Algoritmos evolutivos para problemas de projeto de redes aplicados à filogenia

Karen Honda

Orientador: *Prof. Dr. Alexandre Cláudio Botazzo Delbem*

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação - ICMC-USP, como parte dos requisitos para obtenção do título de Mestre em Ciências de Computação e Matemática Computacional.

“VERSÃO REVISADA APÓS A DEFESA”

1504623
Data da Defesa: 21/11/2005

Visto do Orientador:

Delbem

USP – São Carlos
Janeiro/2006

Agradecimentos

Agradeço a Deus por me dar forças para concluir mais uma etapa em minha vida.

À minha família pelo carinho e compreensão, especialmente à minha mãe que não mede esforços para conseguir o melhor para seus filhos.

Ao Prof. Dr. Alexandre Cláudio Botazzo Delbem. Serei sempre grata à sua orientação e troca de idéias. Suas críticas sempre construtivas, seu incentivo e paciência foram fundamentais para a realização deste trabalho.

Aos meus amigos, em especial à Ana Carla, Chris, Eliane e Valéria pela amizade, apoio e convivência, tornando o período do mestrado mais agradável.

Por fim, agradeço ao CNPq pelo auxílio financeiro.

Resumo

Um dos principais problemas da Biologia é tentar explicar o processo evolutivo das espécies existentes e de que forma essas espécies se relacionam em termos de ancestrais comuns. À determinação dessas relações evolutivas dá-se o nome de filogenia ou reconstrução de árvores filogenéticas. A reconstrução de árvores filogenéticas têm sido importante para uma variedade de problemas, tais como: taxonomia, virologia, filogenômica, alinhamento múltiplo de seqüências, entre outras.

Um problema fundamental em filogenia consiste no fato das espécies ancestrais que existiram no passado não poderem ser observadas diretamente. Assim, é necessário buscar mecanismos para, analisando os organismos atuais, recuperar informações a respeito das relações de parentesco com os organismos ancestrais hipotéticos.

Neste sentido, as técnicas filogenéticas buscam determinar os ancestrais hipotéticos que melhor representam um processo evolutivo que explique as espécies existentes. Os Algoritmos Evolutivos (AEs) têm mostrado resultados significativos em filogenia. Por outro lado, a reconstrução de árvores filogenéticas é um problema de Projeto de Redes (PR) para o qual novas abordagens evolutivas têm sido desenvolvidas recentemente buscando o aumento de eficiência computacional.

Este trabalho investiga a aplicação dessas novas abordagens para filogenia.

Abstract

One of the most important problems in Biology is to comprehend the evolutionary process of existing species and determine how they are related with their common ancestors. The determination of these evolutionary relationships is named phylogeny or phylogenetic tree reconstruction. The reconstruction of phylogenetic trees have shown to be important for a variety of problems, such as: taxonomy, virology, phylogenomic, multiple sequences alignment, among others.

One fundamental problem in phylogeny is that ancestral species cannot be directly observed. In order to overcome this problem, search mechanisms have been employed to reconstruct the relationships among these organisms and their hypothetical ancestors.

Therefore, the phylogenetic techniques search for hypothetical ancestors that best describe an evolutionary process which must explain the today species. Evolutionary Algorithms have shown relevant results in phylogeny. On the other hand, the phylogenetic tree reconstruction is a network design problem for which new evolutionary algorithms with special encoding have been developed in order to improve their efficiency.

This work investigates the application of these new approaches to phylogeny.

Conteúdo

Agradecimentos	i
Resumo	iii
Abstract	v
Lista de Figuras	xii
Lista de Tabelas	xiii
1 Introdução	1
2 Evolução e Filogenia	5
2.1 Lamarekismo	5
2.2 De Darwin à Teoria Sintética da Evolução	7
2.3 Fundamentos da Genética	10
2.3.1 DNA, RNA e Cromossomos	10
2.3.2 Replicação do DNA	11
2.3.3 Mutação	11
2.3.4 Recombinação Gênica	12
2.4 Reconstrução Filogenética	12
2.4.1 Árvores Filogenéticas	13
2.5 Considerações Finais	15
3 Métodos Computacionais para Filogenia	17
3.1 Tipos de Dados: Características × Distância	17

3.2	Métodos de Reconstrução: Clustering × Critério de Ótimo	18
3.3	UPGMA	19
3.3.1	Algoritmo	19
3.4	Neighbor Joining	20
3.4.1	Algoritmo	21
3.5	Máxima Parcimônia	24
3.5.1	Pequeno Problema da Parcimônia	24
3.5.2	Grande Problema da Parcimônia	25
3.6	Máxima Verossimilhança	26
3.6.1	Modelos Evolutivos	27
3.6.2	Cálculo da Verossimilhança de uma Árvore	28
3.7	Árvores de consenso	29
3.8	Teste de confiança	30
3.8.1	Bootstrap	30
3.9	Algoritmos de Busca para Filogenia	30
3.9.1	Busca Exata	30
3.9.2	Busca Heurística	31
3.10	Considerações Finais	33
4	Computação Evolutiva	37
4.1	Os Algoritmos Evolutivos	37
4.2	Algoritmos Genéticos	40
4.2.1	Codificação do Problema	40
4.2.2	Definição da População Inicial	41
4.2.3	Operadores de Reprodução	41
4.2.4	Seleção	42
4.2.5	Função de Fitness	45
4.2.6	A Escolha de Parâmetros	45
4.3	Exemplos de Aplicações de Algoritmos Evolutivos	46
4.3.1	Árvore Geradora Mínima com Restrição de Grau	47
4.3.2	Algoritmos Evolutivos Aplicados à Filogenia	48
4.4	Considerações Finais	57

5	Codificações de Grafos para Algoritmos Evolutivos	61
5.1	Codificações de Árvores	61
5.1.1	Vetor de Características	62
5.1.2	Predecessores	63
5.1.3	Número de Prüfer	64
5.1.4	Chaves Aleatórias de Redes	66
5.1.5	Conjunto de Arestas	68
5.1.6	Nó-profundidade	70
5.1.7	Precedentes Diretos	74
5.2	Considerações Finais	77
6	Proposta	79
7	Resultados	83
8	Considerações Finais	93
8.1	Contribuições	95
8.2	Perspectivas futuras	95
	Bibliografia	97
	Apêndice	105

Lista de Figuras

1.1	Árvore filogenética para um grupo de primatas [61].	2
2.1	Estrutura de <i>DNA</i>	11
2.2	Representação de uma árvore filogenética.	13
2.3	Árvores enraizada e não enraizada.	14
2.4	Ilustração de uma árvore e uma rede filogenéticas [48].	15
3.1	Ilustração do funcionamento do método <i>UPGMA</i> [16].	20
3.2	Árvore não enraizada bifurcada.	21
3.3	Ilustração da primeira iteração do algoritmo <i>Neighbor Joining</i>	22
3.4	Ilustração passo a passo da execução do algoritmo <i>Neighbor Joining</i> para a matriz da Tabela 3.1.	23
3.5	Ilustração do algoritmo de Fitch para três seqüências.	25
3.6	Topologias ilustrando o problema da atração de ramos longos.	27
3.7	Ilustração das <i>OTUs</i> e dos possíveis ancestrais.	29
3.8	<i>Branch and bound</i>	31
3.9	<i>Stepwise addition</i>	32
3.10	<i>Star decomposition</i>	33
3.11	<i>Nearest Neighbor Interchange</i>	34
3.12	<i>Subtree Pruning and Regrafting</i>	34
3.13	<i>Tree Bisection and Reconnection</i>	35
4.1	Ilustração do funcionamento do operador de <i>crossover</i> de um-ponto.	42
4.2	Ilustração do funcionamento do operador de mutação.	43

4.3	Ilustração do método da roleta.	44
4.4	Árvore com 9 nós e sua matriz de adjacências.	51
4.5	Aplicação do operador de <i>crossover</i> utilizado em <i>GA-mt</i>	52
4.6	Decodificação para uma sequência de 4 <i>OTUs</i>	54
4.7	Ilustração de <i>consensus pruning</i>	55
4.8	Operador de <i>crossover</i> implementado para o programa <i>GaPhyl</i>	58
5.1	Árvore com 5 nós.	63
5.2	Grafo orientado com 5 nós, onde 1 representa o nó raiz.	64
5.3	Grafo com 6 nós [50].	65
5.4	Grafo com 5 nós.	68
5.5	Ilustração do operador de mutação por Conjunto de Arestas.	69
5.6	Ilustração do funcionamento do operador de recombinação por Conjunto de Arestas.	72
5.7	Codificação nó-profundidade.	73
5.8	Árvore com 5 nós.	76
5.9	Representação por precedentes diretos para a árvore da Figura 5.8.	76
6.1	Exemplos de situações em que não foram possíveis gerar árvores. Os nós em negrito representam ancestrais hipotéticos, os demais dizem respeito aos nós folhas.	81
7.1	Desempenho da aptidão ao longo de 100 gerações para 55 seqüências utili- zando inicialização aleatória.	87
7.2	Desempenho da aptidão ao longo de 100 gerações para 55 seqüências utili- zando <i>NJ</i> como <i>seeding</i>	88
7.3	Tempo para 10, 20, 30, 40, 50 e 55 seqüências de tamanho 1314 utilizando AE1.	89

Lista de Tabelas

2.1	Número de possíveis árvores enraizadas e não enraizadas para n <i>OTUs</i> . . .	14
3.1	Matriz de distâncias para construção de árvore da Figura 3.4 [46].	23
5.1	Representação por vetor de características para a árvore da Figura 5.1. . .	63
5.2	Representação por predecessores para a árvore da Figura 5.2.	64
5.3	Representação por número de Prüfer para a árvore da Figura 5.3 [50]. . . .	65
5.4	Representação por Chaves Aleatórias de Redes para a árvore da Figura 5.4.	68
5.5	Complexidade de tempo e espaço onde n e m representam as quantidades de vértices e arestas, respectivamente.	77
7.1	AE1 para conjuntos de seqüências com inicialização aleatória.	90
7.2	AE1 para 55 seqüências de tamanho 1314, onde In. Al. significa inicializa- ção aleatória da população sem a utilização de <i>seeding</i>	91
7.3	AE1 para conjuntos de seqüências de tamanho 1314 com inicialização ale- atória.	92

Introdução

A construção de árvores filogenéticas (ou simplesmente filogenias) é um método desenvolvido para responder a algumas questões sobre a evolução biológica [51, 61]. Um dos principais problemas é explicar a história da evolução das espécies existentes hoje. Mais especificamente, os biólogos buscam explicar as relações ancestrais entre as espécies, o que têm sido importante em várias áreas da biologia e disciplinas relacionadas [29].

A taxonomia permite o estabelecimento de uma nomenclatura internacional relativamente uniforme na classificação dos seres vivos. Na virologia, as árvores filogenéticas têm sido utilizadas para definir agrupamentos de vírus. Por meio desses agrupamentos pode-se determinar se os pacientes estão relacionados epidemiologicamente e estudar a população infectada por meio do conhecimento do processo evolutivo do vírus [8]. A filogenia têm sido muito aplicada também no alinhamento múltiplo de seqüências. Alguns programas de alinhamento múltiplo de seqüências requerem uma árvore filogenética para reduzir o tempo de computação necessário. Resultados de predição de genes podem ser substancialmente refinados por meio da filogenia do gene em questão, sendo esta abordagem chamada de filogenômica [17]. Neste caso, a partir de genes com funções conhecidas, é construída uma árvore filogenética para predizer a função do gene em análise.

Para obtenção de uma filogenia, utiliza-se um tipo especial de grafo, chamado árvore (ver Apêndice), que descreve as relações ancestrais entre as espécies (nós do grafo).

As folhas (ou nós terminais) da árvore representam as espécies atualmente existentes e os nós internos correspondem a ancestrais hipotéticos. A Figura 1.1 ilustra uma árvore filogenética para um subconjunto de primatas.

Figura 1.1: Árvore filogenética para um grupo de primatas [61].

Outros tipos de objetos (não somente espécies) também evoluem como, por exemplo, as moléculas de *DNA* e de proteínas. Desta forma, os pesquisadores do campo de biologia molecular estão interessados em construir árvores filogenéticas para esses objetos.

Conforme mencionado anteriormente, os ancestrais de uma espécie são hipotéticos, uma vez que não se tem certeza se um fóssil em particular pertence ou não a uma espécie. Além disso, em geral, não existe informações suficientes sobre os ancestrais que estão mais distantes das espécies existentes hoje. Portanto, uma filogenia criada por um pesquisador, é apenas uma hipótese.

Em geral, há muitas possíveis topologias de árvore que poderiam explicar a história evolutiva de alguns objetos. De fato, existe um número combinatorial de árvores possíveis a serem analisadas, consistindo em um problema *NP*-completo [9]. Na tentativa de solucionar este problema, os métodos utilizando Algoritmos Evolutivos (AEs) têm mostrado resultados relevantes.

Por outro lado, a determinação de árvores filogenéticas é um problema de Projeto de Redes (PR) e para este tipo de problema, os AEs possuem algumas dificuldades de codificação. A utilização dos Tipos Abstratos de Dados (TADs) convencionais para os cromossomos dos AEs tende a produzir muitas soluções ineficazes para problemas envolvendo grafos. Desta forma, uma parte do trabalho computacional é consumida com a geração de soluções inviáveis, limitando o desempenho do AE. Além disso, os TADs

de árvore para AEs de uma forma geral não conseguem atingir todas as características básicas adequadas para codificação de uma árvore [50].

Dessa maneira, as abordagens evolutivas desenvolvidas para filogenia também têm seu desempenho afetado pelas dificuldades de codificação. Assim, uma estrutura de dados mais adequada para representar as árvores filogenéticas computacionalmente pode proporcionar melhorias de desempenho dos AEs para esse problema.

Este trabalho propõe o desenvolvimento de um AE utilizando a representação por conjunto de arestas (Seção 5.1.5) para o problema de filogenia. Estudos preliminares revelam que tal representação possui as características de adequação de codificações de árvores propostas por Gen *et al* [25].

O texto está organizado da seguinte forma: o Capítulo 2 introduz os fundamentos biológicos de evolução e filogenia. O Capítulo 3 apresenta os principais métodos filogenéticos. O Capítulo 4 descreve os principais AEs, limitações da codificação convencional de AEs para problemas envolvendo a reconstrução de árvores e AEs para filogenia encontrados na literatura. O Capítulo 5 aborda as codificações existentes na literatura para AEs aplicados à reconstrução de árvores de grafos. O Capítulo 6 apresenta uma nova proposta de AE para filogenia baseada na codificação por Conjunto de Arestas. O Capítulo 7 mostra a análise dos resultados. O Capítulo 8 apresenta as considerações finais.

Evolução e Filogenia

Como surgiram as espécies existentes na Terra? Essa é uma pergunta que não é fácil de ser respondida. Há controvérsias em relação a esta questão, não somente no campo científico, como também na área ideológica e religiosa. De um lado, o criacionismo, em que os teólogos afirmam que Deus, criou todos os seres (espécies) que foram colocados no mundo já adaptados ao ambiente onde foram criados e permaneceram imitáveis ao longo dos tempos. Por outro lado, o evolucionismo, também conhecido por transformismo, consideram as espécies animais e vegetais atuais como sendo o resultado de lentas e sucessivas modificações sofridas por espécies existentes no passado.

Para explicar mecanismos de evolução das espécies, surgiram várias teorias, dentre as quais destacam-se as teorias de Lamarck, de Darwin e a Teoria Sintética da Evolução (ou Neodarwinismo).

2.1 *Lamarckismo*

O grande naturalista francês, Jean Baptiste de Lamarck (1744-1829) foi o evolucionista mais famoso de seu tempo [2]. Sua teoria foi publicada em 1809 em um livro denominado Filosofia Zoológica. Lamarck sustentou que os padrões de semelhanças dos organismos

originam-se por modificações evolutivas, ou seja, que as espécies tinham relações de ancestralidade em comum. Expôs também que alterações ambientais desencadeariam em uma espécie a necessidade de modificação, no sentido de promover a sua adaptação às novas condições do meio. A espécie que adquirisse novos hábitos, utilizaria com mais frequência certas partes do organismo, fazendo com que estas se desenvolvessem e as partes pouco utilizadas atrofiassem. Assim, a espécie adquiriria características novas, que seriam herdadas por seus descendentes. As idéias de Lamarck podem ser resumidas da seguinte forma:

1. **Lei do uso e desuso:** o uso frequente de partes do organismo conduz à hipertrofia e o desuso prolongado causaria atrofia, o que explicava o desaparecimento das características que não tinham mais utilidade para a nova espécie;
2. **Lei da transmissão das características adquiridas:** as características adquiridas pelo uso ou perdidas pelo desuso seriam transmitidas aos descendentes.

Lamarck utilizou vários exemplos da natureza para explicar sua teoria. Segundo ele, as girafas teriam, a princípio, pescoços curtos e viveriam em ambientes onde a vegetação rasteira era relativamente escassa. Assim, teriam sido forçadas, por imposição do meio, a alimentarem-se de folhas situadas no alto das árvores. Adquiriram, então, o hábito de esticar o pescoço, no esforço para terem acesso ao alimento. Essa característica adquirida foi lentamente transmitida de geração a geração, até resultarem nas atuais girafas de pescoço longo.

A teoria lamarckista logo se popularizou, despertando severas reações de uma sociedade criacionista. Em especial, de um naturalista protestante fortemente adepto do criacionismo e apologista bíblico, Georges Leopold Cuvier, autor respeitado de um denso e valioso trabalho que inclui principalmente anatomia.

A firmeza de Lamarck perante a ira dos conservadores resultaria futuramente em seu obscurecimento e acabaria morrendo na miséria, em 1829. Por outro lado, sua teoria foi considerada viável até 1893, quando um trabalho publicado pelo biólogo alemão August Weismann [66], verificou que organismos superiores apresentam dois tipos de células: as células germinativas (que passam informação genética aos descendentes) e as células somáticas (que compõem o organismo em suas partes não diretamente associadas à reprodução), indicando a impossibilidade de transmitir aos descendentes as características adquiridas pelas células somáticas. Para isso, Weismann realizou seus experimentos em

camundongos. Esse pesquisador cortou, por várias gerações, os rabos dos camundongos. Os descendentes, apesar disso, sempre apresentavam rabos. A partir desse experimento, Weismann demonstrou que a característica adquirida pelos ratos (ausência de cauda) não eram herdadas pelos descendentes.

2.2 De Darwin à Teoria Sintética da Evolução

O interesse de Darwin pela evolução surgiu durante uma viagem que ele realizou ao redor do mundo a bordo do navio inglês *H.M.S. Beagle*. Nessa viagem, que teve duração de 5 anos, Darwin coletou vários exemplares de animais, plantas e fósseis e fez observações sobre as diferenças que encontrava entre indivíduos da mesma espécie.

Após a sua viagem, em 1837, Darwin começou a estudar mais detalhadamente os animais domésticos. Concluiu que as raças de organismos domésticos (galinhas, pombos, etc.) foram criadas pelo homem, que escolhia os indivíduos para os cruzamentos. De geração em geração, ao longo dos anos, eram reproduzidos os indivíduos que possuíam uma determinada característica que fosse de interesse e ao mesmo tempo, outros indivíduos eram impedidos de reproduzir. Esse processo é chamado de seleção artificial. Por este procedimento, eram obtidas novas raças e variedades que interessavam ao homem.

Darwin estava convencido de que as espécies modificavam-se e a partir de então começou a questionar como as espécies mudavam na natureza. Para elucidar esta questão, Darwin passou a estudar os fósseis. Ao comparar os fósseis de diferentes camadas geológicas, Darwin concluiu que os seres vivos modificavam-se ao longo do tempo e que algumas características de animais extintos continuavam existindo em animais atuais. As camadas geológicas mais recentes mostravam fósseis de organismos mais semelhantes aos seres vivos. Por exemplo, foi encontrado na Patagônia o fóssil de um mamífero gigantesco, já extinto, muito semelhante ao tatu que vive na América do Sul.

Apesar de todos os estudos realizados, Darwin ainda procurava uma comprovação da ocorrência da modificação das espécies. Em 1838, ele conheceu a teoria de Malthus [24] sobre o crescimento populacional. Malthus dizia que o potencial de reprodução da população humana é infinitamente maior do que a capacidade da terra de produzir os meios de subsistência. Dizia também que, se uma população não encontrasse obstáculos ao crescimento, haveria um aumento no número de indivíduos de acordo com uma progressão geométrica. Por outro lado, os meios de subsistência aumentariam de acordo com uma progressão aritmética. Malthus tentava imaginar a humanidade submetida às mesmas leis

que regem populações de outras espécies. Esse foi um dos pontos que chamou a atenção de Darwin. As populações poderiam, teoricamente, crescer muito rápido. No entanto, isso não era observado na prática. Para explicar a manutenção de níveis aproximadamente constantes no tamanho da população, Darwin achava que deveria existir uma “luta pela vida”. De uma população com indivíduos diferentes, aqueles que possuíam características mais adaptadas ao ambiente teriam mais oportunidades de sobreviverem e produzirem descendentes. Com o passar do tempo, as diferenças entre os novos indivíduos e os da população original iriam se acentuando a ponto de constituírem espécies novas.

Darwin elaborou toda sua teoria com base nos dados coletados em sua viagem, em observações de animais domésticos e na análise de trabalhos de outros pesquisadores. Em 1859, propôs um mecanismo para explicar a evolução biológica por meio de sua obra “*A Origem das Espécies*” [24]. Segundo Darwin, os organismos mais bem adaptados ao meio têm maiores chances de sobreviver do que os menos adaptados, deixando um número maior de descendentes. Os organismos mais aptos são, portanto, selecionados para aquele ambiente.

A abordagem de Darwin sobre a evolução era bastante distinta daquela de Lamarck. Considere o exemplo da girafa: de acordo com Darwin, animais com pescoço um pouco maior colhiam as folhas com mais facilidade e outros com pescoço um pouco menor, tinham mais dificuldade de se alimentar. Assim, com o tempo, os animais de pescoço comprido foram favorecidos pelo ambiente, isto é, foram selecionados naturalmente e os animais de pescoço menor acabaram sendo extintos naquela região.

Os princípios básicos das idéias de Darwin podem ser resumidos da seguinte maneira:

- Os indivíduos de uma mesma espécie apresentam variações em todos os caracteres, não sendo, portanto, idênticos entre si;
- Todo organismo tem grande capacidade de reprodução, produzindo muitos descendentes. Entretanto, apenas alguns dos descendentes chegam à idade adulta;
- O número de indivíduos de uma espécie é mantido mais ou menos constante ao longo das gerações;
- Assim, existe a chamada “luta pela vida” entre os descendentes, pois apesar de nascerem muitos indivíduos, poucos atingem a maturidade, mantendo constante o número de indivíduos na espécie;

- Na “luta pela vida”, organismos com variações favoráveis às condições do ambiente onde vivem têm maiores chances de sobreviver, quando comparados com os organismos com variações menos favoráveis;
- Os organismos com essas variações vantajosas têm maiores chances de deixar descendentes. Como há transmissão de caracteres de pais para filhos, estes apresentam essas variações vantajosas;
- Desta forma, ao longo das gerações, a atuação da seleção natural sobre os indivíduos mantém ou melhora o grau de adaptação destes ao meio.

Darwin conseguiu elaborar uma teoria que explicava a mudança das espécies, baseando-se no princípio da seleção natural, mas não conseguiu explicar como surgia a variabilidade em indivíduos da mesma espécie. Havia simplesmente constatado que dentro de uma mesma espécie existiam indivíduos diferentes, porém faltava uma teoria satisfatória em relação a hereditariedade [57].

Na época em que Darwin publicou seu livro, em 1859, um naturalista chamado Gregor Mendel estava estudando os mecanismos de herança dos caracteres. Até esse momento não se conheciam os cromossomos nem os genes. Porém, somente no início de 1900 é que os trabalhos de Mendel foram retomados. Em 1920, já era reconhecido que os mecanismos de herança descobertos por Mendel estavam ligados aos cromossomos.

Em 1940, uma teoria evolucionista mais consistente tomou forma, auxiliadas pelas leis da genética de Mendel denominada Teoria Sintética da Evolução ou Neodarwinismo [57]. Esta teoria fundamenta-se nos mecanismos de seleção natural, mutação e recombinação gênica. De acordo com o Neodarwinismo, a variabilidade genética pode ser gerada pela recombinação gênica, que é a variação natural ocorrida com o cruzamento das informações genéticas dos progenitores de um indivíduo (50% do pai e 50% da mãe). Essas informações nunca ocorrem da mesma forma em descendentes diferentes. Além disso, no processo de cópia das informações genéticas dos pais podem ocorrer falhas produzindo um aumento da variabilidade genética em um processo chamado mutação.

As mutações geralmente são insignificantes. Como a maioria das espécies já estão adaptadas ao seu ambiente, a maioria das mutações significativas acabam sendo desvantajosas e são eliminadas pela Seleção Natural. No entanto, quando se diz que um indivíduo é mutante, significa que sua mutação é suficientemente grande para resultar em alguma diferença.

2.3 Fundamentos da Genética

Gregor Mendel foi o primeiro a afirmar que diferenças nas características físicas dos organismos poderiam ser causadas por diferenças nos fatores hereditários. Apenas 7 anos depois da publicação sobre a origem das espécies por Darwin, Mendel publicou sua descoberta. Os estudos de Mendel foram baseados nos fatores hereditários que determinam a constituição física das ervilhas, incluindo tamanho, forma e cor. Esses fatores foram chamados genes. Os genes, por sua vez, são compostos de ácido desoxirribonucleico (*DNA*) [71]. A informação contida no gene é transmitida para as gerações seguintes por meio da replicação do *DNA* e é traduzida por meio de dois processos resultando na síntese de proteínas. As proteínas determinam a forma e estrutura da célula e desempenham variadas funções biológicas [71]. Exemplo de proteínas são os anticorpos que participam da defesa imunológica, e as enzimas que catalisam diversas reações químicas nos seres vivos.

2.3.1 *DNA, RNA e Cromossomos*

A nível bioquímico, o *DNA* é composto por ácidos nucleicos que consistem da combinação de quatro unidades químicas simples denominadas nucleotídeos. Um nucleotídeo é formado por um açúcar (a desoxirribose), um grupo fosfato e uma base nitrogenada. Os nucleotídeos são diferenciados pela base nitrogenada. As principais bases nitrogenadas são: adenina (*A*), guanina (*G*), citosina (*C*) e timina (*T*), sendo as duas primeiras denominadas purinas e as duas últimas chamadas de pirimidinas.

A informação genética pode também ser carregada por outro ácido nucleico, chamado de ácido ribonucleico (*RNA*). O *RNA* possui propriedades similares ao *DNA*, porém com a substituição da desoxirribose pela ribose e da base timina pela uracila (*U*).

Uma molécula de *DNA* é composta por duas seqüências de nucleotídeos (também chamadas de fitas) em direções opostas (anti-paralelas) que se enroscam uma sobre a outra formando o modelo de dupla hélice (separação das duas fitas), proposto por Watson e Crick [71] (ver Figura 2.1). Tal modelo fica armazenado em uma estrutura genética denominada cromossomo. Os cromossomos geralmente ficam agrupados aos pares, chamados de cromossomos homólogos.

A posição que o gene ocupa no cromossomo é denominado *locus*. O mesmo *locus* pode estar ocupado em diferentes cromossomos homólogos por formas gênicas diferentes. Genes

Figura 2.1: Estrutura de *DNA*.

com variações em cromossomos diferentes são denominados alelos e o conjunto completo de cromossomos de um indivíduo é chamado genoma.

2.3.2 Replicação do *DNA*

Para as informações genéticas serem propagadas para as gerações sucessivas, é necessário que o *DNA* seja replicado e, assim, novas cópias das duas fitas sejam produzidas.

O primeiro passo na replicação do *DNA* é a abertura da dupla hélice por proteínas chamadas helicases. Após a separação das duas fitas de *DNA*, novas cópias de cada fita são produzidas por proteínas chamadas polimerases. Cada fita original age como um molde para a produção de uma nova fita. Finalmente, a fita molde é unida à fita recém-sintetizada por enzimas chamadas ligases.

2.3.3 Mutação

Apesar da replicação (cópia) de *DNA* ser um processo eficiente, às vezes pode não funcionar corretamente, implicando em erros ou mutações. Enquanto que algumas mutações vistas nas seqüências de *DNA* acontecem devido a erros durante a cópia, outras ocorrem espontaneamente ou por fatores ambientais como radiações ultravioleta e química.

Há diferentes tipos de mutação. A maioria envolve a substituição de uma base por outra em uma seqüência de *DNA* [71]. Este tipo de mutação pode ser do tipo transição ou transversão. As transições ocorrem se uma purina (*A* ou *G*) é substituída por outra purina ou se uma pirimidina (*C* ou *T*) é substituída por outra pirimidina. As transversões acontecem se uma pirimidina é substituída por uma purina ou vice-versa. A ocorrência de transições é mais freqüente do que de transversões [41].

Uma mutação ocorrida no *DNA* modifica sua estrutura criando um novo alelo para um determinado gene. Essa molécula de *DNA* alterada continua a se duplicar normalmente,

ou seja, pode ser transmitida de forma hereditária para as futuras gerações.

Caso o *DNA* se duplicasse sempre da mesma maneira, sem nunca se modificar, todos os organismos com o mesmo *DNA* seriam exatamente iguais. Aliás, não existiriam outros organismos vivos senão aqueles de bilhões de anos atrás, quando a vida na Terra se iniciou. Assim, por meio da mutação, acredita-se que foi gerada a variabilidade existente entre os organismos vivos.

2.3.4 Recombinação Gênica

Enquanto as mutações ocorrem em cromossomos de forma individual, o processo chamado recombinação acontece entre pares homólogos de cromossomos ou até mesmo, entre cromossomos diferentes.

O processo de recombinação gênica (*crossover*) ocorre durante a meiose¹, onde os cromossomos homólogos formam pares antes da segregação para gametas² diferentes. A grosso modo, partes de cromossomos são trocados entre si.

2.4 Reconstrução Filogenética

A partir da idéia de que os organismos vivos podem ser relacionados por meio de padrões de descendência, partilhando um ancestral comum mais recente ou antigo, surgiu um grande interesse no estudo das relações de parentesco entre os organismos, especialmente na época de Darwin, iniciando-se assim estudos em reconstruções filogenéticas.

Uma filogenia, por mais absurda que possa parecer, consiste em uma possível explicação para a origem das espécies (ou outros objetos) sendo analisadas(os). Para reconstruir árvores que melhor refletem a história evolutiva são necessários métodos de inferência filogenética consistentes com a realidade biológica. Assim, várias abordagens foram desenvolvidas, dentre essas destacam-se: a fenética e a cladística [39, 69]. A fenética (também chamada taxonomia numérica) não foca as relações de parentesco entre os organismos, apenas leva em consideração a similaridade entre as espécies [69]. Por este motivo, privilegia os caracteres diretamente observáveis ou morfológicos, utilizando informações primárias de características fenotípicas (características que podem ser pesadas, medidas, numeradas, etc.). O taxonomista deve, portanto, primeiro descrever o organismo, recorrendo ao

¹Meiose: processo de divisão celular responsável pela formação dos gametas. Caracteriza-se por promover a redução do número de cromossomos da espécie pela metade.

²Gameta: célula encarregada da reprodução por meio da fecundação.

máximo de características que puder. Tais características são apresentadas em uma representação chamada fenograma [68, 51]. Quanto mais características as espécies possuírem em comum, maior será o grau em que essas estão próximas. Entretanto, o fenograma pode não representar a filogenia verdadeira, pois este não considera as taxas diferenciais de evolução para cada característica, apenas a similaridade entre as características.

A cladística baseia-se na hipótese da mais próxima ancestralidade comum entre as espécies [39]. A ancestralidade entre os estágios evolutivos de uma característica é representada em um diagrama hierárquico denominado cladograma.

2.4.1 Árvores Filogenéticas

Uma árvore filogenética é um grafo que mostra as relações evolutivas entre organismos (ou táxons³). A árvore é composta por nós internos, nós terminais e por arestas. A Figura 2.2 ilustra uma árvore filogenética, onde os nós terminais (*A*, *B*, *C* e *D*) representam as unidades taxonômicas operacionais ou simplesmente, *OTUs* (do inglês: *Operational Taxonomic Units*), unidas por arestas aos nós internos (*E*, *F* e *G*) que representam os ancestrais comuns hipotéticos mais recentes desses táxons.

Figura 2.2: Representação de uma árvore filogenética.

Árvore enraizada e não enraizada

Uma árvore filogenética pode ser enraizada ou não enraizada [48, 68]. Em uma árvore enraizada, a raiz representa o ancestral comum do conjunto inteiro das *OTUs* apresentando uma direção que corresponde ao tempo de evolução [48]. Uma árvore não enraizada é uma

³Os táxons podem ser famílias, gêneros, espécies ou populações, ou seja, qualquer nível taxonômico sobre os quais se deseja inferir a história evolutiva.

árvore que especifica somente os relacionamentos entre as *OTUs* (ver Figura 2.3). Uma árvore enraizada é mais comumente construída para estudar os relacionamentos evolutivos.

Figura 2.3: Árvores enraizada e não enraizada.

Número de topologias

Muitos métodos para a reconstrução de árvores filogenéticas utilizam um critério para avaliar a adequação de um dado conjunto de topologias buscando pela árvore que representa a melhor filogenia. No entanto, o número de diferentes topologias para um conjunto de seqüências aumenta muito rapidamente, como ilustrado na Tabela 2.1.

Número de <i>OTUs</i>	Árvores enraizadas	Árvores não enraizadas
2	1	1
3	3	1
4	15	3
5	105	15
6	945	105
7	10395	945
8	135135	10395
9	2027025	135135
10	34459425	2027025

Tabela 2.1: Número de possíveis árvores enraizadas e não enraizadas para n *OTUs*.

O número de árvores possíveis varia de acordo com o número de *OTUs* e com a presença

ou a ausência de raiz. Os números exatos de possibilidades de relações entre *OTUs* para árvores enraizadas e não enraizadas são dados, respectivamente, pelas fórmulas [21]:

$$N = \frac{(2n - 3)!}{2^{n-2}(n - 2)!}$$

e

$$N^* = \frac{(2n - 5)!}{2^{n-3}(n - 3)!}$$

onde n corresponde ao número de táxons.

De fato, o número de possíveis árvores aumenta exponencialmente a cada incremento de *OTUs*. Essa explosão no número de topologias é um problema fundamental para a filogenia [48], onde o objetivo é encontrar uma árvore que melhor explica as relações evolutivas entre as espécies.

Redes

Em geral, um processo evolutivo entre um grupo de espécies é explicado por uma árvore filogenética. Contudo, em casos onde o gene passou pelo processo de recombinação, uma rede pode ser mais apropriada [48]. A Figura 2.4 ilustra uma rede, cuja representação comporta a presença de um ou mais ciclos.

Figura 2.4: Ilustração de uma árvore e uma rede filogenéticas [48].

2.5 Considerações Finais

Com o surgimento das leis da Genética foi possível entender melhor e estudar os processos evolutivos por meio da filogenia. A filogenia, representada geralmente por árvores, é um problema combinatorial em relação ao número de espécies. No Capítulo 3 são apresentados vários métodos para tratar este problema.

Métodos Computacionais para Filogenia

Para inferir as relações evolutivas de um determinado grupo de espécies, considerando seqüências de *DNA*, é necessário a escolha de um método filogenético, bem como seqüências que apresentem ancestralidade comum, ou seja, que sejam homólogas.

Quando não se comparam características homólogas, pode-se incidir no erro de considerar similaridades sem origem comum e, portanto, com histórias evolutivas diferentes. Uma das formas de avaliar esta escolha é incluir nas análises, seqüências de grupos externos (organismos com história evolutiva conhecida em relação ao grupo em estudo), que funcionam como controles no processo de reconstrução de parentescos [53].

Após a seleção das seqüências, é necessário realizar o alinhamento, determinando a similaridade entre as mesmas [35] e então, construir árvores filogenéticas a partir dos métodos existentes. Tais métodos podem ser classificados de acordo com os tipos de dados ou pela técnica utilizada na reconstrução de filogenias.

3.1 Tipos de Dados: Características $\times$ Distância

De acordo com os dados a serem analisados, os métodos podem ser baseados em distâncias ou em características [48]. As distâncias representam uma estimativa da separação

evolutiva entre os pares de organismos. As características dizem respeito, por exemplo, à presença de genes específicos em alguns genomas e ausência em outros.

Os métodos filogenéticos baseados em características analisam cada informação evolutiva conhecida, isto é, ao invés de reduzir todas as variações individuais entre as seqüências para um único valor (métodos baseados em distâncias), cada característica é tratada separadamente para determinar os relacionamentos de ancestralidade mais prováveis.

3.2 Métodos de Reconstrução: *Clustering* $\times$ *Critério de Ótimo*

Uma outra forma de classificação pode ser feita baseando-se nos métodos de reconstrução filogenética, os quais são separados em: métodos de *clustering* e métodos de critério de ótimo. Nos métodos de *clustering* (ou métodos não baseados em modelo [68]), a árvore filogenética é construída passo a passo a partir das *OTUs*. As vantagens dessas abordagens são: facilidade de implementação e eficiência computacional dos programas [48]. Contudo, tais métodos possuem a limitação de sempre produzirem uma única árvore, impossibilitando a avaliação de outras árvores candidatas possivelmente melhores.

A reconstrução filogenética a partir do critério de ótimo (ou método baseado em modelo [68]) escolhe a melhor filogenia de acordo com um critério, dentro de um conjunto de árvores candidatas. Este critério é utilizado para atribuir a cada árvore um *score* que é uma função de relacionamento entre a árvore e os dados. Esta função pode ser, por exemplo, baseada em um modelo de como as seqüências evoluem. Por meio do critério de ótimo pode-se avaliar a qualidade de cada árvore e, então, escolher aquela, dentre várias hipóteses, que melhor explica os dados.

Os métodos de critério de ótimo envolvem a resolução de duas questões: primeiro, a partir de um conjunto de dados e uma árvore, qual é o valor de critério de ótimo para aquela árvore? (Em outras palavras, qual é o número mínimo de eventos evolutivos para explicar tais dados?). Em segundo lugar, de todas as possíveis árvores, qual delas infere o menor número de eventos evolutivos?

A primeira questão pode ser resolvida em tempo polinomial. No entanto, o segundo problema pertence à classe dos problemas *NP*-completos [67]. Para este problema são apresentados na Seção 3.9, alguns algoritmos de busca. Nas Seções seguintes são descritos os principais métodos de reconstrução filogenética.

3.3 UPGMA

O método *UPGMA* (do inglês: *Unweighted Pair Group Method with Arithmetic Mean*) é o método mais simples e mais utilizado. Este método de construção de árvores filogenéticas assume a hipótese do relógio molecular¹ [41] e utiliza um algoritmo de *clustering* hierárquico [37], nos quais as relações topológicas são identificadas por ordem de similaridade e a árvore filogenética é construída passo a passo. Dadas n *OTUs*, primeiro deve-se identificar dentre várias, as duas que são mais similares e tratá-las como uma única, chamada de *OTU* composta. A partir daí, os outros grupos de *OTUs* são observados e identificado o próximo par com maior similaridade, que é novamente arranjado e, assim por diante, até que restem apenas duas *OTUs*.

3.3.1 Algoritmo

Para todos os pares de seqüências são calculadas as distâncias evolutivas e os valores destas distâncias são apresentados em formato matricial, onde d_{ij} representa a distância entre os elementos i e j .

Algoritmo 1 - *Clustering* hierárquico (*UPGMA*) [16].

```

1:  $C \leftarrow \{C_1, C_2, \dots, C_n\}$ ; /* onde  $C_i$  é o cluster definido para a OTU  $i$ . */
2:  $T \leftarrow \{\text{nós folhas}\}$ ; /* a cada nó folha é atribuído peso 0. */
3: while ( número de clusters  $> 1$  ) do
4:   Determine dois clusters  $i$  e  $j$  cuja distância  $d_{ij}$  é mínima;
5:    $C_k \leftarrow C_i \cup C_j$ .
6:   for (todo  $l \neq k$ ) do
7:      $d_{kl} \leftarrow (d_{il}|C_i| + d_{jl}|C_j|)/(|C_i| + |C_j|)$ ;
8:   end for
9:    $C \leftarrow C \cup C_k$ ;
10:   $C \leftarrow C \setminus \{C_i, C_j\}$ ;
11:  Defina um nó  $k$  com os nós filhos  $i$  e  $j$  e atribua peso  $d_{ij}/2$ ;
12: end while
```

O algoritmo para o método *UPGMA* (ver Algoritmo 1) consiste em inicialmente atribuir cada *OTU* i a um *cluster* C_i , onde $1 < i \leq n$, e definir n nós folhas (um nó para cada *OTU*) em uma árvore T , todos com peso zero. Iterativamente, são determinados dois

¹De acordo com o relógio molecular, as linhagens evoluem a uma taxa constante de evolução.

clusters (C_i e C_j) cuja distância d_{ij} é mínima e agrupados em um único *cluster*, criando um novo *cluster* (C_k). As distâncias entre k e todos os outros *clusters* são computadas, os *clusters* C_i e C_j são removidos e em T é inserido um novo nó k , tendo como nós filhos i e j e peso $d_{ij}/2$ (ver Figura 3.1).

Figura 3.1: Ilustração do funcionamento do método *UPGMA* [16].

Uma vantagem do método *UPGMA* advém do fato deste ser computacionalmente rápido. Entretanto, a grande desvantagem deste método consiste em não se poder avaliar várias árvores filogenéticas, uma vez que este método constrói apenas uma árvore. Uma outra desvantagem deve-se ao fato do método assumir linhagens com mesmo tamanho de ramos (relógio molecular).

3.4 Neighbor Joining

Saitou e Nei [46] descreveram um método denominado *Neighbor Joining* (*NJ*) para identificar os pares mais próximos de *OTUs* ou vizinhos (do inglês: *neighbors*), de forma a minimizar o comprimento total da árvore construída. Um par de vizinhos é definido

como sendo formado por um par de *OTUs* conectados por uma aresta em uma árvore não enraizada bifurcada (duas arestas unidas por um nó interno). Como exemplo, considere a Figura 3.2. As *OTUs* *A* e *C* formam um par de vizinhos pois estão conectados por meio de um nó interno, *B*. O número de pares de vizinhos em uma árvore depende de sua topologia. Para uma árvore com n (≥ 4) *OTUs*, o número mínimo é sempre 2, enquanto o número máximo de pares de vizinhos é $n/2$, se n for par, e $(n - 1)/2$, caso contrário.

Figura 3.2: Árvore não enraizada bifurcada.

3.4.1 Algoritmo

O objetivo do algoritmo *Neighbor Joining* é a construção da topologia da árvore considerando-se as distâncias (M_{ij}) entre as *OTUs* e o tamanho das arestas da árvore (d_{ij}) [46].

O algoritmo começa com uma árvore-estrela [46], como ilustrado na Figura 3.3. Em seguida, procura-se o par de vizinhos (que além de estarem próximos um do outro, também estejam distantes do restante das *OTUs*), que minimize a soma total dos ramos da árvore. Em cada iteração, o algoritmo tenta encontrar o ancestral imediato de duas espécies na árvore. Para cada nó i é computado o valor u_i , que é a distância do nó para o restante da árvore. O princípio do método é minimizar a soma total dos ramos, também conhecido como critério de evolução mínima [41]. Dois nós i e j para os quais $M_{ij} - u_i - u_j$ é mínimo, são agrupados formando uma nova espécie na árvore. Enquanto a árvore não estiver com todos os nós com grau 3 ou 1, escolhe-se aquelas *OTUs* que oferecem menor distância (ver Algoritmo 2).

De acordo com a matriz de distâncias dada como entrada (ver Tabela 3.1), uma árvore filogenética é construída utilizando-se o algoritmo *Neighbor Joining* como mostra a Figura 3.4. Inicialmente, o algoritmo seleciona os nós *A* e *B* para serem as primeiras *OTUs* agrupadas, pois $M_{AB} - u_A - u_B$ foi o menor valor encontrado. A seguir são computados

(a) Árvore-estrela sem nenhuma estrutura hierárquica.

(b) Agrupamento das OTUs A e B.

Figura 3.3: Ilustração da primeira iteração do algoritmo *Neighbor Joining*.**Algoritmo 2** - *Neighbor Joining*

- 1: Crie um nó para cada uma das n OTUs.
- 2: **while** (houver mais de dois nós restantes) **do**
- 3: Para cada espécie compute $u_i = \sum_{k \neq i} \frac{M_{ik}}{n-1}$.
- 4: Escolha i e j para os quais $M_{ij} - u_i - u_j$ é mínimo.
- 5: Una as espécies i e j formando uma nova espécie - (ij) , com um nó correspondente em T . Calcule o tamanho das arestas de i e j para o novo nó da seguinte forma:

$$d_{i,(ij)} = \frac{1}{2}M_{ij} + \frac{1}{2}(u_i - u_j), d_{j,(ij)} = \frac{1}{2}M_{ij} + \frac{1}{2}(u_j - u_i) \quad (3.1)$$

- 6: Compute as distâncias entre a nova espécie e a cada outra espécie:

$$M_{(ij)k} = \frac{M_{ik} + M_{jk} - M_{ij}}{2} \quad (3.2)$$

- 7: Exclua as espécies i e j da matriz de distâncias e as substitua por (ij) .
- 8: **end while**
- 9: Conecte os dois nós restantes por uma aresta de tamanho M_{ij} .

os tamanhos das arestas entre os nós pela equação (3.1). As OTUs A e B são combinadas para formar a nova OTU (AB) e as distâncias entre a nova OTU com as restantes são calculadas de acordo com a equação (3.2). Na próxima iteração, é escolhido o par de OTUs E e F. Mais uma vez, são calculados os tamanhos das arestas entre o par (EF) e o restante do grafo. Na 3ª iteração, o par (AB, C) é selecionado. Na iteração seguinte, as OTUs escolhidas são (ABC) e D. Na 5ª iteração, (ABCD, F) é identificado como o par

	A	B	C	D	E	F	G
B	7						
C	8	5					
D	11	8	5				
E	13	10	7	8			
F	16	13	10	11	5		
G	13	10	7	8	6	9	
H	17	14	11	12	10	13	8

Tabela 3.1: Matriz de distâncias para construção de árvore da Figura 3.4 [46].

a ser unido. Na iteração seguinte, as *OTUs* (*ABCDEF*) e *G* são agrupadas. Por último, as *OTUs* (*ABCDEFG*) e *H* (as duas *OTUs* restantes) são unidas. A topologia final é ilustrada na Figura 3.4(f).

Figura 3.4: Ilustração passo a passo da execução do algoritmo *Neighbor Joining* para a matriz da Tabela 3.1.

Com base empírica, o algoritmo *NJ* têm-se mostrado bastante eficiente na reconstrução de árvores filogenéticas com boa estimativa no tamanho dos ramos [65], quando o tamanho das instâncias não é muito grande [60].

3.5 Máxima Parcimônia

O princípio de parcimônia é filosófico e largamente empregado na Ciência. Foi proposto por um filósofo inglês, Ockam no século XVII, e seu enunciado, conhecido como a Navalha de Ockam ou Lei de Parcimônia, é o seguinte: se duas hipóteses explicam os dados com igual eficiência, deve-se adotar a mais simples.

Para análise dos dados, é necessário considerar apenas os sítios informativos² [68], porque nem toda a variabilidade das seqüências é útil para o método de máxima parcimônia, pois os sítios que apresentarem variação em uma única seqüência, contribuirão para todas as árvores com a mesma quantidade de substituições para explicar os dados. Uma regra básica é que a variabilidade das seqüências deve dividir as *OTUs* em pelo menos dois grupos, sendo que cada um tenha um mínimo de dois representantes [41].

A reconstrução de árvores filogenéticas baseada no método de máxima parcimônia explica o processo evolutivo que infere o menor número de mudanças ocorridas. Primeiramente, são selecionados os sítios informativos. Depois, para cada topologia é calculada o número mínimo de alterações necessárias para explicar tais dados. A topologia que oferecer o melhor *score* é escolhida como a árvore mais parcimoniosa.

O subproblema de avaliar o quão adequada é uma árvore é conhecido como o pequeno problema da parcimônia. O subproblema de como procurar uma árvore no conjunto de todas as árvores possíveis, eficientemente, é conhecido como o grande problema da parcimônia.

3.5.1 Pequeno Problema da Parcimônia

O pequeno problema da parcimônia consiste em determinar o custo de uma dada árvore filogenética [22]. Existem algoritmos polinomiais para resolver este problema. Dentre esses, têm-se a parcimônia de Fitch, que utiliza como entrada a topologia de uma árvore filogenética enraizada.

Algoritmo de Fitch

O algoritmo de Fitch calcula a parcimônia analisando sítio por sítio de uma dada seqüência (de *DNA*, por exemplo) e infere o número mínimo de mudanças necessárias para

²Para seqüências de *DNA*, sítios informativos são aqueles em que há pelo menos duas bases distintas nas seqüências sendo consideradas e também tenha pelo menos duas bases distintas em cada sítio.

explicar o processo de evolução. Este método é baseado em programação dinâmica [22]. O algoritmo constrói um conjunto de estados possíveis (possíveis nucleotídeos) para cada nó interno da árvore, que é computado a partir dos estados de seus filhos. O cálculo é iniciado pelas folhas da árvore filogenética. Para cada folha é atribuída o nucleotídeo do sítio em análise. Então, a árvore é percorrida em pós-ordem (todos os filhos do nó atual são visitados primeiro) aplicando-se o operador de parcimônia. Em geral, este operador funciona como a intersecção dos conjuntos de características se esta não for vazia, caso contrário, como a união dos dois conjuntos.

Para ilustração do algoritmo, considere a Figura 3.5, onde estão representadas três seqüências com dois nucleotídeos.

Figura 3.5: Ilustração do algoritmo de Fitch para três seqüências.

Ao aplicar o operador de parcimônia no primeiro sítio das duas primeiras seqüências $\{A, G\}$, tendo como ancestral o nó X , obtém-se $\{A, G\}$. Para o segundo sítio, de $\{T, T\}$ resulta em $\{T\}$. Realizando todas as permutações entre as possibilidades para o primeiro e segundo sítio, o nó X resulta em um ancestral com as seguintes hipóteses: AT e GT . Em relação ao nó Y , considerando os seus nós filhos e aplicando-lhes o operador de parcimônia, obtêm-se como seqüências hipotéticas: AC , AT , GC e GT .

Este exemplo dá origem a duas hipóteses de ancestrais para o nó X e duas para o nó Y , isto é, são quatro as árvores com mesma topologia que podem explicar as relações evolutivas entre as seqüências analisadas. A seguir é descrito o problema de encontrar a melhor árvore filogenética dentre várias.

3.5.2 Grande Problema da Parcimônia

O grande problema da parcimônia é encontrar uma ou mais árvores que melhor expliquem os dados dentre um conjunto de todas possíveis árvores. Este problema é provado

ser *NP*-completo [67], ou seja, a enumeração de todas as árvores é impraticável, dada a sua complexidade: o número de árvores binárias com n folhas é igual a $(2n - 3)!$ [22].

Este é um problema de otimização combinatorial³, no qual técnicas exatas tal como *Branch and Bound* são computacionalmente inviáveis para problemas com instâncias moderadas (por exemplo: de 30 a 40 *OTUs*). Desta forma, algoritmos utilizando heurísticas têm sido desenvolvidos, bem como AEs têm sido aplicados para resolver o problema de encontrar a melhor filogenia [7].

A grande vantagem do método de parcimônia consiste no fato de ter um conceito muito simples, ou seja, sua maneira de interpretar o processo evolutivo pode ser facilmente entendida [41]. De maneira simplificada, o método de parcimônia escolhe aquele caminho que infere o menor número de explicações para um determinado grupo de espécies.

A principal objeção ao método consiste na denominada “atração de ramos longos” [48, 1, 41], que acontece pelo fato de se considerar que a evolução entre as espécies ocorra de forma lenta ou que, pelo menos, ocorra em velocidades similares entre as linhagens. Para exemplificar o problema, considere as Figuras 3.6(a) e 3.6(b), onde duas espécies de evolução muito rápida e duas de evolução mais lenta. Neste exemplo, um dos organismos rápidos (*A* ou *D*) é mais aparentado a um dos mais lentos (*B* ou *C*), e o ramo interno da árvore também é curto. O método de parcimônia agrupa as duas espécies de evolução mais rápida (ver Figura 3.6(b)).

A “atração de ramos longos” ocorre devido à grande quantidade de substituições ocorridas nos ramos longos. Por causa dessas substituições é maior a probabilidade de ocorrerem muitas homoplasias (ver Seção 2.4) entre as seqüências destes táxons, tornando-as artificialmente semelhantes [1].

3.6 Máxima Verossimilhança

O método de máxima verossimilhança para a construção de árvores filogenéticas procura inferir a história que melhor descreve os dados observados, com base em um determinado modelo de evolução, para representar um processo evolutivo que realmente ocor-

³Um problema de otimização é caracterizado por dois elementos: o espaço de busca ou o espaço de soluções possíveis e a função de avaliação. O primeiro é identificado por meio de variáveis e seus possíveis valores. A função de avaliação fornece a informação utilizada para orientar a busca sobre o referido espaço, a fim de identificar uma solução que, preferencialmente, seja um ótimo global, ou certamente, um ótimo local.

Figura 3.6: Topologias ilustrando o problema da atração de ramos longos.

reú [41]. Assim, para cada topologia é computada a probabilidade desta ocorrer segundo um modelo evolutivo, selecionando o comprimento dos ramos de maneira a maximizar a probabilidade dos dados analisados em função da árvore em questão.

Como outros métodos probabilísticos, a máxima verossimilhança compara os desvios entre os valores esperados e os estimados a partir de certos parâmetros. Para isso, é necessário que se tenha modelos para predizerem o que seria esperado em determinada situação. No caso das filogenias moleculares, os modelos devem especificar o modo como as seqüências evoluem. Desse modo, o método verifica o quanto o conjunto formado por dados mais hipótese (árvore) concorda com o que seria esperado (modelo), considerando que caso essas seqüências tivessem realmente evoluído segundo o modelo e árvore utilizados [1].

3.6.1 Modelos Evolutivos

Os modelos evolutivos mais comuns, aplicados ao método de máxima verossimilhança são aqueles de Jukes e Cantor [31], Kimura [33], Felsenstein [19], Hasegawa-Kishino-Yano [30]. A diferença entre esses modelos está no número de tipos de substituição consideradas e se todas as bases possuem ou não a mesma freqüência de substituições [41]. Tais modelos de evolução são apresentados a seguir.

Modelo de um tipo de substituição - (Jukes e Cantor [31] - JC69)

Jukes e Cantor propuseram um modelo bastante simples, que considera que os nucleotídeos aparecem em igual freqüência em uma seqüência de *DNA* e todas as substituições

ocorrem na mesma taxa.

Modelo de dois parâmetros - (Kimura [33] - K80)

Baseado no fato de que substituições do tipo transições (substituições entre purinas ou entre pirimidinas) ocorrem com mais frequência do que transversões (entre uma purina e uma pirimidina ou vice-versa), Kimura desenvolveu um modelo que considera essas diferenças de frequências.

Modelo proporcional (Felsenstein [19] - F81)

Partindo do fato de que a frequência dos quatro nucleotídeos nem sempre é similar, Felsenstein elaborou um modelo para calcular a probabilidade de substituição levando em consideração a frequência de cada uma das bases. Se admitir que as frequências de bases são iguais, tal modelo transforma-se no modelo *JC69* [48].

Modelo HKY85 (Hasegawa-Kishino-Yano [30])

O modelo *HKY85* permite taxas diferentes para transição e transversão, assim como admite frequências desiguais de bases.

3.6.2 Cálculo da Verossimilhança de uma Árvore

A melhor estimativa da filogenia é a árvore mais verossímil [41], ou seja, aquela com maior probabilidade de explicar os dados dentre todas as possíveis árvores. Desta forma, o cálculo da verossimilhança deve ser realizado para todas as possíveis topologias, o que torna um processo computacionalmente inviável dependendo da quantidade de *OTUs* analisadas.

Considere a topologia da Figura 3.7. Suponha que *A*, *C* e *T* são os nucleotídeos observados em um determinado sítio para um grupo de espécies. O cálculo da probabilidade desse sítio envolve a probabilidade de o nucleotídeo observado ser *T*, dadas as probabilidades de o nucleotídeo em *X* ter sido *A* e mudado para *T*, ter sido *C* e mudado para *T*, ter sido *G* e mudado para *T* e de não ter sofrido alteração (ou seja, era *T* em *X*). Essas probabilidades são estimadas por meio de um modelo evolutivo, como os descritos anteriormente.

O cálculo anterior retorna a probabilidade de ocorrência dos nucleotídeos em um único sítio. A probabilidade de uma árvore explicar um conjunto de seqüências é o produto das

Figura 3.7: Ilustração das *OTUs* e dos possíveis ancestrais.

probabilidades de cada um dos sítios:

$$L = L_1 * L_2 * \dots * L_n = \prod_{i=1}^n L_i, \quad (3.3)$$

onde n é o número de sítios, L_i é a verossimilhança calculada para o sítio i , considerando-se todas as possíveis transições da topologia em estudo.

Diferente dos métodos de distância que reduzem os dados a um valor, o método de máxima verossimilhança utiliza o máximo de informações disponíveis [68] como, por exemplo, as probabilidades dos dados, o modelo de evolução, etc. Contudo, possui a desvantagem de necessitar de grande tempo de computação durante os cálculos das probabilidades especialmente quando a quantidade de *OTUs* a serem analisadas é grande.

3.7 Árvores de consenso

O objetivo da filogenia é reconstruir a verdadeira história de evolução de um grupo de espécies. No entanto, a aplicação dos métodos filogenéticos pode resultar em múltiplas árvores. Qual delas é a que melhor reflete a verdadeira filogenia? Em virtude desses resultados conflitantes, costuma-se utilizar métodos para juntar informações comuns a duas ou mais árvores em uma árvore de consenso (em inglês: *consensus tree*).

Uma árvore de consenso não é uma filogenia. Geralmente tal árvore contém politomias⁴. Assim, interpretar uma árvore de consenso como uma filogenia implica que as politomias contidas indicam divergências múltiplas ao invés da interpretação de falta de resolução.

⁴Uma árvore possui politomias quando uma espécie ancestral dá origem a três ou mais espécies.

3.8 Teste de confiança

3.8.1 Bootstrap

Trata-se de uma simples reamostragem com substituição pseudo-aleatória dos dados [41]. A idéia é reconstruir uma árvore original com todos os sítios em questão e a cada reamostragem é construída uma árvore réplica com reposição de sítios. Desta maneira, um sítio pode aparecer mais de uma vez e outros podem não ser amostrados. Ao final, a consistência do método é determinada por verificar quantas topologias recuperaram a árvore original.

3.9 Algoritmos de Busca para Filogenia

O problema da filogenia torna-se computacionalmente complexo à medida que aumenta o número de táxons. Desta forma, cabe ao pesquisador encontrar uma árvore dentre várias, aquela que melhor explica os relacionamentos evolutivos entre as espécies. Os métodos de critério de ótimo permitem avaliar diversas árvores em busca da melhor filogenia. No entanto, o número de árvores possíveis cresce combinatorialmente. Para lidar com estes problemas, vários métodos de busca têm sido aplicados.

3.9.1 Busca Exata

Busca exaustiva

A busca exaustiva consiste em enumerar todas as possíveis árvores e avaliar cada uma delas de acordo com um critério de ótimo para encontrar aquela de melhor custo.

Devido ao enorme crescimento de possíveis árvores para cada quantidade de táxons, a busca exaustiva torna-se inviável. Neste caso, é necessária a utilização de heurísticas para a exploração do espaço de busca.

Branch and bound

Inicialmente é definido um *upper bound*. Pode ser relacionado à técnica *stepwise addition* (ver Seção 3.9.2). A busca é iniciada com uma árvore com três nós e a cada inserção de ramos verifica se este é ou não promissor (ou seja, se tal inclusão não excede o *upper bound*) para continuar a busca na árvore (ver Figura 3.8).

Figura 3.8: *Branch and bound*.

3.9.2 Busca Heurística

Stepwise Addition

O método *stepwise addition* inicia com três táxons e então adiciona mais táxons, um de cada vez (tentando adicioná-lo em cada um dos possíveis ramos do passo anterior) de maneira a obter a melhor árvore de acordo com um critério de ótimo (ver Figura 3.9).

Star Decomposition

O algoritmo inicia com todos os táxons em uma árvore-estrela, formando pares de táxons até que a árvore possua apenas nós com grau 3 (nós internos) ou 1 (nós folhas) (ver Figura 3.10). Um exemplo de utilização desta técnica é o algoritmo de *Neighbor Joining*.

Branch Swapping

Os métodos de *branch swapping* funcionam basicamente da seguinte forma: uma sub-árvore é podada e rearranjada a um outro ramo da árvore restante. Em geral, estes

Figura 3.9: *Stepwise addition*.

métodos são utilizados para a otimização de ramos na tentativa de encontrar uma árvore mais parcimoniosa.

Métodos como *stepwise addition* empregam *branch swapping* a cada adição de ramos, onde são realizados rearranjos na árvore em análise. Se qualquer troca de ramos resultar em uma árvore com um valor melhor de critério de ótimo, esta árvore torna-se a árvore a ser otimizada. O *branch swapping* é utilizado basicamente de três formas:

- **Nearest Neighbor Interchange - NNI:** este método identifica um ramo interior que conecta quatro sub-árvores, troca duas das sub-árvores em extremidades opostas do ramo. Dois rearranjos são possíveis (ver Figura 3.11).
- **Subtree Pruning and Regrafting - SPR:** identifica e destaca uma sub-árvore para reinseri-la a todas as possíveis posições. NNI produz um subconjunto das árvores geradas por SPR.
- **Tree Bisection and Reconnection - TBR:** este método divide a árvore em duas partes e a reconecta por um par de ramos, tentando todos os possíveis pares

Figura 3.10: *Star decomposition*.

de ramos para a reconexão. *NNI* e *SPR* produz subconjuntos de árvores geradas pelo método *TBR*.

Algoritmos Evolutivos

Cada vez mais os AEs vêm sendo utilizados na tentativa de solucionar o problema de filogenia. A Seção 4.3.2 contém uma revisão da literatura sobre AEs aplicados à Filogenia.

3.10 Considerações Finais

Após o estudo de todos os métodos filogenéticos apresentados, surge uma pergunta: qual método deve ser utilizado? A resposta depende do conjunto de dados de entrada como, por exemplo, o tamanho das seqüências e a similaridade entre as mesmas.

Cada método tem suas particularidades, fornecendo vantagens e também algumas desvantagens quanto à sua utilização. Alguns autores utilizam mais de um método para

Figura 3.11: *Nearest Neighbor Interchange*. Figura 3.12: *Subtree Pruning and Regrafting*.

a resolução de seu problema, fazendo um comparativo para verificar qual oferece melhor desempenho ou produz uma filogenia mais consistente para aquele problema.

Se o tamanho das seqüências for pequeno, é aconselhável que seja aplicado os métodos de *clustering*. Segundo Sourdís *et al* [64], quando a quantidade de nucleotídeos utilizada e a quantidade de substituição de nucleotídeos forem pequenas, a probabilidade de obter a topologia correta é menor nos métodos de parcimônia do que em métodos de distância.

O método de parcimônia tem sido utilizado com freqüência desde a década de 1960 na análise de características morfológicas, métodos de máxima verossimilhança possuem uma base teórica forte, enquanto métodos de distância possuem procedimentos mais adequados para avaliar a confiabilidade das topologias [41].

Quando a quantidade de seqüências é grande, torna-se inviável a aplicação de buscas exaustivas, sendo necessária a recorrência a métodos que minimizem a exploração do espaço de busca. Neste Capítulo foram apresentados vários métodos para este fim. Pes-

Figura 3.13: *Tree Bisection and Reconnection*.

quisadores têm realizado uma mescla de algoritmos na tentativa de explorar o melhor de cada método para tirar conclusões.

Computação Evolutiva

A Computação Evolutiva (CE) é uma técnica baseada em um paradigma inspirado na evolução natural formalizada por Darwin (ver Seção 2.2), tendo sido aplicada principalmente à resolução de problemas de otimização. Uma das vantagens da CE é que seus algoritmos são constituídos por passos genéricos e adaptáveis, capazes de resolver um grande número de problemas, necessitando apenas de poucas alterações.

4.1 Os Algoritmos Evolutivos

Dois aspectos destacados a seguir são fundamentais para o desempenho de uma abordagem evolutiva:

- codificação dos indivíduos, onde cada indivíduo da população representa um candidato à solução de um problema;
- uma função de *fitness* que indica o valor de aptidão de cada indivíduo, mostrando o quão próximo está este valor da solução procurada, dentro do contexto de cada problema.

A representação das soluções corresponde a uma descrição de cada indivíduo da população por meio de uma lista ordenada (denominado cromossomo) ou árvore de atributos.

descritos a partir de um alfabeto finito. Cada atributo da lista ou árvore é equivalente a um gene, onde os possíveis valores que um determinado gene pode assumir são denominados alelos e a posição do cromossomo em que estes aparecem é chamada de locus. Estas soluções são avaliadas por algum critério, produzindo valores (ou *fitness*) que indicam o quanto cada uma das soluções estão adequadas ao problema. Os valores obtidos na função de *fitness* são utilizados na seleção de indivíduos para reprodução (geração de novas soluções com base em soluções anteriores), simulando o processo de seleção natural de Darwin. Operadores de reprodução de recombinação (ou *crossover*) e mutação aplicados à lista ordenada ou árvore de atributos simulam, respectivamente, o cruzamento e a mutação que ocorrem nos seres vivos.

Em geral, as técnicas de CE têm em comum a manutenção de uma população de soluções potenciais, a utilização de processos de seleção baseados no valor de *fitness* de um indivíduo e o emprego de operadores para produção de novos indivíduos. Diversas abordagens para sistemas baseados em evolução têm sido propostas, sendo que as principais diferenças estão relacionadas com a representação da solução, o método de seleção e operadores utilizados. No entanto, existem outros elementos que diferenciam as abordagens evolutivas tais como: a estrutura de dados utilizada para codificar um indivíduo e os métodos utilizados na geração da população inicial. Por outro lado, elas compartilham o princípio comum de que uma população de indivíduos é transformada ao longo das iterações e a competição dos indivíduos pela sobrevivência produz a evolução da população.

As abordagens mais frequentemente utilizadas têm sido: os algoritmos genéticos, as estratégias evolutivas, a programação evolutiva e a programação genética. As primeiras idéias envolvendo algoritmos genéticos podem ser encontradas em trabalhos de John Holland na década de 1960 [11]. Os algoritmos genéticos foram desenvolvidos com o propósito de formalizar matematicamente e explicar processos de adaptação em sistemas naturais e desenvolver sistemas artificiais (em computador) buscando simular os mecanismos originais encontrados em sistemas naturais [66]. Os algoritmos genéticos caracterizam-se pelo emprego dos operadores de *crossover* e mutação e um “*string*” (denominado cromossomo) como estrutura de dados para representar o indivíduo. Na Seção 4.2 é detalhado o funcionamento dos algoritmos genéticos.

A programação evolutiva foi originalmente definida por Fogel *et al* [23], como uma técnica para criar inteligência artificial por meio da evolução de máquinas de estado finito [11]. A programação evolutiva emprega apenas mutação.

Na década de 1960, também foram propostas as estratégias evolutivas com o objetivo de solucionar problemas de otimização de parâmetros [11]. Em virtude de empregarem apenas operadores de mutação, grandes contribuições em relação à análise e síntese destes operadores foram elaboradas em relação à proposta inicial [66]. As estratégias evolutivas, em geral, focam a relação entre espécies e não entre indivíduos de uma espécie. Neste contexto, o operador de recombinação entre indivíduos não é relevante.

A programação genética é uma extensão dos AGs e foi desenvolvida com o intuito de evoluir um conjunto de programas de computador, ao invés de um *string* de bits. Sua utilização tem sido estendida a problemas de diversas áreas do conhecimento como, por exemplo, biotecnologia, engenharia elétrica, análises financeiras, processamento de imagens, reconhecimento de padrões, mineração de dados, linguagem natural, dentre muitas outras [66].

Apesar dessas abordagens terem sido desenvolvidas de forma independente, seus algoritmos possuem uma estrutura comum [42]. O termo algoritmo evolutivo será utilizado como uma denominação comum para essas abordagens. A estrutura básica de um algoritmo evolutivo é apresentada pelo Algoritmo 3.

Algoritmo 3 - Algoritmo evolutivo [42].

```

1:  $t \leftarrow 0$ ;
2: inicialize  $P(t)$ ;
3: avalie  $P(t)$ ;
4: while (not(condição de parada)) do
5:   altere  $P(t)$ ;
6:   avalie  $P(t)$ ;
7:    $t \leftarrow t + 1$ ;
8:   selecione  $P(t)$  a partir de  $P(t - 1)$ ;
9: end while

```

Um algoritmo evolutivo mantém uma **população** de indivíduos $P(t)$ na geração t . Cada indivíduo representa um candidato à solução do problema em estudo e é avaliado por meio de uma **função de fitness**. Em seguida, os indivíduos mais adaptados são passados por um processo de **seleção** para formar a população da próxima geração. Parte dos indivíduos são submetidos a um processo de alteração por meio de operadores de reprodução: mutação e/ou *crossover*, para formar novas soluções. A **mutação** cria novos indivíduos por meio de pequenas modificações de atributos em um indivíduo. O

crossover gera novos indivíduos pela combinação de dois ou mais indivíduos. A cada geração verifica-se a condição de parada do algoritmo. Esta condição geralmente indica a existência na população de um indivíduo que represente uma solução aceitável para o problema ou se o número máximo de gerações foi atingido.

4.2 Algoritmos Genéticos

Os algoritmos genéticos (AGs) são algoritmos de otimização global, baseados nos mecanismos de seleção natural e também da genética. Um indivíduo da população é representado por um único cromossomo, o qual contém a codificação (genótipo) de uma possível solução do problema (fenótipo). Cromossomos são geralmente implementados na forma de listas de atributos ou *arrays*, onde cada atributo é conhecido como gene. Os possíveis valores que um determinado gene pode assumir são denominados alelos. Os indivíduos na população são então submetidos a um processo de evolução que, no AG, corresponde a um procedimento de busca em um espaço de possíveis soluções para o problema.

Nas próximas Seções são detalhados os principais componentes de um algoritmo genético.

4.2.1 Codificação do Problema

O primeiro aspecto a ser considerado para implementar um AG para problemas de busca ou otimização é a representação do problema, isto é, fazer um mapeamento entre o espaço de soluções e o espaço de codificação de modo a preservar o significado do problema original. Além disso, a definição inadequada da codificação pode levar a problemas de convergência, geração de soluções infactíveis e operadores de reprodução computacionalmente ineficientes.

A representação de um problema é codificada em uma estrutura de cromossomo que deve representar uma solução como um todo e ser a mais simples possível. O cromossomo pode ser representado por números inteiros, reais ou binários.

Em problemas de otimização com restrições, a codificação adotada pode fazer com que indivíduos modificados por *crossover*/mutação sejam inválidos. Nestes casos, cuidados especiais devem ser tomados na definição da codificação e/ou dos operadores que utilizam essa codificação. No Capítulo 5 são descritos codificações específicas para problemas envolvendo grafos (árvores).

4.2.2 Definição da População Inicial

O conjunto de indivíduos candidatos a possíveis soluções é denominado população. Sobre esta população ocorre todo o processo evolutivo.

O método mais comum utilizado na criação da população é a inicialização aleatória dos indivíduos dentro do espaço solução [66]. Se houver algum conhecimento prévio a respeito do problema, isso pode ser utilizado na inicialização da população. Tal estratégia é chamada de *seeding*.

Em problemas com restrições, deve-se evitar a geração de indivíduos inviáveis na etapa de inicialização. Por exemplo: no problema da árvore geradora mínima, a solução deve representar apenas árvores, não um grafo qualquer.

4.2.3 Operadores de Reprodução

O principal objetivo dos operadores de reprodução consiste em transformar a população por meio de sucessivas gerações, estendendo a busca até alcançar um resultado satisfatório. Os operadores de reprodução chamados de operadores genéticos nos AGs são essenciais na introdução da variabilidade da população e manutenção de características de adaptação adquiridas pelas gerações anteriores.

Crossover

O operador de *crossover* cria novos indivíduos por meio da combinação de dois ou mais indivíduos de acordo com uma certa probabilidade de cruzamento (taxa de *crossover*).

O operador de *crossover* pode ser empregado de várias maneiras:

- Um-ponto: um ponto de cruzamento no cromossomo é escolhido e a partir deste ponto as informações genéticas dos pais são trocadas. As informações anteriores a este ponto em um dos pais são ligadas a informações posteriores do outro pai;
- Multi-pontos: é uma generalização da idéia anterior, onde muitos pontos de cruzamento podem ser utilizados;
- Uniforme: a cada posição no cromossomo é associado aleatoriamente o valor 0 ou 1. Se for 0, o gene do cromossomo filho recebe o gene do primeiro pai, caso contrário, do segundo.

O operador mais comumente empregado é o *crossover* de um-ponto. Para a aplicação deste operador, dois indivíduos (pais) são seleccionados, como ilustrado na Figura 4.1(a) e aleatoriamente é escolhido um ponto de corte nos pais e os segmentos de cromossomos criados a partir do ponto de corte são trocados, formando os indivíduos filhos (ver Figura 4.1(b)).

(a) Representação de dois indivíduos.

(b) Criação de dois novos indivíduos.

Figura 4.1: Ilustração do funcionamento do operador de *crossover* de um-ponto.

Mutação

O operador de mutação modifica aleatoriamente um ou mais genes de um cromossomo. A probabilidade de ocorrência de mutação em um gene é denominada taxa de mutação. Geralmente, são atribuídos valores pequenos para a taxa de mutação. A idéia intuitiva por trás do operador de mutação é criar uma diversidade extra na população, de forma a evitar que as soluções do algoritmo concentrem-se em um máximo (mínimo) local, permitindo assim, a exploração de novas regiões no espaço de soluções.

Considerando a codificação binária, o operador de mutação padrão simplesmente inverte o valor de um *bit* em um cromossomo. Assim, se o bit seleccionado para mutação tem valor 1, o seu valor passará a ser 0 após a aplicação da mutação e vice-versa, como ilustrado nas Figuras 4.2(a) e 4.2(b).

4.2.4 Seleção

Dada uma população em que a cada indivíduo foi atribuído um valor de *fitness*, existe vários métodos para seleccionar os indivíduos sobre os quais serão aplicados os operadores

(a) Operador de mutação a ser submetido em uma posição do cromossomo.

(b) Indivíduo após a aplicação do operador de mutação.

Figura 4.2: Ilustração do funcionamento do operador de mutação.

de reprodução. Há diversas formas de seleção, dentre essas destacam-se os métodos de seleção por roleta e por torneio. Uma outra estratégia de seleção é por ranqueamento.

No método de seleção por roleta, cada indivíduo da população é representado na roleta proporcionalmente ao seu índice de aptidão [26, 42, 43]. Desta forma, aos indivíduos com alta aptidão é dada uma porção maior da roleta, enquanto que os indivíduos de aptidão mais baixa recebem uma porção relativamente menor. De acordo com Goldberg [26], o método da roleta pode ser descrito pelo Algoritmo 4.

Algoritmo 4 - Método da Roleta - adaptado do livro de Goldberg [26].

- 1: $T \leftarrow$ soma dos valores de aptidão de todos os indivíduos da população;
 - 2: $i \leftarrow 1$;
 - 3: $S \leftarrow 0$;
 - 4: $r \leftarrow$ valor aleatório no intervalo entre 0 e T ;
 - 5: **repeat**
 - 6: $S \leftarrow S +$ valor de *fitness* do indivíduo i ;
 - 7: $i \leftarrow i + 1$;
 - 8: **until** ($S \geq r$ **or** $i =$ tamanho da população);
 - 9: **return** i ;
-

Na Figura 4.3 é apresentado um exemplo de seleção por roleta. Supondo $r = 39$, o algoritmo itera o valor de S , iniciando com zero e adicionando o valor de *fitness* do indivíduo 1 ($S = 18$). Em seguida é acrescentada a aptidão do indivíduo 2 ($S = 25$) e, assim sucessivamente, até que $S = 60 > 39$ e o algoritmo retorna $i = 5$.

Um outro método é a seleção por torneio, onde um número k de indivíduos da po-

Figura 4.3: Ilustração do método da roleta.

pulação é escolhido aleatoriamente. Deste grupo, de acordo com uma probabilidade p , o indivíduo que apresentar o maior valor de *fitness* é selecionado. A seleção por torneio para $n = 2$ pode ser implementada pelo Algoritmo 5 [43].

A seleção por torneio tende a escolher indivíduos melhores do que pela seleção por roleta, uma vez que a seleção por torneio seleciona um indivíduo com aptidão melhor de uma subpopulação.

Na seleção por ranqueamento, os indivíduos são ordenados de acordo com seu valor de *fitness*, onde os melhores indivíduos são selecionados.

Algoritmo 5 - Seleção por Torneio - adaptado do livro de Mitchell [43].

- 1: $k \leftarrow 0.75$;
 - 2: Escolha 2 indivíduos da população aleatoriamente;
 - 3: $r \leftarrow$ valor aleatório entre 0 e 1;
 - 4: **if** ($r < k$) **then**
 - 5: O indivíduo com maior aptidão, dentre os dois indivíduos selecionados anteriormente, é escolhido;
 - 6: **else**
 - 7: O indivíduo com pior aptidão é selecionado;
 - 8: **end if**
-

Elitismo

O método de seleção mais empregado para acelerar a convergência dos AGs é o Elitismo [43], que retém um certo número de “melhores” indivíduos a cada geração e os

conservam para a próxima iteração. Tais indivíduos podem ser perdidos se não forem selecionados para reprodução ou se forem destruídos por cruzamento ou mutação. Muitos pesquisadores têm encontrado no elitismo vantagens significativas para o aumento do desempenho dos AGs [43].

O Elitismo consiste basicamente em selecionar uma elite de E membros (aqueles indivíduos que tiverem maiores aptidões) a ser incorporada diretamente aos indivíduos da próxima geração. Geralmente, a elite tem um tamanho reduzido, 1 a 2 indivíduos para cada 50 indivíduos. Sua amostragem pode ser direta (escolhendo-se simplesmente os melhores) ou por sorteio (escolhendo-se os melhores entre os melhores da população).

O elitismo é comumente utilizado em algoritmos genéticos *steady-state*, onde os indivíduos criados são utilizados para modificar a população inteira. O elitismo é introduzido na comparação de dois indivíduos filhos com dois indivíduos pais, mantendo os dois indivíduos de maior valor de *fitness* na população [10].

4.2.5 Função de Fitness

A cada indivíduo da população é atribuído um valor de aptidão determinado por uma função de *fitness* que mede o quão adequada é a solução em análise. Tal função deve ser muito representativa e diferenciar na proporção correta as soluções inadequadas das mais apropriadas.

4.2.6 A Escolha de Parâmetros

Além da forma como o cromossomo é codificado, existem vários parâmetros do AG que podem ser escolhidos para melhorar o seu desempenho, adaptando o AG às características particulares de determinadas classes de problemas. Entre os parâmetros, os mais importantes são: o tamanho da população, a taxa de substituição do indivíduo da população por novos indivíduos, a taxa de *crossover* e a taxa de mutação.

Tamanho da População: o tamanho da população afeta o desempenho global e a eficiência dos AGs. Com uma população pequena o desempenho pode ser baixo, pois deste modo a população fornece uma pequena cobertura do espaço de busca do problema. Uma população grande geralmente fornece uma cobertura representativa do domínio do problema. No entanto, para se trabalhar com grandes populações são necessários maiores recursos computacionais e que o algoritmo requeira tempo de computação muito grande.

Taxa de *Crossover*: quanto maior for esta taxa, mais rapidamente novas informações serão introduzidas na população. No entanto, isto pode gerar um efeito inapropriado pois, a maior parte da população será substituída podendo ocorrer a perda de informações responsáveis pela alta aptidão. Com um valor baixo, o algoritmo pode tornar-se muito lento.

Taxa de Mutação: uma taxa de mutação baixa previne que o algoritmo fique estagnado em um ponto no espaço de busca. Com uma taxa muito alta, a busca torna-se praticamente aleatória, aumentando a possibilidade de não encontrar uma solução adequada.

Taxa de Substituição: indica a porcentagem da população que será substituída na próxima geração. Se o valor desta taxa for muito alto, pode ocorrer perda de estruturas de alta aptidão. Com um valor baixo, a convergência do algoritmo pode tornar-se lenta.

A influência de cada parâmetro no desempenho do algoritmo depende da classe de problemas que está sendo considerada. Assim, a determinação de um conjunto de valores otimizado para estes parâmetros depende da realização de um grande número de testes.

Os valores geralmente encontrados na literatura estão na faixa de 60% a 65% para a probabilidade de *crossover* e entre 0,1 e 5% para a probabilidade de mutação. O tamanho da população e o número de gerações dependem da complexidade do problema de otimização e devem ser determinados experimentalmente. No entanto, deve ser observado que o tamanho da população e o número de gerações definem diretamente o tamanho da cobertura do espaço de busca.

Existem estudos que utilizam um AG como método de otimização para a escolha dos parâmetros de outro AG (meta-AGs), devido à importância da escolha correta destes parâmetros ou em que os valores para tais parâmetros serão ajustados para o próprio AG como parte dos genes em evolução (AGs auto-adaptativos).

4.3 Exemplos de Aplicações de Algoritmos Evolutivos

Os AEs possuem uma larga aplicação em muitas áreas científicas, dentre as quais podem ser citados problemas de otimização, aprendizado de máquina, desenvolvimento de estratégias e fórmulas matemáticas, análise de modelos econômicos, problemas de engenharia, diversas aplicações na Biologia, etc. [44].

Nas Seções seguintes é descrito o problema da árvore geradora mínima com restrição de grau e mostrados trabalhos envolvendo AEs para filogenia. Esses e outros problemas

envolvendo árvores geradoras têm sido extensivamente utilizados em uma variedade de problemas de projeto de redes [25].

4.3.1 Árvore Geradora Mínima com Restrição de Grau

O problema da árvore geradora mínima (*MST* - do inglês: *Minimum Spanning Tree*) foi inicialmente formulado em 1926 por Boruvka [28]. Desde então a formulação do problema de *MST* tem sido aplicado em muitos problemas de otimização combinatorial tais como: problema de transporte, desenvolvimento de redes de telecomunicações, sistemas distribuídos e assim por diante. Alguns algoritmos em tempo polinomial para resolver o problema de *MST* foram desenvolvidos por Kruskal, Prim, Dijkstra e Sollin [25].

As extensões do problema da árvore geradora mínima, como por exemplo, árvore geradora mínima com restrição de grau, árvore geradora mínima generalizada, árvore geradora mínima probabilística, entre outros, são problemas *NP*-difíceis, para os quais não se conhecem algoritmos de ordem polinomial que os resolvam.

O problema da árvore geradora mínima com restrição de grau (*dc-MST*, do inglês: *degree constrained Minimum Spanning Tree*) é bastante estudado [34] e de grande importância em desenvolvimento de redes de comunicações e circuitos integrados [55]. Devido a esse problema ser *NP*-difícil, AEs têm sido cada vez mais aplicados na busca de soluções ótimas ou sub-ótimas [34, 55, 25, 27].

Descrição do problema de *dc-MST*

Dado $G = (V, E)$, um grafo conexo e não-orientado onde $V = \{v_1, v_2, \dots, v_n\}$ é o conjunto de nós de G e $E = \{e_1, e_2, \dots, e_m\}$ é o conjunto de arestas de G . O conjunto $W = \{w_1, w_2, \dots, w_m\}$ representa o custo (ou peso) de cada aresta onde este é estritamente um número real positivo.

Uma árvore geradora é um conjunto mínimo de arestas de E que conectam todos os nós em V . Pelo menos uma árvore geradora pode ser encontrada no grafo G , uma vez que o grafo é conexo.

A árvore geradora mínima com restrição de grau, denotada por T_{dc}^* , é a árvore geradora cuja soma total dos pesos das arestas é mínimo e todos os nós possuem grau limitado.

O problema pode ser definido da seguinte maneira:

$$T_{dc}^* = \min_T \sum_{e_{ij} \in E, d_i \leq b_i} w_{ij}$$

onde para cada nó v_i , o grau d_i é no máximo b_i e $d_i, b_i \geq 1$.

Por conveniência, em geral, considera-se apenas o caso simétrico em um grafo completo, ou seja, $w_{ij} = w_{ji}$ e os limites de graus $b_i = b$ para todo $v_i \in V$.

4.3.2 Algoritmos Evolutivos Aplicados à Filogenia

Cada vez mais os algoritmos evolutivos têm sido aplicados ao problema de filogenia. As vantagens em se utilizar tais algoritmos está principalmente em sua tendência em evitar ótimos locais e apresentar desempenho superior em relação a outras estratégias de busca global. A seguir são descritos os trabalhos utilizando AEs associados ao método de máxima verossimilhança e parcimônia.

Matsuda [40]

O algoritmo proposto por Matsuda [40] empregou a representação baseada em grafos. A população foi inicializada aleatoriamente reproduzindo um número fixo de árvores (selecionadas pelo método da roleta) por meio dos operadores de *crossover* e mutação. Como tais operadores podem alterar a melhor solução a cada geração, utilizou-se o elitismo.

O operador de mutação empregou a estratégia de troca de ramos. O operador de *crossover* foi baseado no princípio de evolução mínima e pode ser descrito como a seguir:

- Selecione duas árvores i e j aleatoriamente;
- Calcule o tamanho dos ramos;
- Calcule a diferença relativa entre cada dois nós x e y nas árvores i e j da seguinte maneira:

$$r_{(ij)}(x, y) = \frac{|d_i(x, y) - d_j(x, y)|}{d_i(x, y) + d_j(x, y)}$$

onde $d_i(x, y)$ denota a distância entre x e y na árvore i enquanto $d_j(x, y)$ a distância entre x e y na árvore j ;

- Encontre o par de táxons (X, Y) com a maior diferença relativa. Se $d_i(X, Y) < d_j(X, Y)$ então i é denominada árvore referência. Caso contrário, se $d_j(X, Y) < d_i(X, Y)$ então j é considerada a árvore referência;
- Combine i e j da seguinte maneira:

- Escolha um nó m da árvore referência que não pertença à sub-árvore mínima contendo os táxons X e Y ;
- Mantenha m e a sub-árvore mínima contendo X e Y . Remova todos os outros táxons da árvore referência. Seja s , este resultado;
- Remova todos os táxons que aparecem em s na outra árvore, exceto m (caso m pertença à s). Denomine t este resultado;
- Combine s e t , sobrepondo m em s e t .

Os testes foram realizados para 15 seqüências de proteínas. Para todos os testes, com uma população de 20 árvores, após 130 gerações encontrou-se uma aptidão x , de forma que, após adicionais 8 gerações, todas as 20 árvores haviam alcançado o mesmo valor x . Em comparação aos métodos *UPGMA*, *NJ*, Máxima Parcimônia e Máxima Verossimilhança com diferentes algoritmos de busca, o algoritmo genético mostrou melhor desempenho. Como trabalho futuro, o autor sugeriu que seja construído um método para determinar os melhores parâmetros do algoritmo genético.

Lewis [38]

Lewis [38] desenvolveu o programa *GAML* (*Genetic Algorithm for Maximum Likelihood phylogeny inference*). O autor justificou a aplicação dos algoritmos genéticos na busca filogenética, devido à sua habilidade em encontrar soluções ótimas ou sub-ótimas em tempos razoáveis de computação, mesmo utilizando métodos complexos de evolução tal como máxima verossimilhança, especialmente quando o número de instâncias é grande.

A população foi inicializada aleatoriamente e os comprimentos dos ramos foram definidos com um valor arbitrário. Durante cada geração do algoritmo genético foi calculado o *fitness* que corresponde ao logaritmo da verossimilhança¹ ($\ln L$) de cada árvore. Não foi realizada nenhuma otimização de ramos durante este cálculo.

A população (de tamanho n) foi ordenada de acordo com o *fitness* de cada árvore. O indivíduo que possuísse o maior *fitness* gerava k cópias para a próxima geração. Os restantes $n - k$ indivíduos resultaram de cópias de um indivíduo escolhido aleatoriamente. A esta população denominou-se população filha e a outra de população pai. Todos os indivíduos da população filha foram submetidos a alteração no tamanho dos ramos e aos operadores de mutação e *crossover*.

¹Tal verossimilhança foi calculada assumindo *HKY85* como modelo de evolução.

A alteração no tamanho dos ramos, apesar de ser realizada em todos os indivíduos, não atingiu todos os ramos de uma árvore. O tamanho foi modificado de acordo com a distribuição gamma. A mutação aplicada corresponde ao método *SPR* (ver Seção 3.9.2). A única diferença consiste no fato de apenas uma sub-árvore ser podada aleatoriamente e enxertada em uma posição definida também aleatoriamente.

Após um indivíduo pai T_1 ter sido escolhido e gerado um indivíduo filho, a recombinação foi realizada com uma determinada probabilidade onde um segundo indivíduo pai T_2 foi escolhido aleatoriamente da população pai. É importante ressaltar que os indivíduos da população filha sofreram alteração no tamanho de ramos possibilitando esta operação. A recombinação aconteceu da seguinte maneira:

- Uma sub-árvore S_1 foi escolhida aleatoriamente para ser removida de T_1 ;
- Em T_2 , os táxons pertencentes à S_1 foram removidos, resultando na sub-árvore S_2 ;
- A sub-árvore S_1 foi inserida em S_2 , resultando na nova árvore recombinada.

Os testes foram realizados para seqüências de nucleotídeos, mostrando desempenho consideravelmente melhor em relação ao programa *PAUP* para um conjunto de 55 seqüências de *DNA*. Neste trabalho, o autor comentou a possibilidade de distribuir as tarefas pelos processadores, o que ocorreu posteriormente em [3], onde tal algoritmo foi paralelizado e testado com instâncias maiores de 200 táxons. Neste algoritmo foi realizada a otimização de ramos.

Skourikhine [63]

Skourikhine [63] desenvolveu um algoritmo genético auto-adaptativo utilizando o método de máxima verossimilhança com o modelo de evolução *HKY85*. Nos AGs auto-adaptativos, os parâmetros evoluem junto com a população. Os parâmetros utilizados foram: probabilidade de mutação no tamanho dos ramos da árvore, probabilidade de mutação na topologia e probabilidade de mutação de um parâmetro do modelo *HKY85*.

Os testes foram realizados para um conjunto de 55 espécies. Os resultados obtidos foram comparados com programas do pacote *PHYLIP* e as árvores encontradas foram equivalentes.

Prado e Von Zuben [51, 52]

Prado e Von Zuben [51, 52] desenvolveram um software denominado *PTP* (do inglês: *Phylogenetic Tree Project*) com duas opções de reconstrução filogenética com matriz de distâncias e máxima verossimilhança utilizando árvores enraizadas.

A codificação da população foi baseada em árvores representadas por uma matriz de adjacências como ilustra a Figura 4.4.

Figura 4.4: Árvore com 9 nós e sua matriz de adjacências.

Para exploração do espaço de busca foi desenvolvido três operadores de mutação onde podem ser realizadas mutação entre folhas, entre ramos e entre folhas e ramos. Todas essas mutações consistem na troca de posições de folhas e/ou ramos na árvore.

Em *PTP* foi implementada uma condição de parada baseada na topologia das três melhores candidatas. Segundo os autores, quando estas forem iguais considera-se que o algoritmo não alcançará melhores resultados. Simulações mostraram que este critério sempre foi suficiente para encontrar uma boa solução. Até mesmo quando valores maiores do que três foram utilizados nenhuma melhoria foi atingida.

O autor realizou testes para pequenas quantidade de seqüências de bases comparando com *PHYLP* [20] e *PAML* [70]. Os resultados ficaram entre um e outro, mas Prado [51] concluiu que não haveria como fazer comparações diretas.

Katoh et al [32]

Katoh et al [32] aplicaram algoritmo genético com o método de máxima verossimilhança para seqüências de proteínas, sendo este denominado *GA-mt* (do inglês: *Genetic Algorithm-based ML method that outputs multiple trees*) devido ao fato de buscar não somente a árvore ótima, mas também aquelas próximas do ótimo.

A inicialização da população foi realizada segundo o algoritmo de *Neighbor Joining* e a mutação ocorreu com o método *Tree Bisection and Reconnection (TBR)* entre dois ramos de uma árvore, selecionados aleatoriamente.

Duas árvores selecionadas de acordo com o *fitness* trocam suas partes entre si durante o processo de *crossover* (ver Figuras 4.5(a) e 4.5(b)).

(a) Representação de dois indivíduos.

(b) Criação de dois novos indivíduos.

Figura 4.5: Aplicação do operador de *crossover* utilizado em *GA-mt*.

GA-mt mostrou-se bastante eficiente na busca de árvores próximas do ótimo, apresentando tempo de computação 1% daquele requerido pelo método de máxima verossimilhança autêntico. Para comparação com métodos de distância e outras heurísticas utilizou-se um índice C definido como $C = P_C/P_m$, onde P_m é o número de partições na árvore considerada como de solução ótima e P_C é o número de partições que são idênticas na árvore reconstruída com a árvore de solução ótima. O índice C diminui com o aumento

do número de táxons enquanto que para o *GA-ml* aumenta ou permanece constante com o aumento do número de táxons.

Em relação às heurísticas comparadas, chegou-se à conclusão de que o tempo de computação requerido para alcançar um nível ótimo para *C* depende fortemente das árvores iniciais.

Cotta et al [7]

Cotta *et al* [7] compararam duas estratégias: técnica direta, que faz busca direta por todas as soluções e técnica indireta, na qual é necessário um mapeamento do espaço de busca e um decodificador. Na técnica direta é utilizada uma codificação e o operador de recombinação funciona da seguinte maneira: dados dois pais P_1 e P_2 , uma sub-árvore é escolhida de P_1 e em P_2 é realizada uma busca para identificar onde estão os nós que pertencem à P_1 e então são removidos e a sub-árvore é inserida em P_1 .

Para a operação de mutação três estratégias foram utilizadas:

- duas *OTUs* são selecionadas e trocadas de posições;
- duas folhas vizinhas são trocadas;
- uma sub-árvore é selecionada aleatoriamente de uma árvore e rearranjada aleatoriamente.

Na técnica indireta, as *OTUs* são organizadas numa seqüência e inseridas uma de cada vez. Seja $S = \{o_1, \dots, o_n\}$ a seqüência de *OTUs* e S_{aux} , o espaço de busca auxiliar definido por $S_{aux} = \prod_{i=1}^{n-2} N_{2i}$, onde $N_k = \{1, \dots, k\}$. O algoritmo para decodificação pode ser dado da seguinte maneira:

- Seja T uma árvore contendo h como raiz e o_1 e o_2 como nós folha;
- Para cada $i \in [1, \dots, n - 2]$:

Numere de 1 a $2i$ os ramos de T . Seja s_i o ramo que une h à sub-árvore U ;

Substitua U por V em T , onde V é uma sub-árvore que contém U e o_{i+2} como nós folha e h' , onde h' é o novo nó interno.

Um exemplo de decodificação pode ser visto na Figura 4.6.

Contudo, tal técnica possui a desvantagem no fato de ter a ordem de cada adição de ramos fixada por meio de uma seqüência que determinará a topologia da árvore. Uma

(a) $n = 4$ e $S = \{o_1, o_2, o_3, o_4\}$. (b) para $i = 1$. (c) para $i = 2$.Figura 4.6: Decodificação para uma sequência de 4 *OTUs*.

versão deste decodificador foi implementada de maneira que este encontra o ponto mais adequado de inserção para cada táxon. Este decodificador foi denominado decodificador permutacional.

O método de “ranqueamento” foi utilizado juntamente com o elitismo durante a seleção. Os parâmetros do algoritmo foram fixados e nenhum refinamento foi tentado. Duas instâncias foram testadas.

O algoritmo que utilizou a técnica direta mostrou-se superior em relação ao do segundo grupo a não ser quando foi testado decodificador permutacional. Este método forneceu de forma consistente soluções melhores a um baixo custo computacional. No entanto, tal método não oferece escalabilidade quando o número de *OTUs* aumenta.

Lemmon et al [36]

Lemmon *et al* [36] na tentativa de lidar com grandes instâncias, utilizaram metapopulações no algoritmo genético denominado *METAPIGA* (do inglês: *phylogeny inference using meta-population genetic algorithm*) com a estratégia de árvore de consenso (ver

Seção 3.7) em seus operadores, preservando sub-árvores com melhor custo. Nesta estratégia, duas ou mais populações interagem na busca da melhor solução. A comunicação entre as populações é realizada por *consensus pruning* (*CP*), onde as partes consenso com as melhores árvores de outras populações são preservadas durante o processo de mutação (ver Figura 4.7). Durante a mutação foram utilizados os métodos *SPR*, *NNI*, troca de táxons e troca de sub-árvores. Entre ramos de consenso não é permitida nenhuma modificação.

Figura 4.7: Ilustração de *consensus pruning*.

O operador de recombinação é baseado no operador implementado no programa *GAML* [38] descrito anteriormente. Este operador envolve a identificação de uma região de consenso entre duas árvores de populações diferentes, seguida pela troca de sub-árvores definida pela região de consenso. Por definição, este procedimento não necessita de qualquer poda de táxons devido ao fato de que o ramo de consenso define em ambas árvores as mesmas partições.

Dois critérios de parada foram implementados:

- as melhores árvores de todas as populações possuem a mesma topologia;
- todas possíveis modificações topológicas permitidas foram realizadas pelo último consenso entre as melhores árvores de todas populações e tais alterações não melhorou a aptidão das soluções.

Os autores chegaram à conclusão de que sob *CP*, aumentando o número de populações diminui a interdependência entre as populações dentro de uma única rodada. Em relação ao tamanho da população, pequenas populações produzem resultados com mais rapidez e acurácia, independente do tamanho do conjunto de dados.

O *METAPIGA* mostrou ser sete vezes mais rápido que *PAUP* quando utilizou o modelo de evolução *JC69*. Ao considerar o modelo *HKY85*, esta vantagem subiu para 800.

Moilanen [45]

Moilanen [45] desenvolveu um programa utilizando o método da parcimônia denominado *Parsigal* (do inglês: *Parsimony analysis using a genetic algorithm*). Tal programa combina busca local com AEs. A busca local foi implementada com *NNI* e *SPR*. A utilidade da busca local no problema estudado pode ser vista da seguinte forma: quando uma sub-árvore ótima é examinada, é geralmente óbvio que a árvore possa ser melhorada com um pequeno rearranjo de nós.

A função de *fitness* utilizada foi a parcimônia de Wagner [18] para caracteres binários. A população inicial é gerada por métodos de *clustering UPGMA* e *WPGMA*. A seleção da população é feita pelo método da roleta utilizando a estratégia elitista. O operador de recombinação troca sub-árvores entre dois indivíduos pai. Uma sub-árvore é escolhida aleatoriamente em cada pai e enxertada em outro pai em uma posição aleatória. Se acontecer repetição de táxons, aquele que pertencer à sub-árvore é eliminado.

O autor analisou o desempenho do *Parsigal* também em relação a outros softwares que aplicam a parcimônia e concluiu que o desempenho foi razoavelmente bom, apesar de não poder fazer uma comparação direta, uma vez que seu programa encontra apenas uma árvore.

O efeito do procedimento de inicialização também foi analisado e verificou-se que o algoritmo evolutivo é pouco dependente deste método. Além disso, foi ressaltada a importância de uma computação eficiente do conjunto de estados e comprimento dos ramos para caracteres binários, onde a cada mudança de ramos na árvore é preciso refazer os cálculos, afetando o desempenho global do programa.

Outro teste foi realizado variando-se o tamanho da população e o número de gerações e analisando seus efeitos na combinação do AE com busca local.

Congdon et al [5]

Para melhorar o desempenho do *PHYLIP* [20] quando o número de instâncias é grande, foram aplicados os algoritmos genéticos por Congdon *et al* [5] resultando no programa *GAPhyI*, capaz de encontrar um maior número de árvores parcimoniosas no mesmo tempo de computação comparado ao *PHYLIP*.

O operador de recombinação funciona da seguinte forma:

- Dadas duas árvores T_1 e T_2 , uma espécie é escolhida aleatoriamente em T_1 . Uma sub-árvore s que inclui este nó (excluindo a escolha da sub-árvore que contém só este nó e a árvore inteira) é removida de T_1 ;
- Em T_2 , encontre a menor sub-árvore t que contém todas as espécies pertencentes a sub-árvore s ;
- Substitua a sub-árvore s por t em P_1 e remova qualquer espécie que estiver duplicada;
- Repita o processo trocando-se os papéis dos pais, o que resultará em duas árvores filha.

Um exemplo do funcionamento do operador de *crossover* é ilustrado na Figura 4.8. Considere C , a espécie escolhida aleatoriamente em T_1 .

Os autores implementaram dois operadores de mutação. O primeiro, escolhe aleatoriamente duas espécies e trocam suas posições. O segundo operador obtém aleatoriamente uma sub-árvore, rotacionando-a e reinserindo-a na árvore.

Foram realizados testes adicionais com os operadores. Cada teste ajustando a probabilidade de um deles em 0% e depois em 100%, mostrando a contribuição relevante do operador de *crossover* e do segundo operador de mutação.

4.4 Considerações Finais

A CE representa computacionalmente os processos evolutivos da natureza por meio de um conceito muito simples: os indivíduos mais adaptados passam para a próxima geração sofrendo variações como (*crossover* e/ou mutação).

Neste Capítulo, dando enfoque aos AGs, foram descritos seus principais componentes como: a codificação de indivíduos, o *crossover* e a mutação, que representam fatores críticos para o desempenho de um AG. Se estes não forem apropriadamente definidos para cada problema, podem produzir resultados inadequados.

Cada vez mais os AEs vêm sendo utilizados em problemas de otimização combinatorial, apresentando resultados motivadores em problemas envolvendo árvores geradoras [25]. Diferente dos métodos tradicionais, os AEs possuem um conjunto de procedimentos genéricos que podem ser adaptados a cada problema. Além disso, ao invés de considerar uma possível solução por vez, os AEs trabalham com várias soluções adaptáveis, ou seja, não

(a) Representação de dois indivíduos.

(b) Árvore resultante após aplicação do operador de *crossover*.Figura 4.8: Operador de *crossover* implementado para o programa *GaPhyl*.

é necessário reiniciar todo o processo de busca para cada uma, realiza-se apenas alguns ajustes.

Os algoritmos evolutivos têm também apresentado resultados relevantes em problemas envolvendo redes de comunicação, como as extensões do problema de árvore geradora mínima, problema de transportes, filogenia entre outros.

Em especial, na literatura em filogenia, pôde-se observar diversas estratégias para melhorar o desempenho dos algoritmos evolutivos. A maioria associa métodos de busca local como *branch swapping*, justificando tal combinação pelo fato da busca local ser rápida apesar de não evitar ótimos locais. Já no caso dos algoritmos evolutivos, acontece o contrário. Desta forma é possível explorar as melhores características e cada técnica para obter os resultados. Lemmon *et al* [36] mostraram a importância de se trabalhar com meta-populações quando se lida com seqüências maiores. Outra estratégia que per-

mite aumentar significativamente o desempenho de AEs para Filogenia é a paralelização conforme apresentado em [3].

Codificações de Grafos para Algoritmos Evolutivos

Além de ter uma função de *fitness* adequada, uma etapa importante no projeto de AEs para encontrar soluções para um dado problema consiste em decidir quais informações serão armazenadas e como serão codificadas. Este fator torna-se crítico para problemas de Projeto de Redes (PR) como ocorre em filogenia.

As Seções seguintes apresentam diversas propostas de codificação de árvores. Para cada uma são mostrados operadores adequados para solução do problema de PR para árvore geradora mínima com restrição de grau. Esse tipo de árvore possui similaridades com as árvores filogenéticas. Além disso, o problema de *dc-MST* tem sido utilizado também como referência para comparação de abordagens desenvolvidas para problemas de PR.

5.1 Codificações de Árvores

As codificações de árvores são críticas para as abordagens que utilizam AEs, uma vez que qualquer novo indivíduo gerado deve corresponder a uma árvore [50]. De acordo com Palmer [50], um AE eficiente para os problemas que envolvem árvores como soluções, devem possuir as seguintes características de codificação:

- Todas as soluções geradas pelo AE devem ser viáveis, ou seja, devem representar somente árvores;
- Cada cromossomo deve representar uma e somente uma árvore e vice-versa. Não deve existir ambigüidade durante os processos de codificação e decodificação;
- Os operadores de mutação e recombinação e os processos de codificação e decodificação devem ser computacionalmente eficientes para o AE ser aplicado a problemas onde o tamanho da entrada é grande;
- Os operadores de reprodução devem oferecer localidade, isto é, pequenas alterações na representação introduzem pequenas mudanças na árvore. Assim, quando aplica-se o operador, grande parte das características dos indivíduos alterados passam para a próxima geração. Isso ajuda o algoritmo a convergir para uma população com alto valor de *fitness*.

O ideal seria que todas as representações de árvores possuísem essas propriedades, mas, na prática, poucas codificações obedecem essas regras. Nas Seções seguintes são descritas as representações mais tradicionais da literatura e outras que foram recentemente publicadas, assim como os métodos de inicialização, *crossover* e mutação aplicados ao problema de *dc-MST*.

5.1.1 Vetor de Características

A codificação por vetor de características (também chamada de codificação por arestas [25]) é representada por um vetor binário utilizado para indicar se uma possível aresta é utilizada ou não em um grafo [59]. Em um grafo completo (ver Apêndice) com n nós, existem $m = n(n-1)/2$ possíveis arestas. Assim, pode-se utilizar um vetor de características E_T (ver Tabela 5.1) de tamanho m para codificar uma árvore (ver Figura 5.1) com n nós. Cada aresta possível deve ser enumerada e atribuída a uma posição do vetor. Tal posição assume o valor 0 ou 1, dependendo de sua presença na árvore. Há, portanto, $2^{n(n-1)/2}$ possíveis valores para E_T . Contudo, a probabilidade de E_T , criado aleatoriamente, representar uma árvore é extremamente ínfima, da ordem de $2^{-[n(n/2-\log(n))]}$ [49].

A codificação/decodificação pode ser implementada em tempo $O(n^2)$ [49]. Este tempo é decorrente desta representação trabalhar sempre com todas as possíveis arestas do grafo.

Figura 5.1: Árvore com 5 nós.

1	0	0	0	0	1	1	1	0	0
0-1	0-2	0-3	0-4	1-2	1-3	1-4	2-3	2-4	3-4

Tabela 5.1: Representação por vetor de características para a árvore da Figura 5.1.

Vetor de Características Aplicado ao Problema de dc-MST

A inicialização da população pode ser realizada de maneira aleatória com 50% de probabilidade de cada posição assumir os valores 0 ou 1. O operador de *crossover* pode ser aplicado, de acordo com a taxa de cruzamento, em um certo ponto, fazendo assim, a troca de material genético entre dois indivíduos. A mutação consiste em alterar um *bit* (se for 0, passa a ser 1 e vice-versa) em um (ou mais) ponto(s) do cromossomo de acordo com a taxa de mutação.

À medida que aumenta o tamanho do grafo, diminui-se drasticamente a probabilidade de obter uma árvore com os processos descritos acima. Além disso, deve-se obedecer também a restrição de grau, que reduz ainda mais o número de soluções factíveis geradas.

5.1.2 Predecessores

Na representação por predecessores, como o próprio nome já indica, uma árvore é codificada por determinar para cada nó (a partir da raiz r), o seu antecessor. Seja $Pred[i] = j$, onde j é o primeiro nó no caminho de i até r em T . Por convenção, assumamos $Pred[r] = r$. Como exemplo, na Tabela 5.2 é mostrado um vetor representando a árvore da Figura 5.2.

Uma árvore enraizada é representada por uma única sequência de n dígitos, onde os dígitos são números entre 1 e n . Além disso, qualquer árvore pode ser representada por n desses dígitos. Portanto, esta é uma representação que cobre todo o espaço de solução.

O processo de codificação/decodificação, tendo a lista apenas de arestas pertencentes à árvore (espaço $O(n)$), pode ser realizado em tempo $O(n)$ [50].

Figura 5.2: Grafo orientado com 5 nós, onde 1 representa o nó raiz.

1	1	1	2	3
---	---	---	---	---

Tabela 5.2: Representação por predecessores para a árvore da Figura 5.2.

Predecessores Aplicado ao Problema de dc-MST

De maneira análoga à representação por vetor de características, a inicialização, o *crossover* e a mutação podem ser realizados considerando n dígitos ao invés de *bits*.

Essa representação também pode gerar soluções inválidas, onde o cromossomo não representa uma árvore ou não obedece a restrição de grau. Para contornar esse problema, podem ser aplicados mecanismos de penalidade (atribuir aptidão relativamente baixa, por exemplo) e de reparo (fazer ajustes no grafo de modo que passe a representar uma árvore atendendo também a restrição de grau). No entanto, a aplicação desses mecanismos provoca, em geral, o aumento da complexidade computacional dos procedimentos de geração de novos indivíduos e avaliação dos mesmos.

5.1.3 Número de Prüfer

O número de Prüfer é definido da seguinte forma: dada uma árvore T com n nós, o número de Prüfer (P) é um número de $n - 2$ dígitos, onde esses dígitos são números entre 1 e n e são definidos pelo Algoritmo 6.

Considere a Figura 5.3. A partir do grafo, a conversão, passo a passo, para o número de Prüfer é realizada da seguinte maneira: inicialmente, 2 é o nó folha de menor rótulo. O nó adjacente a 2 é adicionado a P ($P = 3$). O nó 2 é então removido junto com a aresta $(2, 3)$. Na próxima iteração, 4 é selecionado como o nó folha de menor rótulo, 3 é colocado em P ($P = 33$). O nó 4 é removido de T . A seguir, 3 é o nó folha de menor rótulo, como 1 é o nó adjacente a 3, este é adicionado a P ($P = 331$). O nó 3 é removido

Algoritmo 6 - Codificação por número de Prüfer [50].

```

1: while (houver mais de dois nós a serem considerados) do
2:    $i \leftarrow$  nó folha de menor rótulo;
3:    $j \leftarrow$  nó adjacente a  $i$ ;
4:   Adicione  $j$  a  $P$ ;
5:   Remova o nó  $i$  e a aresta  $(i, j)$  de  $T$ .
6: end while

```

de T . No próximo passo, 5 é o nó folha de menor rótulo e 1 é nó adjacente a ele, portanto, $P = 3311$ (ver Tabela 5.3). Neste estágio, o grafo possui apenas dois nós e o algoritmo pára.

Figura 5.3: Grafo com 6 nós [50].

3	3	1	1
---	---	---	---

Tabela 5.3: Representação por número de Prüfer para a árvore da Figura 5.3 [50].

O procedimento para a decodificação por número de Prüfer (ver Algoritmo 7) considera P como sendo o número de Prüfer e $\bar{P}$ o conjunto de todos nós não pertencentes a P . Tais nós são denominados elegíveis para posteriores considerações. Enquanto houver pelo menos um dígito em P , o nó de menor rótulo em $\bar{P}$ é atribuído a i , j recebe o dígito mais à esquerda de P , (i, j) é adicionado em T , i é removido de $\bar{P}$ e j de P . Caso não exista mais nenhum j em P , j é colocado em $\bar{P}$. Se não restar mais nenhum dígito em P , há dois nós remanescentes em $\bar{P}$ e a aresta correspondente a esses nós é adicionada à árvore.

É fácil ver que uma das principais desvantagens do número de Prüfer é sua localidade relativamente baixa, uma vez que a mudança de somente um dígito de um número de Prüfer pode mudar drasticamente a árvore resultante [50]. Uma outra desvantagem do número de Prüfer ocorre para grafos esparsos. Neste caso, a geração de árvores válidas

Algoritmo 7 - Decodificação por número de Prüfer [50].

1: **while** (restar pelo menos um dígito em P) **do**

2: $i \leftarrow$ nó elegível com o menor rótulo.

3: $j \leftarrow$ dígito mais a esquerda de P .

4: Adicione a aresta (i, j) em T .

5: Remova i de $\bar{P}$ e j de P .

6: **if** ($j \notin P$) **then**

7: Coloque j em $\bar{P}$.

8: **end if**

9: **end while**

 /* Seja r e s os dois nós restantes: */

10: Adicione a aresta (r, s) na árvore.

utilizando o número de Prüfer torna-se improvável. Além disso, é importante salientar que há diversos problemas de projeto de redes que não correspondem a grafos completos.

Número de Prüfer Aplicado ao Problema de dc-MST

A representação do cromossomo deve conter explicitamente o grau de cada nó. Para controlar esse fator na codificação por número de Prüfer, basta verificar quantas vezes o nó ocorre na codificação. Se o grau do nó for d , significa que este ocorrerá no máximo $d - 1$ vezes na codificação.

A inicialização da população pode ocorrer de forma aleatória. O operador de *crossover* pode ser de um-ponto, o operador de mutação corresponde à alteração de um dígito no cromossomo por outro no intervalo entre 1 e n (n representa o número de nós no grafo) [25].

Devido à condição de restrição de grau, os cromossomos gerados para a população inicial, bem como os resultantes dos operadores de *crossover* e mutação, podem violar a restrição de grau. É necessário reparar os cromossomos inválidos, ou seja, alterar os nós que não estão obedecendo a limitação de grau. Para isso, o grau do nó que viola a restrição de grau é decrementado pela substituição aleatória deste nó por outro nó que não pertença ao cromossomo.

5.1.4 Chaves Aleatórias de Redes

Na codificação por Chaves Aleatórias de Redes (em inglês: *Network Random Keys*), um cromossomo é um *string* com pesos, um para cada aresta. Para identificar a árvore

representada pelo cromossomo, as arestas são ordenadas de acordo com seus pesos e o algoritmo de Kruskal [54]. Para árvore geradora mínima considera-se as arestas de acordo com tal ordenação.

Uma sequência de chaves de l números aleatórios $s_i \in [0, 1]$, onde $i \in \{0, \dots, l-1\}$, representa uma permutação p de tamanho l . Para a permutação p de tamanho $m = n(n-1)/2$ (onde cada aresta é identificada por algum $p[i]$), é construída uma árvore com n nós da seguinte maneira: para cada aresta verifica-se se esta pode ser inserida no grafo, ou seja, se não forma ciclo. Esse passo é repetido até que complete $n-1$ arestas, formando assim, uma árvore. Assim, uma única e válida árvore pode ser construída para qualquer sequência de chaves possível [62].

Algoritmo 8 - *Chaves Aleatórias de Redes* [59, 62].

```

1:  $i \leftarrow 0$ ;
2:  $G \leftarrow (V, E)$ ; /* onde  $|V| = n$  e  $E = \emptyset$ . */
3: while ( $|E| < n-1$ ) do
4:    $j \leftarrow p[i]$ .
5:   if ( not ciclo( $\text{insercao}(j, E)$ )) then
6:      $E \leftarrow E \cup \{j\}$ ;
7:   end if.
8:    $i \leftarrow i + 1$ ;
9: end while.
```

Para ilustração da codificação por Chaves Aleatórias de Redes, considere a Figura 5.4 e a Tabela 5.4. A permutação $p = 8 \rightarrow 1 \rightarrow 6 \rightarrow 2 \rightarrow 5 \rightarrow 10 \rightarrow 7 \rightarrow 3 \rightarrow 9 \rightarrow 4$ pode ser construída a partir de uma sequência de 10 números aleatórios. O algoritmo começa construindo o grafo G escolhendo a aresta 2-3 (posição 8 na Tabela 5.4). Em seguida, a aresta 0-1 é selecionada. Depois a aresta 1-3 é a que possui menor custo. Logo após, se a aresta 0-2 for adicionada, haverá a formação de ciclo no grafo 0-1-3-2-0, então esta aresta não é inserida. A próxima aresta considerada pelo algoritmo é a aresta 1-2, que também contribui para a criação de ciclo, sendo assim, descartada. A aresta 3-4 é então escolhida e inserida no grafo. Neste momento, o grafo completa $n-1$ arestas e o algoritmo pára.

Segundo Schindler *et al* [62], a mutação de uma chave resulta na mesma árvore, ou em uma árvore com no máximo duas arestas diferentes possuindo, dessa maneira, alta localidade. Além disso, os operadores de recombinação, geram filhos que herdam as propriedades de seus pais, provendo assim, boa hereditariedade. A representação por Chaves

Figura 5.4: Grafo com 5 nós.

posição	1	2	3	4	5	6	7	8	9	10
valor	0.82	0.69	0.31	0.12	0.47	0.76	0.38	0.97	0.21	0.45
aresta	0-1	0-2	0-3	0-4	1-2	1-3	1-4	2-3	2-4	3-4

Tabela 5.4: Representação por Chaves Aleatórias de Redes para a árvore da Figura 5.4.

Aleatórias de Redes sempre codifica soluções válidas. Porém, a factibilidade das soluções é garantida por meio do processamento adicional de um algoritmo para a verificação de ciclos [6] a cada nova aresta selecionada.

A seqüência de chaves que é utilizada para representar uma árvore considera todas as m possíveis arestas que um grafo com n nós pode ter. Construir uma árvore resulta na ordenação das m chaves que requer um tempo de computação $O(m \log(m))$ [59].

5.1.5 Conjunto de Arestas

A codificação por conjunto de arestas consiste em representar as arestas explicitamente na estrutura de dados. A seguir é descrita a codificação aplicada ao problema da árvore geradora mínima com restrição de grau.

Conjunto de Arestas Aplicado ao Problema de dc-MST

Na codificação por conjunto de arestas, a população deve conter somente soluções factíveis, isto é, o algoritmo deve gerar apenas árvores. Para este propósito, foi utilizado um algoritmo de inicialização de população (ver Algoritmo 9) baseado no algoritmo de árvore geradora mínima de Kruskal. Considere E o conjunto de arestas, V o conjunto de nós pertencentes ao grafo dado como entrada e E_T o conjunto de arestas que vão compor a árvore geradora resultante da inicialização. Para cada aresta $(i, j) \in E$ escolhida aleatoriamente, são verificados se os graus dos nós satisfazem a restrição de grau e se existe a formação de ciclo com a inserção desta aresta. Caso não haja a violação de grau e nem a formação de ciclos, a aresta (i, j) é incluída na solução. Este processo é repetido até

que o grafo possua $(n - 1)$ arestas, representando desta forma, uma árvore.

Algoritmo 9 - Inicialização por Conjunto de Arestas [55].

```

1:  $E_T \leftarrow \emptyset$ ;
2: for (todas arestas  $(i, j) \in E$  em ordem aleatória) do
3:   if ( $\text{grau}(i) < d$  and  $\text{grau}(j) < d$  and not  $\text{conectado}(i, j, E_T)$ ) then
4:      $E_T \leftarrow E_T \cup \{(i, j)\}$ ;
5:     if ( $|E_T| = |V| - 1$ ) then
6:       return  $E_T$ ;
7:   end if
8: end if
9: end for

```

O operador de mutação provê alta localidade, isto é, uma pequena alteração no cromossomo acarreta poucas mudanças na árvore. O princípio básico do operador de mutação é escolher aleatoriamente uma aresta a ser inserida e remover uma aresta pertencente ao ciclo formado pela inclusão da nova aresta. Inicialmente, uma aresta (i, j) é escolhida para inserção. O nó i é selecionado aleatoriamente e para garantia de que não haverá violação de grau, é necessário que os nós i e j não estejam simultaneamente com grau igual a d . O próximo passo é determinar o caminho (L) conectando os nós i a j . Em seguida, uma aresta (a, b) é escolhida para ser removida da árvore. Por último, (i, j) é inserido e (a, b) é removido da solução. Os passos do processo de mutação podem ser descritos pelo Algoritmo 10. Um exemplo de funcionamento deste operador é ilustrado na Figura 5.5.

Figura 5.5: Ilustração do operador de mutação por Conjunto de Arestas.

O operador de recombinação (*crossover*) foi desenvolvido de maneira a garantir uma

Algoritmo 10 - Mutação por Conjunto de Arestas [55].

```

/* escolha a aresta  $(i, j)$  para inserção: */
1: Escolha  $i \in V$  aleatoriamente;
2: Escolha  $j \in V \setminus \{i\}$  | grau( $j$ ) <  $d$  e  $(i, j) \notin E_T$  aleatoriamente;
/* escolha a aresta  $(a, b)$  para remoção: */
3:  $L \leftarrow \{(k, l) \in E_T \mid (k, l) \text{ compõe o caminho de } i \text{ a } j\}$ ;
4: if (grau( $i$ ) =  $d$ ) then
5:    $(a, b) \leftarrow (a, b) \in L \mid a = i \vee b = i$ ;
6: else
7:   Escolha  $(a, b) \in L$  aleatoriamente;
8: end if
9:  $E_T \leftarrow E_T \cup \{(i, j)\} \setminus \{(a, b)\}$ ;
10: return  $E_T$ ;

```

hereditariedade adequada, isto é, uma nova árvore é construída herdando o máximo possível de arestas de seus pais. Inicialmente, as arestas presentes em ambos os pais (E_T^1 e E_T^2) são incluídas na nova árvore (E_T), sempre obedecendo a restrição de grau. O próximo passo é inserir as arestas do conjunto F , pertencentes à disjunção das arestas de seus pais. Se, após a inclusão desses conjuntos, uma árvore geradora com restrição de grau d válida não foi formada, então cada componente desconexa U_k é determinada e arestas são selecionadas aleatoriamente para conexão de cada componente à árvore que está sendo formada. No Algoritmo 11 são descritos os passos da aplicação do operador de recombinação. Para ilustrar o funcionamento do operador de *crossover*, considere a Figura 5.6.

Os operadores de mutação e recombinação podem ser implementados em tempo $O(n)$ e inicialização das soluções, em tempo $O(n^2)$ [55]. Buscando melhorar o desempenho do algoritmo, foi proposta em [55] a utilização de heurísticas sobre a inicialização e os operadores de mutação e recombinação.

5.1.6 Nó-profundidade

A codificação nó-profundidade consiste de apenas nós e suas respectivas profundidades, formando pares do tipo $(\text{nó}, \text{profundidade})$ [12].

Algoritmo 11 - *Crossover* por Conjunto de Arestas [55].

```

1:  $E_T \leftarrow E_T^1 \cap E_T^2$ 
2:  $F \leftarrow E_T^1 \cup E_T^2 \setminus E_T$ ;
3: for (todas arestas  $(i, j) \in F$  em ordem aleatória) do
4:   if ( $\text{grau}(i) < d$  and  $\text{grau}(j) < d$  and  $(i, j) \notin E_T$ ) then
5:      $E_T \leftarrow E_T \cup \{(i, j)\}$ ;
6:     if ( $|E_T| = |V| - 1$ ) then
7:       return  $E_T$ ;
8:     end if
9:   end if
10: end for
    /* determine todas componentes desconexas:  $U_k$  */
11:  $U \leftarrow \{U_k\}, \forall i, j \in V, i \neq j$ ;
12:  $i \in U_k \wedge \text{conectado}(i, j, E_T) \longrightarrow j \in U_k$ ,
13:  $i \in U_k \wedge \text{not}(\text{conectado}(i, j, E_T)) \longrightarrow j \notin U_k$ .
14:  $\bigcup_k U_k = V$ ;
    /* conecte as componentes aleatoriamente */
15: for ( todo  $U_k \in U \setminus U_1$  em ordem aleatória ) do
16:   Escolha  $i \in U_1 \mid \text{grau}(i) < d$  aleatoriamente;
17:   Escolha  $j \in U_k \mid \text{grau}(j) < d$  aleatoriamente;
18:    $E_T \leftarrow E_T \cup (i, j)$ ;
19:    $U_1 \leftarrow U_1 \cup U_k$ ;
20: end for
21: return  $E_T$ ;

```

Figura 5.6: Ilustração do funcionamento do operador de recombinação por Conjunto de Arestas.

Representação Nó-profundidade

A representação é realizada a partir de uma busca em profundidade (ver Figuras 5.7(a) e 5.7(b)).

É importante ressaltar que esta representação produz florestas geradoras [12], enquanto que outras codificações trabalham com árvores geradoras [25, 34].

Operadores:

A codificação nó-profundidade apresenta dois operadores bastante similares para gerar novas florestas, denominados operador 1 e operador 2. Estes operadores produzem novas florestas geradoras F' de um grafo G quando aplicadas a outra floresta F de G [13]. Ambos os operadores transferem uma sub-árvore de uma árvore T_{dc} para T_{para} . Enquanto que no operador 1 a raiz da sub-árvore podada é a mesma em T_{dc} e T_{para} , no operador 2, um novo nó (diferente da raiz) é escolhido para ser a nova raiz da sub-árvore em T_{para} [12].

(a) Um grafo G com uma árvore geradora T . T está indicada pelas arestas espessas.

profundidade	0	1	2	1	2	2	3	4	3	4	4
nó	1	4	7	3	2	6	5	8	9	10	11

(b) Representação nó-profundidade.

Figura 5.7: Codificação nó-profundidade.

O operador 1 necessita a consideração de dois nós em especial: o nó p , que indica a raiz da sub-árvore a ser transferida e o nó adjacente a , que não pertence a T_{de} mas é adjacente a p . O operador 2 requer além desses dois nós, o nó r que será a nova raiz da sub-árvore.

Operador 1

Assuma que os nós p e a foram previamente escolhidos e a implementação da codificação utiliza *arrays*. Considere também como conhecidos os índices de p (i_p) e a (i_a) nos *arrays* T_{de} e T_{para} .

O operador 1 pode ser descrito pelo Algoritmo 12.

Algoritmo 12 - Operador 1 [13].

- 1: Determine o intervalo $(i_p - i_l)$ de índices em T_{de} correspondente à sub-árvore enraizada pelo nó p . Encontre i_l ;
 - 2: Copie os dados do intervalo $(i_p - i_l)$ de T_{de} em um *array* temporário T_{temp} ;
 - 3: Crie um *array* T'_{para} contendo os nós de T_{para} e de T_{temp} (isto é, gere uma nova árvore conectando a sub-árvore podada a T_{para});
 - 4: Construa um *array* T'_{de} compreendendo os nós de T_{de} sem os nós de T_{temp} ;
 - 5: Copie F para F' mudando o ponteiro de T_{de} para T'_{de} e de T_{para} para T'_{para} ;
-

Operador 2

O operador 2 necessita a consideração de três nós para a descrição do algoritmo: o nó a ser podado (p), a nova raiz (r) e o nó adjacente (a). Os nós p e r pertencem a T_{dc} e o nó a está em T_{para} . As diferenças entre os operadores 1 e 2 estão nos passos 2 e 3, ou seja, somente a formação e armazenamento das sub-árvores podadas em *arrays* temporários são diferentes [13].

Para o operador 2, o procedimento de cópia da sub-árvore podada pode ser definido por dois passos. O primeiro passo é similar ao passo 2 do operador 1, diferindo no índice. Ao invés de ser i_p , considera-se i_r . O *array* retornado por esse procedimento é denominado T_{temp1} .

O segundo passo considera os nós no caminho de r a p (isto é, $r_0, r_1, r_2, \dots, r_n$, onde $r_0 = r$ e $r_n = p$) como raízes de sub-árvores. A sub-árvore enraizada por r_1 contém a sub-árvore enraizada por r_0 . A sub-árvore enraizada por r_2 contém a sub-árvore enraizada por r_1 e assim por diante. O algoritmo para o segundo passo faz a cópia das sub-árvores enraizadas por r_i , onde $i \in \{1, \dots, n\}$, sem copiar a sub-árvore enraizada por r_{i-1} e armazena as sub-árvores em um *array* temporário T_{temp2} .

No passo 3 do operador 2, ao invés de T'_{para} ser criado a partir de T_{para} e T_{temp} , como no passo 3 do operador 1, T'_{para} é construído utilizando os *arrays* T_{para} , T_{temp1} e T_{temp2} .

Deve-se observar que esses operadores sempre produzem novas soluções factíveis sem a necessidade de algoritmos de reparo ou de verificação de ciclos [6]. Algoritmos eficientes para determinação dos nós p , r e a apropriados são apresentados em [12].

Nó-profundidade Aplicado ao Problema de dc-MST

As árvores iniciais podem ser geradas por um algoritmo de árvore geradora mínima. Em relação à aplicação dos operadores, para o operador 1 verifica-se o nó a em T_{para} obedece a restrição de grau após a inserção da sub-árvore de T_{dc} .

5.1.7 Precedentes Diretos

Uma rede de distribuição de energia representada na forma de grafos é composta por centenas de nós correspondendo às sub-estações, pontos consumidores, pontos de conexões e as arestas representando cabos elétricos. Em operação normal, cada ponto consumidor ou de conexão é ligado a uma sub-estação por apenas um único caminho. Portanto, uma rede quando em operação representa uma árvore geradora, sendo o objetivo da reconfigu-

ração de distribuição de energia encontrar aquela de custo mínimo. Tal problema poderia ser resolvido eficientemente com algoritmos para problema de árvore geradora mínima, se o custo para cada aresta fosse fixo, ou seja, não variasse de acordo com o fluxo de energia. A busca de soluções ótimas para árvores geradoras com custo variável cresce exponencialmente à medida que aumenta o tamanho da rede.

Em [4] foi proposta uma representação para o problema, no qual o cromossomo que representa a árvore geradora pode ser representado por um conjunto de nós parcialmente ordenados $(A, \leq)$, onde:

1. $a \leq a$;
2. $a \leq b$ e $b \leq a$ implica $a = b$;
3. $a \leq b$ e $b \leq c$ implica $a \leq c$;

$\forall a, b, c \in A$.

Um elemento a é denominado precedente direto de b em A (denotado por $b \leftarrow a$) se e somente se:

- i. $a \neq b$;
- ii. $a \leq b$;
- iii. $\nexists c \in A$ tal que $a \leq c$ e $c \leq b$.

Similarmente, b é dito sucessor direto de a . Utilizando estes conceitos podemos definir a codificação em dois *arrays* (X, Y) de mesmo tamanho, onde em X serão armazenados todos os elementos, exceto a raiz e em Y , os precedentes diretos dos elementos em X (ver Figuras 5.8 e 5.9). Desta maneira, $X \leftarrow Y$ representa uma árvore geradora. Esta operação de codificação pode ser realizada com o auxílio de uma busca em largura. Quando se realiza a alteração de precedência é necessário verificar se esta é consistente, ou seja, se as regras (i-ii-iii) são obedecidas. Caso contrário, é dita inconsistente.

O algoritmo de recombinação é realizado por intercâmbio de caminhos entre duas árvores geradoras. Os principais passos para gerar as soluções filhas são mostrados no Algoritmo 13.

Seja a o menor elemento do caminho C . Denote por $S(x)$ os sucessores diretos de x em C . Considere um conjunto E que conterá os elementos da árvore, inicialmente $E = S(a)$. Diante dessas premissas, o passo 3 pode ser realizado pelo Algoritmo 14.

A recombinação proposta apresenta as seguintes vantagens, segundo os autores [4]:

Figura 5.8: Árvore com 5 nós.

Figura 5.9: Representação por precedentes diretos para a árvore da Figura 5.8.

Algoritmo 13 - Recombinação por precedentes diretos.

- 1: Selecione aleatoriamente dois nós a e b ;
- 2: Encontre os caminhos C_1 e C_2 entre a e b : C_1 em T_1 e C_2 em T_2 ;
- 3: Se for possível, submeta C_2 a T_1 e C_1 a T_2 .
- 4: Caso contrário, volte ao passo 1.

Algoritmo 14 - Intercâmbio entre caminho e árvore.

- 1: Altere T , modificando toda relação de precedência $x \leftarrow y$ de T para $x \leftarrow z$ de C , se e somente se, $x \in E$ e $x \leftarrow z$ é consistente em T .
- 2: Faça $E = \cup S(x \in E)$. Se $E \neq \emptyset$, volte ao passo anterior.

- baixo custo computacional: alterar a precedência de um elemento envolve exclusivamente uma operação de substituição no *array* de precedentes diretos.
- convergência rápida: troca de caminhos durante a recombinação propagam grandes blocos de informação, reduzindo o espaço de busca significativamente.
- viabilidade: uma simples verificação de consistência durante o intercâmbio de caminhos garante a factibilidade da solução.

No entanto, a verificação de viabilidade da solução requer uma busca em largura que é $O(n + m)$ reduzindo a eficiência computacional do método.

Precedentes Diretos Aplicado ao Problema de dc-MST

De forma indireta, o grau de cada nó está representado pela quantidade de vezes em que este aparece nos vetores X e Y . Ao recombinar soluções no problema de *dc-MST* deve ser feita uma varredura nos vetores para obter o grau de cada nó e adicionar a verificação de restrição de grau no passo 1 do Algoritmo 14.

5.2 Considerações Finais

A Tabela 5.5 sintetiza as complexidades de tempo e espaço das codificações mostradas neste Capítulo.

Representação	Tempo	Espaço
Vetor de características	$O(m)$	$O(m)$
Predecessores	$O(n)$	$O(n)$
Número de Prüfer	$O(n \log n)$	$O(n)$
Chaves Aleatórias de Redes	$O(m \log m)$	$O(m)$
Conjunto de Arestas	$O(n)$	$O(n)$
Nó-profundidade	$O(n)$	$O(n)$
Precedentes diretos	$O(n + m)$	$O(n)$

Tabela 5.5: Complexidade de tempo e espaço onde n e m representam as quantidades de vértices e arestas, respectivamente.

AEs para problemas de PR necessitam de uma codificação especial de cromossomo, preferencialmente que obedeçam as propriedades descritas na Seção 5.1. Ao longo deste Capítulo, foram apresentadas codificações como a de Prüfer que, infelizmente, não possuem as propriedades de codificação, devido a sua baixa localidade e hereditariedade [27]. Já a codificação por predecessores, apesar de possuir tempo linear no processo de codificação/decodificação, traz complexidade maior a um AE pois esta solução gera soluções infactíveis, necessitando de algoritmos de reparo e penalidade.

Por outro lado, codificações como a representação por Conjunto de Arestas possuem desempenhos em tempo linear e satisfatórios para grandes entradas. A representação Nó-profundidade aplicada ao problema de *dc-MST* tem apresentado tempos de computação relativamente baixos e alta localidade [13] em testes realizados em grafos com até 1000 nós, com pesos gerados aleatoriamente.

Proposta

Este projeto de mestrado objetiva a implementação de um programa de computador capaz de construir árvores filogenéticas que melhor expliquem as relações evolutivas entre as espécies. Como este objetivo corresponde a um problema combinatorial, é necessário que o programa proposto, ao trabalhar com grandes entradas, seja eficiente o suficiente para obter árvores adequadas em um tempo de computação razoável.

Tal eficiência será buscada pelo desenvolvimento de um AE com uma codificação para árvores que permita a geração de novos indivíduos rapidamente sem a perda de propriedades importantes para uma representação de árvores.

Para a obtenção de um programa de computador capaz de construir árvores para uma grande quantidade de seqüências, investigou-se o emprego de AEs e codificação por Conjunto de Arestas [55] (ver Seção 5.1.5).

A escolha da codificação foi determinada com base em revisões na literatura, verificando-se que a representação por Conjunto de Arestas possui operadores $O(n)$ em tempo e espaço e obedece as propriedades de codificação descritas em [50]. Apesar da representação nó-profundidade também ser $O(n)$ em tempo e espaço, a codificação por Conjunto de Arestas oferece facilidade para implementação e para adaptação a problemas distintos, além de possuir resultados encontrados na literatura que comprovam sua superioridade em relação a outras codificações existentes.

Uma árvore filogenética pode ser vista como um problema de projeto de redes, mais especificamente como uma árvore geradora com restrição de grau (*dc-MST*) igual a três, ou seja, uma *3-MST*. Desta maneira, o trabalho foi dividido em duas etapas. A primeira consistiu em implementar os AEs utilizando a codificação Conjunto de Arestas (ver Seção 5.1.5) para a obtenção da árvore geradora mínima com restrição de grau. Este problema requer formas mais adequadas para representação de árvores, assim como o problema de filogenia. Assim, a resolução do problema da árvore geradora mínima com restrição de grau pode ser visto como um estágio para a obtenção do algoritmo para filogenia.

A segunda etapa trata-se de adaptar essa abordagem ao problema de filogenia. A restrição de grau para o problema é três. A população inicial é gerada da seguinte forma:

- 1) Dado um conjunto com n (número de espécies) nós, todos os nós de grau zero são conectados primeiro para evitar a formação de grafos não conexos (ver Figura 6.1(a)).
- 2) A cada inclusão de aresta (i, j) verifica-se se os nós pertencem a componentes diferentes e se a quantidade de nós com grau dois é diferente de um em ambas as componentes, caso contrário pode acontecer a construção de árvores que não sejam binárias¹ (ver Figura 6.1(b)). Caso contrário, seleciona-se aleatoriamente outra aresta.
- 3) Os passos 1 e 2 são realizados até completar uma árvore.

Foram implementadas três variações para o algoritