, na subseção 7.2.4.

A seguir verifica-se que este bloco é incluído no programa que busca uma melhoria de uma

dada solução, chamado "escravo", e o valor i, referenciado como entrada acima, será recebido do programa chamado "mestre".

7.2.2 Estratégia de Paralelização de Busca Local

Em nossa implementação do algoritmo paralelo usamos uma estratégia de tarefas múltiplas independentes entre "$Nproc$" processadores disponíveis. Cada processador recebe uma cópia do mesmo programa escravo (*Slave*) juntamente com os dados de entrada enviados pelo programa mestre (*Master*).

As tarefas são distribuídas uniformemente entre os processadores de modo que seus tempos de execução sejam aproximadamente iguais. Os seguintes dados são recebidos por cada processo: a matriz de características $\mathscr{M}$, uma configuração de "partida" Z_0 e uma posição i que indica que $Z_0(i)$ é um dos elementos do tipo de permuta referenciada na subseção 7.2.1. Cada processo escravo i faz uma busca local e devolve ao processo mestre a melhor (menor) solução encontrada $S_i = f(Z_i^*)$, onde f é a função custo (valor da parcimônia) da árvore correspondente à configuração Z_i^*. A Figura 7.4 dá idéia dessa forma de processamento.

Como maiores vizinhanças são exploradas por conta da paralelização esperam-se resultados diferentes de cada busca local independente, visto que o programa mestre controla a diversificação destas vizinhanças. Para um determinado grupo de processadores é enviada uma mesma configuração de partida Z_0, para outros, podem ser enviados configurações distintas.

O valor (ou valores) de Z_0 são aqueles fornecidos como solução da heurística definida no capítulo anterior. A opção de ser uma ou mais depende se para uma dada instância, referida heurística fornece soluções distintas, ou não. Ao receber todos as soluções fornecidas pelos processos escravos, o programa mestre escolhe a melhor delas, no caso a de menor custo, como resultado final.

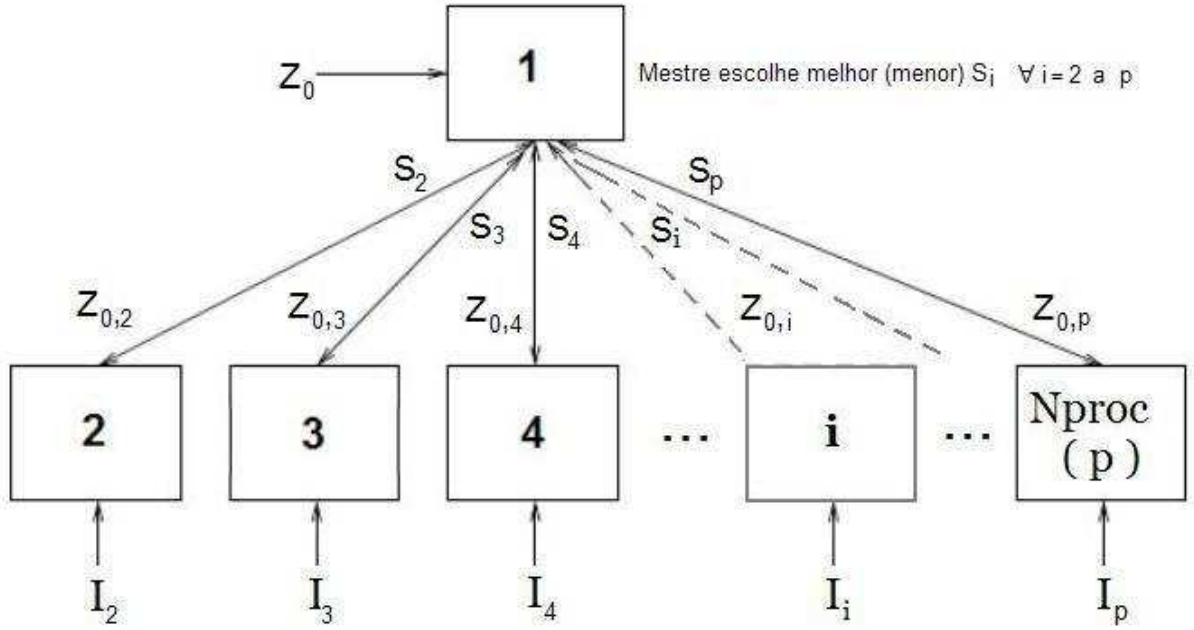


Figura 7.4 Fluxo com *Nproc* processos paralelos. A notação $Z_{0,i}$ refere-se aos dados enviados para cada processador escravo "i". Estes dados incluem a matriz de característica (instância do problema) de ordem $m \times n$ e a configuração de partida (que pode ser igual ou distinta para cada "i"). O valor de I_i é uma semente (*seed*) para a função de geração de números randômicos. S_i é a solução obtida por cada processo "i" e enviada ao processador 1 (mestre)

7.2.3 Módulos Mestre e Escravo

A seguir apresentamos os Algoritmos 21 e 22 correspondentes respectivamente aos programas mestre e escravo.

Algoritmo 21: Programa Mestre

Input: $m, n, \mathscr{M}, z$
$\mathscr{M}$ – Matriz com m táxons e n características
z – configuração incial para melhoria

Output: z^*, s^*
s – solução parcial
z – configuração correspondente à solução s
s^* – "melhor" solução obtida
z^* – configuração correspondente à solução s^*

Data: Tid – vetor de identificação das tarefas (programa escravo)
$Nproc$ – número de processos
nos – número de nós da árvore
$seed$ – parâmetro inicial para a função de randomização

```
21.1  begin
21.2      Nproc ← m − 3
21.3      spawn_slaves(Nproc,Tid)                (dispara Nproc escravos)
21.4      if mod(m,2) = 0 then
21.5          k ← 2m − 5
21.6      else
21.7          k ← 2m − 4
21.8      endif
21.9      for i ← 1 to Nproc do
21.10         i2 ← k − i1
21.11         seed ← random()
21.12         send(seed,i1,i2,m,n,z,M,Tid(i))      (envia dados)
21.13     endfor
21.14     ≪ waiting ≫          (espera pelos resultados dos processos escravos)
21.15     s* ← ∞
21.16     for i ← 1 to Nproc do
21.17         recv(s,z)          (recebe dados)
21.18         if s < s* then
21.19             s* ← s
21.20             z* ← z
21.21         endif
21.22     endfor
21.23 end
```

Algoritmo 22: Programa Escravo

Input: $m, n, \mathscr{M}, z_0$
$\mathscr{M}$ – Matriz com m táxons e n características
z_0 – configuração incial para melhoria

Output: z, s
s – "melhor" solução obtida
z – configuração correspondente à solução s
$custo$ – custo da solução atual

Data: z_1 – vetor de configuração auxiliar
nos – número de nós da árvore

```
22.1  begin
22.2      recv(seed,ind(1),ind(2),m,n,z0,M)          (recebe dados)
22.3      nos ← 2m − 2
22.4      s ← cost(z0,m,n,nos,∞,M)
22.5      z ← z0
22.6      for ivez ← 1 to 2 do
22.7          i ← ind(ivez)
22.8          for j ← i + 1 to nos − 2 do
22.9              for k ← j + 1 to nos − 1 do
22.10                 z1 ← z0
22.11                 for l ← 1 to 5 do
22.12                     Permuta3(z1,i,j,k,l)
22.13                     if .not.viavel(z1) then
22.14                         Permuta3(z1,random(seed),j,k,l)
22.15                     endif
22.16                     if viavel(z1) then
22.17                         custo ← cost(z1,m,n,nos,s,M)
22.18                         if custo < s then
22.19                             s ← custo
22.20                             z ← z1
22.21                         endif
22.22                     endif
22.23                 endfor
22.24             endfor
22.25         endfor
22.26     endfor
22.27     send(s,z)                  (envia dados)
22.28 end
```

7.2.4 Análise da Complexidade do Algoritmo Paralelo

Teorema 7. *O Algoritmo 21 (mestre) tem complexidade de tempo $O(m)$.*

Demonstração. Sejam as constantes c_1, relativa ao tempo de execução dos comandos externos aos dois laços **for**{i} do Algoritmo 21 e c_2 e c_3, respectivamente aos comandos internos dos mesmos laços e, sabendo que $Nproc = m-3$, sua função de tempo é dada por:

$$f(m) = c_1 + c_2 Nproc + c_3 Nproc \quad \Rightarrow \quad f(m) = c_1 + (m-3)(c_2 + c_3)$$

portanto a ordem de complexidade do Algoritmo 21 é $O(m)$.

□

Teorema 8. *O Algoritmo 17 (escravo) tem complexidade de tempo $O(m^3 n)$.*

Demonstração. Sejam as constantes c_1, relativa ao tempo de execução dos comandos externos ao laço **for**{j} do Algoritmo 22 e c_2 aos comandos internos do laço **for**{1} e, sabendo que $nos = 2m-2$ e a complexidade da função *cost* é $O(mn)$, sua função de tempo é dada por:

$$f(m,n) = c_1 + 5(c_2 + O(mn))(m-4)(m-3)/2 \Rightarrow f(m) = c_1 + 2.5(c_2 + mn)(m-4)(m-3)$$

portanto a ordem de complexidade do Algoritmo 22 é $O(m^3 n)$. □

Teorema 9. *O Algoritmo paralelo tem complexidade de tempo $O(m^3 n)$.*

Demonstração. Como o algoritmo paralelo é do tipo "mestre-escravo" e os programas são executados simultâneamente, sua complexidade será a composição dos dois Algoritmos 21 e 22, assim, sua complexidade será:

$$O(m) + O(m^3 n) = O(m^3 n)$$

□

7.3 Experimentos Computacionais III

7.3.1 Recursos Computacionais

Todos os testes, apresentados na Tabela 7.1, foram executados numa máquina paralela formada por 28 nós com processadores Intel Xeon DP e um *Cluster* com processador dual. Cada nó possui 1 *Gbytes* de memória RAM interligados por um *Switch Gigabit Ethernet*. Esta máquina está instalada no LIA (Laboratório de Pesquisa em Ciência da Computação) do Departamento de Computação da Universidade Federal do Ceará.

Os algoritmos foram implementados utilizando as rotinas do PVM (*Parallel Virtual Machine* versão 3.4.2 através da linguagem Fortran (compilador GNU/*g*77 versão 3.3.5).

Para algumas compilações e simulações, também foi utilizada a máquina instalada, na mesma Universidade, no LACAD (Laboratório de Computação de Alto Desempenho) que possui a seguinte configuração: um cluster formado por 4 nós com processadores AMD64 com 2.0 Ghz Dual Core, 2GB de memória RAM e disco rígido 70 *Gbytes* SCSI (*Small Computer System Interface*).

7.3.2 Modelo das Instâncias e Resultados Obtidos

As instâncias para os testes do algoritmo paralelo correspondem às mesmas utilizadas para a heurística do capítulo anterior acrescida de sua saída (solução).

A Figura 7.5 mostra um modelo de um arquivo de entrada para uma determinada instância. Se mais de uma configuração for disponível, a mesma deve ser acrescentada no final do arquivo. Por motivo de espaço, a instância não está mostrada em sua totalidade.

A Tabela 7.1 apresenta os resultados obtidos ao ser processado o algoritmo paralelo para todas as instâncias disponíveis. A análise destes resultados é feita na subseção seguinte.

```
58 86
01-Ancestor       0??00??0000???00?00000?0?0?000?0?1000001000?000000?0010??0?0??000????00?000001???000?1
02-Enyalioides    0?000001?????0?011??0??0???1??0????000????0?0?010?0??010??0000?0?0??0000000?0010????01
03-Morunasuarus   00000?0110????00?10?00?000?10100???0000?0??0?001000?0?10?0000000??00??0?00?0001??1110?0
04-Hoplocercus    0?0?0??100?0?0????0????00?0101001110?0?1???0?00?00??001?1?00?10??0?0??0?00001?????1?01
05-Oplurus        0????0000?0???0?0000?0?0???1000??10000010?0?0?0110??00??????00?1111???00????1110000?11
...               ...                                   ...                                          ...
...               ...                                   ...                                          ...
...               ...                                   ...                                          ...
54-Liolaemus      ?00??00?0??0?0?0???1????1?0?1?0?1?000??1??111?0?00??0???1?000???0??0?00??01?111???00?
55-Vilcunia       ??000?00?0?0?000?11?0?01??0??1?0?1????0?00?1?1?1??0?0?10000?000?0???0??1?1?1??0?00001
56-Ctenoblepharis ?0?00?0??00??00??1?1?1??00?1?10??1100??0??0?011??10110100000?0?0000000101111111???0?1?10
57-Phymaturus     0?0???0??0?000001101?001000?010??00?000001?00??00????01??00000000??0000011?01?101??0011
58-Leiocephalus   ?00?00000000????111?1???0001?100?0??0?01?01?0?1?0101?0?00?00000?00000?0??0?1001?0000?01
111  61  87  76  63  63  64  69  69  70  71  71  68  72  72  66  64  65  65  79
 80  81  84  78  78  85  79  82  60  62  60  89  90  90  91  96  93  94  92  92
 89  61  88 102 101  98 103 103 104 104 100 108 107  77  59  59 112 109  99  62
 76 113  86  67  66  67  68  73  70  74  75  73  74  75  77  87  86  80  81  82
 83  83  84  85 110  88 114 111  91  97  95  93  94  95  96  97  98  99 100 101
102 106 105 105 106 107 108 109 110 112 113 115 114 115 256
```

Figura 7.5 Formato do arquivo de entrada (parcial) do Algoritmo Paralelo para a instância ETHE40 ($m = 58$) e ($n = 86$). São apresentadas as características e rótulos dos táxons 01 a 05 e 54 a 58 bem como o vetor de entrada Z e seu custo (256)

Tabela 7.1 Resultados obtidos pelo algoritmo paralelo. As instâncias comparativas são idênticas àquelas contidas na Tabela 6.6 (página 132), exceto nos casos em que a filogenia era perfeita. Os resultados na coluna "melhor solução" para as duas últimas instâncias foram considerados os melhores conhecidos

Instâncias Comparativas					Melhor Solução	Soluções das Heurísticas					Redução P/S	Aproximação
						Serial (S)		Paralela (P)		Tempo		
N^{o}	Ref.	m	n	Nproc	Valor	t_1(s)	Valor	t_2(s)	Valor	$t_1 + t_2$	%	Fator
1	GRIS	47	93	44	172	128	172	98	172	226	—	1,0000
2	ANGI	49	59	46	216	66	217	72	217	138	—	1,0046
3	TENU	56	179	53	682	952	685	347	685	1299	—	1,0000
4	ETHE	58	86	55	372	338	374	192	374	530	—	1,0100
5	ROPA	75	82	72	325	533	330	505	328	1038	0,61	1,0092
6	GOLO	77	97	74	496	527	504	644	502	1171	0,40	1,0121
7	SCHU	113	146	110	759	2832	760	2085	760	4917	—	1,0013
8	CARP	117	110	114	548	1966	550	1753	550	3719	—	1,0036
9	TST01	45	61	42	547	45	559	69	555	114	0,72	1,0146
10	TST14	67	99	64	1173	416	1205	441	1202	857	0,25	1,0200
11	TST15	69	77	66	765	273	772	387	772	660	—	1,0092
12	TST20	75	79	72	673	347	677	569	677	916	—	1,0059
13	ETHE20	58	86	55	309	234	309	199	309	433	—	1,0000
14	ETHE40	58	86	55	252	152	256	227	252	379	1,56	1,0000
Médias					—	702	—	542	—	1244	0,71	1,0068

7.3.3 Análise dos Resultados

Inicialmente foi feita uma simulação do algoritmo para verificar seu comportamento em relação ao número de soluções viáveis geradas pelo procedimento *Permuta3* executado por cada processo escravo. A Tabela 7.2 apresenta estes resultados para uma das instâncias e a Figura 7.6 apresenta essa mesma tabela numa forma gráfica.

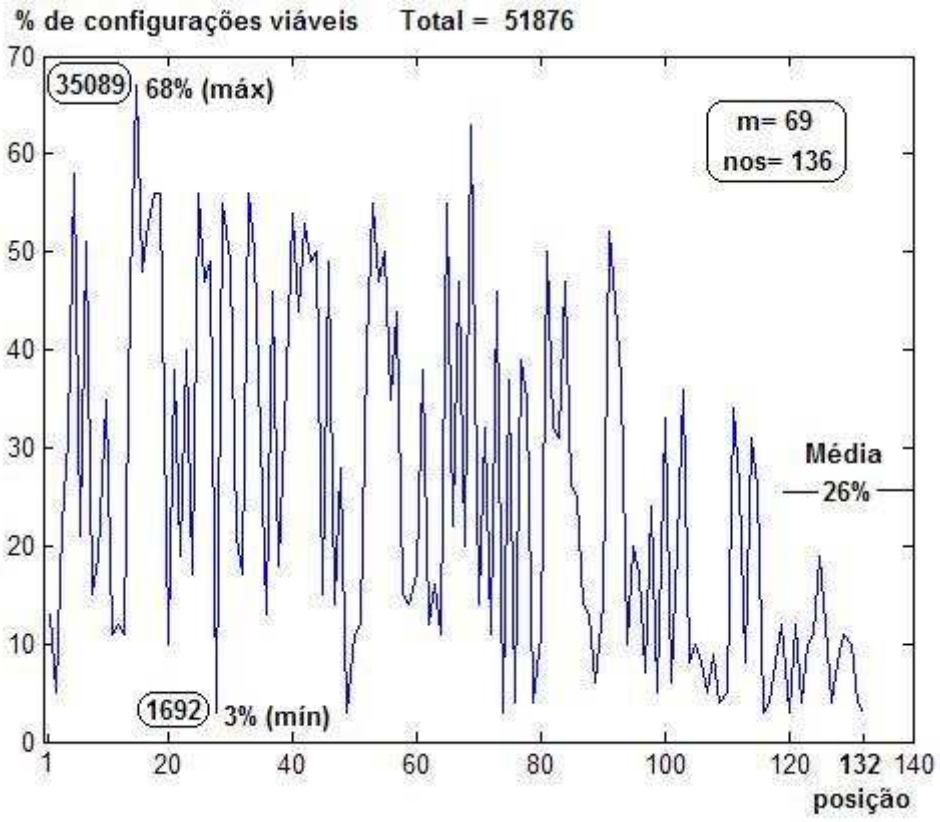


Figura 7.6 Resultado da Simulação do Algoritmo Paralelo para a instância TST15 ($m = 69$ e $n = 77$). São mostrados os percentuais de configurações viáveis para cada partida (posição i)

Na Figura 7.7 observa-se o comportamento médio para outras instâncias para esta mesma análise. Verifica-se que há um padrão nos resultados, no sentido de que à medida que há um envolvimento das posições de mais alta ordem do vetor de configurações o processo de permuta gera menos configurações viáveis.

Uma segunda análise prévia foi feita em duas instâncias para observar o efeito da alocação

Tabela 7.2 Resultado da Simulação da execução do algoritmo paralelo para a instância TST15 ($m = 69$ e $n = 77$). Em cada linha estão informados, na ordem, a identificação do processo, a posição i (pivô) das permutas realizadas, o total de configurações inviáveis, o total de viáveis e o percentual dessas em relação ao total. A tabela está mostrada parcialmente de um total de 132 processos e 60 milhões de configurações geradas, sendo 13 milhões (cerca de 22%) de configurações viáveis

Processo	Posição	Inviáveis	Viáveis	%
2167	100	320535	11118	3
2178	101	269420	11182	3
2247	61	533952	21897	3
2216	65	727235	29367	3
2220	60	684782	28742	4
2239	67	544061	34636	5
2159	102	362667	24202	6
2259	62	482996	36082	6
2231	64	594922	42799	6
2165	107	320724	27708	7
2236	63	557740	42853	7
2169	108	297804	28018	8
2227	71	604722	59060	8
2188	103	240598	24653	9
2171	112	281079	28678	9
2168	113	291230	31974	9
2222	68	646468	64081	9
2210	69	743198	80520	9
2211	70	732912	79025	9
2183	104	236000	28894	10
2192	105	225426	26616	10
2177	110	251878	28486	10
2253	66	463198	54928	10
2224	72	612016	68426	10
2148	114	432873	54789	11
2260	73	466728	59609	11
2229	74	574230	78009	11
2190	106	219998	33234	13
2207	75	747037	113452	13
2172	118	262514	50337	16
2248	76	453840	89514	16
2185	116	215528	44844	17
2160	121	311678	65790	17
2237	81	480404	98650	17
2208	83	697730	150264	17
2187	119	212988	48574	18

Processo	Posição	Inviáveis	Viáveis	%
2240	80	460460	107884	18
2180	109	223682	53707	19
2254	79	431139	102695	19
2191	111	202010	55744	21
2238	77	464298	124871	21
2234	82	483568	134161	21
2174	117	222002	70500	24
2184	120	202600	64317	24
2255	78	399210	129983	24
2249	84	423524	135776	24
2162	124	268148	94326	26
2161	127	278622	102297	26
2170	129	230396	84478	26
2155	48	305472	114717	27
2152	49	323138	124540	27
2179	115	205506	82355	28
2181	128	193165	77203	28
2156	50	284268	124973	30
2252	86	371490	171269	31
2250	85	354642	170267	32
2176	130	189903	98315	34
2153	132	286044	152233	34
2173	125	190166	106858	35
2157	53	268028	144783	35
2164	51	219887	128545	36
2175	126	182522	109147	37
2186	131	167528	100341	37
2182	122	159024	108488	40
2189	123	157242	108009	40
2166	52	203600	142809	41
2151	56	252921	214868	45
2150	57	255868	211326	45
2149	58	254328	233691	47
2158	54	188082	226157	54
2147	55	212398	285855	57
2154	59	150124	278990	65

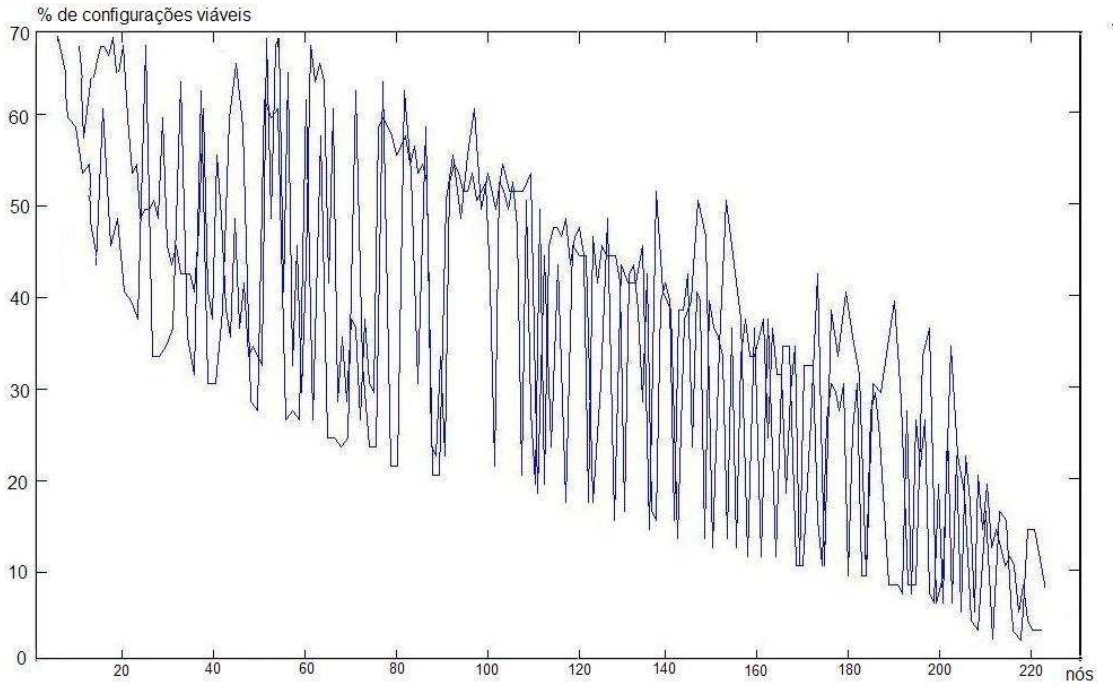


Figura 7.7 Outros Resultados da Simulação do Algoritmo Paralelo. O gráfico mostra o resultado médio para duas outras instâncias. Diferentemente da Figura 7.6 os nós estão ordenados independente de sua posição *i* no vetor de configuração

de mais processadores para participar do processo de paralelização. O que se observa pelos resultados apresentados na Figura 7.8 que a melhor performance de tempo é obtida quando se utiliza em torno de 40 processadores.

O gráfico da referida figura corresponde à uma média de três execuções para cada instância. Esta foi a estratégia utilizada como forma de amenizar a influência da carga do uso dos processadores em relação aos processos concorrentes (internos ou externos) no momento de sua alocação.

Com base nessa análise a melhor opção seria escolher como número ideal de processadores valores entre 40 e 60. Porém, como a menor instância é $m = 45$ que corresponde a 88 nós e a maior é $m = 117$ ou 232 nós, seria necessário dividir a faixa [1,nós] em duas, três ou quatro subfaixas para que todas as posições fossem processadas. Como para até 200 processadores o tempo não duplica, optamos por escolher uma única faixa utilizando simultaneamente todos os nós.

Estes valores estão informados na Tabela 7.1 onde se observa na coluna NPROC que a variação é proporcional a m e equivalente ao número (nós-3).

Analisando so resultados da Tabela 7.1 verifica-se que o tempo médio de execução dos dois algoritmos (serial e paralelo) para as 12 primeiras instâncias é de aproximadamente 21 minutos (1244 segundos) contra 3h e 50 minutos para os melhores resultados conhecidos em Ribeiro & Vianna [RV05] (ver Figura 7.9). Obtivemos uma redução média de tempo em torno de 12,8% associada a uma aproximação média da melhor solução de 0,7%. Portanto, podemos esperar que as técnicas aqui utilizadas geram boas soluções para o problema da filogenia baseada em características.

A Figura 7.10 mostra um caso de melhoria comparando as soluções dos programas serial e paralelo para a instância ETHE40.

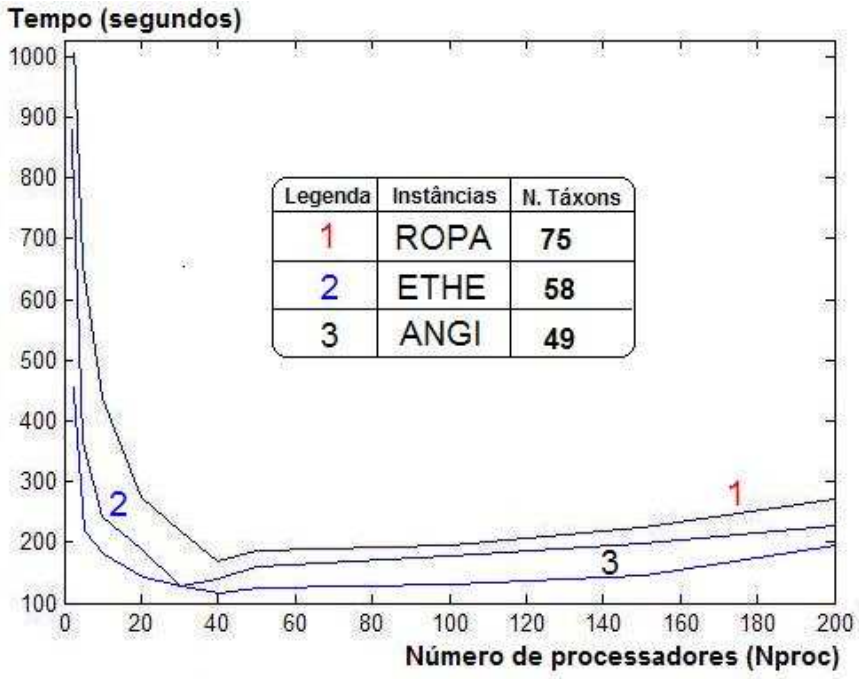


Figura 7.8 Análise do tempo de execução do algoritmo paralelo proposto em relação ao número de processadores. Foram utilizados nos três testes os seguintes valores para NPROC: 2, 5, 10, 20, 30, 40, 50, 100, 150, 200. Observa-se que os tempos para cada instância utilizando dois processadores têm início em pontos distintos

Figura 7.9 Comparação do tempo de execução do algoritmo paralelo para as instâncias comr resultados conhecidos. Inclui os tempos correspondentes da Heurística serial contidos na Figura 6.8 da página 131

```
111  76  87  61  63  63  67  69  69  70  71  71  73  72  68  66  65  64  64  79
 80  81  84  78  78  85  79  82  60  62  60  89  90  90  91  96  93  94  92  92
 89  61  88 102 101  98 103 103 104 104 100 108 107  77  59  59 112 109  99  62
 76 113  86  65  66  67  68  72  70  74  75  73  74  75  77  87  86  80  81  82
 83  83  84  85 110  88 114 111  91  97  95  93  94  95  96  97  98  99 100 101
102 106 105 105 106 107 108 109 110 112 113 115 114 115 252

((57-Phymaturus,((26-Enyaliosaurus,(23-Iguana,((22-Brachylophus,(20-Amblyrhynchus,
27-Sauromalus)79)81,(28-Dipsosaurus,(21-Conolophus,(24-Cyclura,25-Ctenosaura)78)80)
82)83)84)85,(58-Leiocephalus,(52-Stenocercus,(53-Proctotretus,((44-ETropidurus,
(45-WTropidurus,(51-Ophryoessoides,((55-Vilcunia,56-Ctenoblepharis)59,(46-Tapinurus,
((33-Sceloporus,34-Sator)90,(36-Urosaurus,((35-Uta,(32-Petrosaurus,41-Phrynosoma)89)
91,(38-Holbrookia,(37-Callisaurus,(39-Cophosaurus,40-Uma)92)93)94)95)96)97)98)99)100)
101)102,((47-Uranoscodon,48-Plica)103,(49-Uracentron,50-Strobilurus)104)105)106)107)
108)109)110)112,((03-Morunasuarus,(02-Enyalioides,(04-Hoplocercus,42-Crotaphytus)61)
76)87,((30-Laemanctus,(29-Basiliscus,31-Corytophanes)60)62,(01-Ancestor,(43-Gambelia,
((05-Oplurus,06-Chalarodon)63,(54-Liolaemus,((11-Diplolaemus,12-Pristidactylus)71,
((10-Aperopristis,(08-Enyalius,09-Leiosaurus)69)70,(13-Urostrophus,(14-Anisolepis,
(15-Aptycholaemus,(07-Polychrus,(16-Chamaeleolis,(17-Phenacosaurus,(18-Chamaelinorops,
19-Anolis)64)65)66)67)68)72)73)74)75)77)86)88)111)113)114);
```

Figura 7.10 Solução obtida pelo algoritmo paralelo para a instância ETHE40, representado pelo vetor Z, o custo desta solução (252) e a árvore no formato Newick [NEW07]

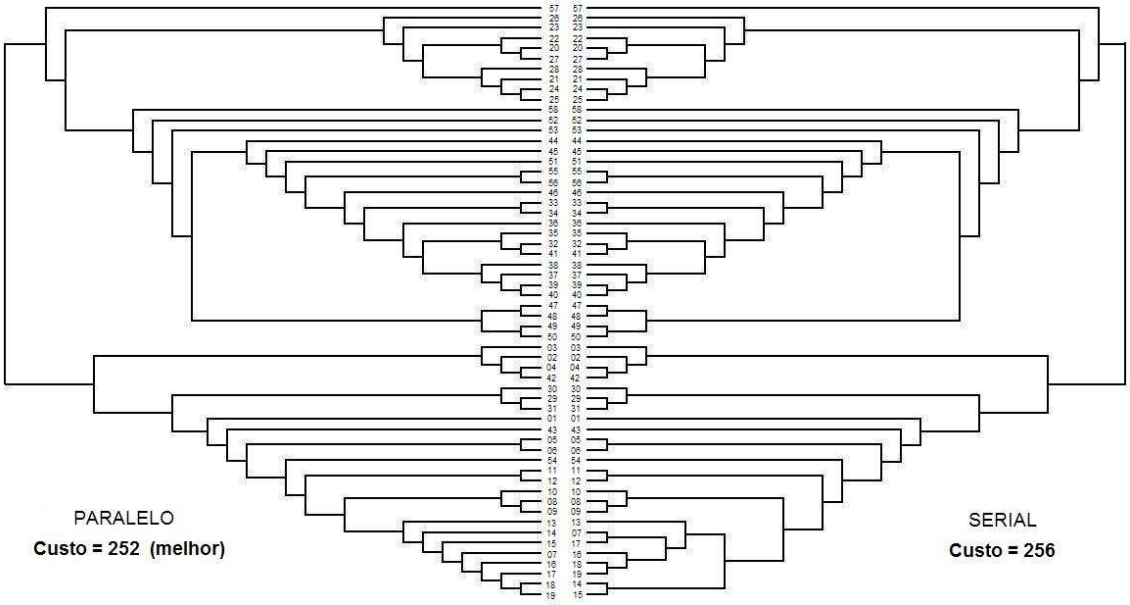


Figura 7.11 Comparação dos resultados Paralelo e Serial para a instância ETHE40. Observe que as topologias das árvores têm uma pequena diferença na parte inferior da figura. A seqüência de nós da árvore da esquerda é $< 13, 14, 15, 07, 11, 17, 18, 19 >$ e a da direita é $< 13, 07, 17, 16, 18, 19, 14, 15 >$

7.4 Considerações Finais

Observamos que um dos motivos para o algoritmo paralelo não ter conseguido uma **melhoria** da solução para algumas instâncias deve-se ao fato de que a heurística serial, usada na fase anterior, provavelmente tenha atingido um "mínimo local".

Percebemos que a simples paralelização daquela heurística não traria melhores resultados, nem uma significativa redução de tempo de execução pois sua convergência é rápida e a dificuldade encontrada na obtenção de melhores resultados, como já foi citado, deve-se ao problema da geração da população inicial e não ao esforço computacional. A limitação do número de configurações de partidas distintas prejudica o desempenho da heurística, ou seja, há necessidade de descobrir para o problema uma forma de obter uma maior diversificação das configurações.

Desta forma optamos por criar um algoritmo paralelo de melhoria iterativa, a partir das soluções fornecidas pela heurística, a fim de intensificar o processo de busca numa vizinhança variada. Vimos que com a programação paralela foi possível estender esse processo para toda a vizinhança de acordo com o critério de perturbação adotado sem comprometer o tempo total de execução dos algoritmos propostos.

CAPÍTULO 8

Conclusão e Trabalhos Futuros

Após a apresentação das definições sobre sistemática filogenética, conceitos sobre biologia computacional e informações necessárias para entender os métodos de construção das árvores filogenéticas, este trabalho deixa o leitor apto a compreender o Problema da Filogenia.

Durante a fase de concepção das heurísticas criadas neste trabalho foram testados procedimentos utilizando técnicas da otimização combinatória permutacional [VSS04]. Os resultados obtidos não foram satisfatórios pois geravam muitas configurações inviáveis. Foi observado também que é totalmente imprevisível determinar a variação da função objetivo entre vizinhos com configurações "próximas", daí optamos por utilizar múltiplas vizinhanças para obter uma maior diversificação e tentar fugir dos mínimos locais.

Uma das grandes dificuldades na resolução deste problema consiste na limitação do número de configurações iniciais variadas, ou seja, com custos distintos. Se fosse possível reproduzir uma população inicial maior, haveria uma maior probabilidade de sucesso na obtenção de melhores soluções pois o processo de busca local teria um acréscimo no total de partidas múltiplas.

Portanto, acreditamos que melhores resultados poderiam ser obtidos pelos algoritmos desenvolvidos se houvesse uma maior diversificação na geração de novas configurações iniciais. Outras recombinações, além daquelas apresentadas na Subseção 6.5.3, poderiam ser criadas e processadas de forma serial ou paralela sempre com a preocupação que não haja um grande acréscimo no tempo global de execução das rotinas.

Constatamos, através dos experimentos computacionais, que a forma de abstração dos dados escolhida para representar uma árvore filogenética, bem como a utilização de operações lógicas implementadas reduziram o tempo de execução dos programas, se comparados com operações com vetores ou o uso de outro tipo de estrutura de dados.

O uso de uma boa função de randomização, desenvolvida por Park & Miller [PM88], e do controle de "exploração" das regiões dentro do espaço de busca com certeza contribuíiram para

a qualidade das soluções obtidas.

Esperamos que esta forma de abordagem e tratamento na resolução deste problema da filogenia desperte interesse aos pesquisadores na área de bioinformática e que novas técnicas possam ser criadas a partir da idéia concebida neste trabalho.

Para **Trabalhos Futuros**, com base no assunto desta Tese, poderíamos propor:

- Criar novas rotinas de recombinação para esta mesma abordagem do problema;
- De modo similar aos procedimentos especificamente desenvolvidos para a estrutura de dados aqui utilizada (vetores), poderia ser realizado um estudo configurando as árvores filogenéticas na forma de *strings* de acordo com o formato padrão Newick;
- Utilizar de forma mais robusta a paralelização. Uma idéia seria cada processo realizar modificações numa determinada região (subárvore) do vetor de configuração e através do compartilhamento de dados o processo mestre faria o gerenciamento de busca de forma cooperativa e
- Tratar o problema mais complexo, levando em consideração matrizes com características múltiplas, ou seja, “não binárias”.

# # #

Referências Bibliográficas

[ABJ+99] Bruce Alberts, Dennis Bray, Alexander Johnson, Julian Lewis, Martin Raff, Keith Roberts, and Peter Walter. *Fundamentos da Biologia Celular: Uma Introdução à Biologia Molecular da Célula.* Tradução: Carlos Termignoni et al., ArtMed Editora, Artes Médicas Sul, Porto Alegre, 1999.

[Adi00] Said Sadique Adi. Ferramentas de auxílio ao seqüenciamento de DNA por montagem de fragmentos: um estudo comparativo. Dissertação de mestrado, Instituto de Matemática e Estatística da Universidade de São Paulo – IME/USP, São Paulo, Março 2000. Orientado por: Prof. Dr. Carlos Eduardo Ferreira.

[AFB93] Richa Agarwala and David Fernandez-Baca. A polynomial-time algorithm for the perfect phylogeny problem when the number of character states is fixed. In *IEEE Symposium on Foundations of Computer Science*, pages 140–147, 1993.

[AM04] José M. Amabis and Gilberto R. Martho. *Biologia dos Organismos*, volume 2. Editora Moderna, São Paulo, 2. edição, 2004.

[Amo97] Dalton S. Amorim. *Elementos Básicos de Sistemática Filogenética.* Holos Editora, Ribeirão Preto, SP, 2. edição, 1997.

[AMS+97] S. F. Altschul, Thomas Madden, Alejandro A. Schäffer, Jinghui Zhang, Zheng Zhang, Webb Miller, and David J. Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25(17):3389–3402, 1997.

[AR02] Alexandre A. Andreatta and Celso C. Ribeiro. Heuristic for the phylogeny problem. *Journal of Heuristics*, 8:429–447, 2002. http://dblp.uni-trier.de.

[Bar94] Blaise Barney. *Introduction to PVM*. Ithaca, Cornell Theory Center, New York, USA, 1994.

[BFW93] Hans L. Bodlaender, Michael R. Fellows, and Tandy J. Warnow. Two strikes against perfect phylogeny. In *Proceedings of the 19th International Colloquium on Automata, Languages and Programming*, volume 623 of *Lecture Notes in Computer Science*, pages 273–283, Berlin:Springer-Verlag, 1993.

[BGP05] David Bryant, Nicolas Galtier, and Marie-Anne Poursat. Likelihood calculation in molecular phylogenetics. In *Mathematics of Evolution & Phylogeny*, chapter 2, pages 33–62. Olivier Gascuel, editor, Oxford University Press Inc, New York, 2005.

[BLA06] BLAST. http://ncbi.nih.gov/blast, acessado em: Jul/2006, 2006.

[Ble96] G. E. Blellock. Programming parallel algorithms. *Communications of the ACM*, 39(3), Mar, 1996.

[BLP97] V. Bafna, E. L. Lawler, and P. A. Pevzner. Approximation algorithm for multiple sequence alignment. *Theorical Computer System*, 182:233–244, 1997.

[Boa03] Paulo O. Boaventura Netto. *Grafos: Teoria, Modelos, Algoritmos*. Edgard Blücher, São Paulo, 3. edição, 2003.

[BV01] P. Bonizzoni and G. D. Vedova. The complexity of multiple sequence alignment with SP-score that is metric. *Theorical Computer System*, 259:63–79, 2001.

[Cha98] M. A. Charleston. Spectrum: Spectral analysis of phylogenetic data. *Bioinformatics*, 14(11):98–99, 1998.

[CJC00] J. Choi, H. Jung, and H. Cho. Phylodraw: A phylogenetic tree drawing system. *Bioinformatics*, 16(11):1056–1058, 2000.

[CLRS02] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press/McGraw-Hill, Cambridge, MA/New York:, 2. edition, 2002.

[CLU06] CLUSTALW. http://www.ebi.ac.uk/clustalw, acessado em: Jul/2006, 2006.

[Coo71] Stephen A. Cook. The complexity of theorem proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, pages 151–158, New York, 1971.

[CRM99] Neil A. Campbell, Jane B. Reece, and Lawrence G. Mitchell. *Biology*. Benjamim/Cummings, Menlo Park, California, 5. edition, 1999.

[Dar98] Charles Darwin. *The Origin of Species*. Oxford University Press, 1998. Edited with an Introduction and Notes by Gillian Beer.

[DDB06] DDBJ. http://www.ddbj.nig.ac.jp, acessado em: Jul/2006, 2006.

[DG05] Richard Desper and Olivier Gascuel. The minimum evolution distance-based approach to phylogenetic inference. In *Mathematics of Evolution & Phylogeny*, chapter 1, pages 1–32. Olivier Gascuel, editor, Oxford University Press Inc, New York, 2005.

[DJS86] William H. E. Day, David S. Johnson, and David Sankoff. The computational complexity of inferring rooted phylogenies by parsimony. *Mathematics Biosciences*, 81:33–42, 1986.

[DK87] A. Dress and M. Krüger. Parsimonious phylogenetic tree in metric spaces and simulated annealing. *Advanced in Applied Mathematics*, 8:8–37, 1987.

[DS86] William H. E. Day and David Sankoff. Computational complexity of inferring phylogenies by compatibility. *Systematics Zoology*, 35(2):224–229, 1986.

[EBI06] EBI. http://www.ebi.ac.uk, acessado em: Jul/2006, 2006.

[EMB06] EMBL. http://www.ebi.ac.uk, acessado em: Jul/2006, 2006.

[Far89] James S. Farris. The retention index and the rescaled consistency index. *Cladistics*, 5:417–419, 1989.

[FAS06] FASTA. http://www.ebi.ac.uk/fasta, acessado em: Jul/2006, 2006.

[FD87] D. Feng and R. F. Doolittle. Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *Journal of Molecular Evolution*, 25:351–360, 1987.

[Fea04] Nicholas Fearn. *Aprendendo a Filosofar em 25 lições: do poço de Tales à desconstrução de Derrida.* Original: Zeno and the Tortoise (How to think like a Philosopher); publicado em 2001 por Atlantic Books em Londres; tradução de Maria Luiza X. de A. Borges. Jorge Zahar Editora, Rio de Janeiro, 2004.

[Fel93] Joe Felsenstein. PHYLIP - *Phylogeny Inference Package: Computer Program for Inferring Phylogenies, version 6.32.* University of Washington, Seatle, WA, 1993. http://evolution.genetics.washington.edu/phylip.html, last time accessed: May/2006.

[FG82] L. R. Foulds and R. L. Graham. The Steiner problem in phylogeny is *NP* - Complete. *Advances in Applied Mathematics*, 3:43–49, 1982.

[Fit84] Walter M. Fitch. Cladistic and other methods: Problems, pitfalls and potentials. In T. Duncan and T. F. Stuessy, editors, *Cladistics: Perspectives on the Reconstruction of Evolutionary History*, volume 12, pages 221–252. Columbia University Press, 1984.

[Fut03] Douglas Futuyama. *Biologia Evolutiva.* Funpec Editora, Ribeirão Preto, SP, 2. edição, 2003.

[GBD94] A. Geist, A. Benguelim, and John Dongarra. PVM3*, User Guide and Reference Manual.* Oak Ridge National Laboratory, Tennessee, USA, 1994.

[Gen06] GenBank. http://www.ncbi.nlm.nih.gov, acessado em: Jul/2006, 2006.

[GGLD05] Olivier Gascuel, Stéphanie Guindon, Franckm Lethiec, and Patrice Duroux. PHYML *Online - A Web Server for Fast Maximum Likelihood-based Phylogenetic Inference.* LIRMM - Laboratoire d'Informatique, de Robotique et de Mi-

croélectronique de Montpellier, 2005. https://lists.lirmm.fr/wws/info/phyml-info, last time accessed: May/2006.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, Bell Telephone Laboratories, New York, 1979.

[GKL95] Fred Glover, James P. Kelly, and Manuel Laguna. Genetic algorithms and tabu search: hybrids for optimization. *Computers and Operations Research*, 22(1):111 – 134, January 1995.

[GL93] Fred Glover and Manuel Laguna. Tabu search. In C. Reeves, editor, *Modern Heuristic Techniques for Combinatorial Problems*, Oxford, England, 1993. Blackwell Scientific Publishing.

[GNR01] J. Gramm, R. Niedermeier, and P. Rossmanith. Exact solutions for closest string and related problems. In Springer Verlag, editor, *Proceedings of the 12th Annual International Symposium on Algorithms and Computation (ISAAC 2001)*, volume 2223, pages 441–452, 2001.

[Gol96] Pablo A. Goloboff. Methods for faster parsimony analysis. *Cladistics*, 12:199–220, 1996.

[Gre06] P. Green. Documentation for Phrap and Cross_match. http://www.phrap.org/phredphrap/phrap.html, last accessed: Ago/2006, 2006.

[Gus97] D. Gusfield. *Algorithms on strings, trees and sequences*. Cambridge University Press,Forthcoming. Computer Science and Computational Biology, 1997.

[Hol75] John H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. University of Michigan Press, Ann Arbor, MI, 1975.

[IBM94] International Business Machine IBM. *Optimization and Tuning Guide for Fortran, C and C++*. Laboratory Information Development, Canada, 1994.

[JC69] T. H. Jukes and R. C. Cantor. Evolution of protein molecules. In H. N. Munro, editor, *Mammalian Protein Metabolism*, chapter 2, pages 21–132. Academic Press, New York, 1969.

[JN90] L. Jin and M. Nei. Limitations of the evolutionary parsimony method of phylogenetic analysis. *Molecular Biology and Evolutions*, 7:82–102, 1990.

[Jon94] Jeff A. Jones. Parallelizing the phylogeny problem. Doctoral thesis, University of California, Berkley, California, Dec, 1994.

[KFHW98] I. J. Kitching, P. L. Forey, C. J. Humphries, and D. M. Williams. *Cladistics: The theory and pratice of parsimony analysis*. Oxford University Press, Londres, 1998.

[KGKG03] Vipin A. Kumar, Anna Grama, George Karypis, and Ansshul Gupta. *Introduction to Parallel Computing: Design and Analysis of Algorithm*. Pearson Education, Addison-Wesley, 2 edition, 2003.

[KGV83] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[Kim80] M. Kimura. A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences. *Molecular Evolutions*, 16:111–120, 1980.

[KNT+05] T. M. Keane, T. J. Naughton, S. A. A. Travers, J. O. McLnerney, and G. P. McCormack. DPRml: Distributed phylogeny reconstruction by maximum likelihood. *Bioinformatics*, 21(7):969–974, 2005.

[KTN04] S. Kumar, K. Tamura, and M. Nei. MEGA3: Integrated software for molecular evolutionary genetics analysis and sequence alignment. *Bioinformatics*, 5(2):150–163, 2004.

[KW92] S. K. Kannan and T. J. Warrow. Triangulating 3-colored graph. *SIAM Journal on Discrete Mathematics*, 5(2):249–258, 1992.

[LMW02] M. Li, B. Ma, and L. Wang. On the closest string and substring problems. *Journal of the ACM*, 49(2):157–171, 2002.

[MLOP04] Cláudio N. Meneses, Zhaosong Lu, Carlos A. S. Oliveira, and Panos M. Pardalos. Optimal solutions for the closest string problem via integer programming. *INFORMS Journal on Computing*, 16(4):419–429, 2004.

[Mou01] David W. Mount. *Bioinformatics: Sequence and Genome Analysis.* Cold Spring Harbor Laboratory Press, New York, 2001.

[MS95] João Meidanis and João C. Setubal. Multiple alignment of biological sequences with gap flexibility. In *Proceedings of the Latin American Theorical Informatics*, volume 911, pages 411–426, Berlin, 1995. Springer-Verlag. Lecture Notes in Computer Science.

[Mul06] Multalin. http://prodes.toulouse.inra.fr/multalin/multalin.html, acessado em: Jul/2006, 2006.

[MUM06] MUMmer. http://mummer.sourceforge.net, acessado em: Jul/2006, 2006.

[NCB06] NCBI. http://www.ncbi.nlm.nih.gov, acessado em: Jul/2006, 2006.

[NEW07] NEWICK. *Newick tree format.* Department of Genome Sciences - University of Washington, Seatle, WA, 2007. available in http://evolution.genetics.washington.edu/phylip/newicktree.html, last time accessed: Feb/2007.

[Pag01] Roderic D. M. Page. *TreeView for Win32, version 1.6.6.* University of Glasgow, Scotland, UK, 2001. http://taxonomy.zoology.gla.ac.uk/rod/rod.html, last time accessed: May/2006.

[PDB06] PDB. http://www.rcsb.org/pdb, acessado em: Jul/2006, 2006.

[PHM99] F. Harvey Pough, John B. Heiser, and William N. McFarland. *A Vida dos Vertebrados.* Atheneu Editora, São Paulo, 4. edição, 1999.

[PHY06] PHYLIP. Tutorial: Phylogenetic analysis using parsimony. http://home.cc.umanitoba.ca/ psgendb/GDE/phylogeny/parsimony/phylip.parsimony.html, , last accessed: Sep/2006, 2006.

[PM88] Stephen K. Park and Keith W. Miller. Random number generators: Good ones are hard to find. *Communications of the ACM*, 31(10):1192–1201, Mar 1988.

[Pon90] Caroline C. Pond. *Diversity of Organisms*. Hodder & Stoughton/The Open University, London, 1990.

[PR01] Leônidas S. Pitsoulis and Maurício G. C. Resende. Greedy Randomized Adaptive Search Procedures. In Panos M.Pardalos and Maurício G. C. Resende, editors, *Handbook of Applied Optimization*, pages 168–181. Oxford University Press, 2001.

[PSOH03] William K. Purves, David Sadava, Gordon H. Orians, and Craig Heller. *Life: The Science of Biology*. W.H. Freeman & Co., São Paulo, 7. edition, 2003.

[QM06] José Augusto A. Quitzau and João Meidanis. A fully resolved consensus between fully resolved phylogenetic trees. *GMR-Genetic and Molecular Research*, 5(1):269–283, 2006.

[RLB00] P. Rice, I. Longden, and A. Bleasby. Emboss: The european molecular biology open software suite. *Trends in Genetics*, 6(16):276–277, 2000. http://emboss.sourceforge.net, last accessed: Jul/2006.

[Rod03] Estela Maria Rodrigues. Algoritmos para comparação de Árvores filogenéticas e o problema dos pontos de recombinação. Tese de doutorado, Instituto de Matemática e Estatística da Universidade de São Paulo – IME/USP, São Paulo, 2003. Orientado por: Profa. Dra. Yoshiko Wakabayashi.

[RV05] Celso C. Ribeiro and Dalessandro S. Vianna. A GRASP/VND heuristic for the phylogeny problem using a neighborhood structure. *International Transaction in Operational Research*, 12(3):325+, 2005.

[SDB00] Mike Steel, Andreas W. M. Dress, and Sebastian Böcker. Point of view simple but fundamental limitations on supertree and consensus tree methods. Publisher: Taylor & Francis. *Systematic Biology*, 49(2):363–368, 2000.

[SH03] Stephen C. Stearns and Rolf F. Hoekstra. *Evolução: Uma Introdução*. Traduzido do original: Evolution: an Introduction. Atheneu Editora, São Paulo, 2003.

[SHB+01] Craig A. Stewart, David Hart, Donald K. Berry, Gary J. Olsen, Eric A. Wernert, and William Fischer. Parallel implementation and performance of fastDNAml - a program for maximum likelihood phylogenetic inference. In *Proceedings of Supercomputing Conference - SC'01*, pages 32+, ACM/IEEE. Denver, 2001.

[SLMW02] Alexandros P. Stamatakis, Thomas Ludwig, Harold Meier, and Marty J. Wolf. Accelerating parallel maximum likelihood-based phylogenetic tree calculations using subtree equality vectors. In *Proceedings of Supercomputing Conference - SC'02*, ACM/IEEE. Baltimore, 2002.

[SM87] D. L. Swofford and W. P. Maddison. Reconstucting ancestral character states under Wagner parsimony. *Mathematics Biosciences*, 87:199–229, 1987.

[SM94] Jayme L. Szwarcfiter and L. Markenson. *Estrutura de Dados e seus Algoritmos*. LTC Editora, Rio de Janeiro, 1994.

[SM97] João C. Setubal and João Meidanis. *Introduction to Computational Molecular Biology*. Brooks/Cole Publishing Co., Pacific Grove, 1997.

[SN87] N. Saitou and M. Nei. The neighbor-joining method: A new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution*, 4(4):406–425, 1987.

[SO90] D. L. Swofford and G. L. Olsen. Phylogeny reconstruction. In *Molecular Systematics*, Sinauer Associates, pages 411–501, Sunderland, Massachusetts, 1990.

[SOWH96] D. L. Swofford, G. L. Olsen, P. J. Waddell, and D. M. Hillis. Phylogenetic inference. In C. Moritz D. M. Hillis and B. K. Mable, editors, *Molecular Systematics*, Sinauer Associates, pages 411–501, Sunderland, Massachusetts, 1996.

[Sta05] Alexandros P. Stamatakis. An efficient program for phylogenetic inference using simulated annealing. In *Proceedings of 19th IEEE International - Parallel and Distributed Processing Symposion*, Denver, CO, USA, 2005.

[Ste92] Michael A. Steel. The complexity of reconstructing trees from qualitative characters and subtrees. *Journal of Classification*, 9:91–116, 1992.

[Ste00] Stephen C. Stearns. *Evolution: an Introduction*. Oxford University Press, England, 2000.

[Sut95] Granger Sutton. *Tigr Assembler: Program for shotgun fragment assembly*. The Institute for Genomic Research, 1995. http://www.tigr.org/Software/assembler, last accessed: Ago/2006.

[SW81] T. F. Smith and M. S. Waterman. Identification of common molecular subsequence. *Journal of Molecular Biology*, 1(147):195–197, 1981.

[SWB+99] Craig A. Stewart, Tan Tin Wee, Markus Buchhorn, David Hart, Donald K. Berry, Louxin Zhang, Meena Sakharkar, Eric A. Wernert, William Fischer, and Donald F. McMullen. Evolutionary biology and high performance computing. In R. F. Enenkel, editor, *Software Tools for Computational Biology*, 1999. www.indiana.edu/rac/staff_papers.html.

[Tam92] K. Tamura. Estimation of the number of nucleotide substitutions when there are strong transition-transversion and g + c-content biases. *Molecular Biology and Evolutions*, 9:678–687, 1992.

[TN84] F. Tajima and M. Nei. Estimation of evolutionary distance between nucleotide sequences. *Molecular Biology and Evolutions*, 1(3):269–285, 1984.

[VBB+06] Gerardo Valdisio R. Viana, Ana Luiza B. P. Barros, Tariana M. F. Batista, Gustavo A. L. Campos, Raimundo B. Costa, Rodrigo Maggioni, and Diana M. Oliveira. Phylotree - an integrated and automatic tool to generate phylogenetic trees. In *Proceedings ISMB 2006 14th Annual International Conference on Intelligente Systems for Molecular Biology*, Fortaleza-Ceará, Brazil, August 6-10, 2006. Poster C-29 <http://ismb2006.cbi.cnptia.embrapa.br/posters_list.php>.

[VGFM07] Gerardo Valdisio R. Viana, Fernando C. Gomes, Carlos E. Ferreira, and Cláudio N. Meneses. Uma implementação eficiente de uma heurística de busca local em multi-vizinhanças para um problema de filogenia. *Submetido ao XXXIX Simpósio Brasileiro de Pesquisa Operacional - SBPO*, 2007. (Paper # 28972, em 30/03/2007).

[VGM+06] Gerardo Valdisio R. Viana, Fernando C. Gomes, Hélio A. S. Moura, Cláudio N. Meneses, and Panos M. Pardalos. Phylodraweb: Phylogenetic trees drawing web service. www.lia.ufc.br/ valdisio/PhyloDraWeb.pdf, Jul 2006.

[VGMP] Gerardo Valdisio R. Viana, Fernando C. Gomes, Cláudio N. Meneses, and Panos M. Pardalos. A parallel multistart algorithm for the closest string selection. *Computing and Operation Research*, Available online, 19 April. DOI link, http://dx.doi.org/10.1016/j.cor.2007.04.002.

[VGMP04] Gerardo Valdisio R. Viana, Fernando C. Gomes, Cláudio N. Meneses, and Panos M. Pardalos. Parallel algorithm for the closest string selection. In *Proceedings BIOMAT 2004 IV Brazilian Symposium on Mathematical and Computational Biology / I International Symposium on Mathematical and Computational Biology*, Ilhéus-BA, Brazil, 2004. www.biomat.org/biomat4/ilheus.html.

[VGMP06] Gerardo Valdisio R. Viana, Fernando C. Gomes, Cláudio N. Meneses, and Panos M. Pardalos. Experimental analysis of approximation algorithms for the vertex cover and set covering problems. *Computing and Operation Research*, 33(12):3520–3534, Dec 2006.

[Via98] Gerardo Valdisio R. Viana. *Meta-heurísticas e Programação Paralela em Otimização Combinatória.* Edições UFC, Fortaleza, 1998.

[vLA87] Peter J. M. van Laarhoven and Emile H. L. Aarts. *Simulated Annelaing: Theory and Applications*. Kluwer Academic Publishers, Norwell, MA, USA, 1987.

[VSS04] Gerardo Valdisio R. Viana, José Lassance C. Silva, and Nei Yoshihiro Soma. Um algoritmo genético híbrido construtivo para problemas de otimização combinatória permutacional. *Congresso Latino-Iberoamericano de Investigatión de Operaciones y Sistemas*, 2004. XII CLAIO.

[VTS04] Gerardo Valdisio R. Viana, Antônio Clécio F. Thomaz, and José Lassance C. Silva. Um algoritmo aproximativo aplicado ao "Set Partitioning Problem". *Simpósio Brasileiro de Pesquisa Operacional Livro de Resumos*, 1:105–105, Nov 2004. XXXVI SBPO.

[Wik06a] Wikipédia. *Aminoácidos, Nucleotídeos, Cladística & Taxonomia.* Wikimedia Foundation, 2006. http://pt.wikipedia.org/wiki, acessado em: Mai/2006.

[Wik06b] Wikipedia. *Bioinformatics.* Wikimedia Foundation, 2006. http://en.wikipedia.org/wiki, acessado em: Jun/2006.

[Wik06c] Wikipedia. *Bioinformatics/FASTA*. Wikimedia Foundation, 2006. http://en.wikipedia.org/wiki/FASTA, acessado em: Jul/2006.

[Wik06d] Wikipedia. *Phylogenetic Tree of Life.* Wikimedia Foundation, 2006. http://en.wikipedia.org/wiki/Phylogenetic_Tree, last accessed: Set/2006.

[WJ94] L. Wang and T. Jiang. On the complexity of multiple sequence alignment. *Journal of Computational Biology*, 1:337–348, 1994.

[WJL96] Lusheng Wang, Tao Jiang, and E. L. Lawler. Approximation algorithm for tree alignment with a given phylogeny. *Algorithmica*, 16(3):302–315, 1996.

APÊNDICE A

Abreviaturas e Siglas

AVS	Alinhamento de Várias Seqüências
BD	Banco de Dados
BLAST	Basic Local Alignment Search Tool
CPU	Central Processing Unit
DDBJ	DNA Data Bank of Japan
DFH	Drosophila homolog of the Friedreich's ataxia disease gene
DNA	DeoxyriboNucleic Acid
EBI	European Bioinformatics Institute
EMBC	European Molecular Biology Conference
EMBL	European Molecular Biology Laboratory
EMBOSS	The European Molecular Biology Open Software Suite
FASTA	Fast Alignment
GENBANK	Genetic Sequence Database
GRASP	Greedy Randomized Adaptive Search Procedures
HIV	Human Immunodeficiency Virus
IBM	International Business Machine
ICZN	International Code of Zoological Nomenclature
ICBN	International Code of Botanical Nomenclature
ICNB	International Code of Nomenclature of Bacteria
ICTV	International Committee on Tanonomy of Viruses
IP	Internet Protocol
JPG	Joint Photographic experts Group (Jpeg)
LACAD	Laboratório de Computação de Alto Desempenho
LIA	Laboratórios de Pesquisa em Ciência da Computação
NCBI	National Center for the Biotechnology Information

NIG	National Institute of Genetics
NIH	National Institute of Health
NP	Non-deterministic Polynomial
NPC	NP-Complete, referem-se aos problemas mais difíceis da classe de problemas NP
PDB	Protein Data Bank
PFC	Problema da Filogenia baseada em Características
PFD	Problema da Filogenia baseada em Distâncias
PFP	Problema da Filogenia Perfeita
PNG	Portable Network Graphics
PVM	Parallel Virtual Machine
PSMP	Problema da Seqüência Mais Próxima
RAM	Ramdom Access Memory
RNA	RiboNucleic Acid
s.p.g.	sem perda de generalidade
SCSI	Small Computer System Interface
SVG	Scalable Vector Graphics
UPGMA	Unweighted Pair-Group Method using Arithmetic average
WPGMA	Weighted Pair-group Method using Arithmetic average

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo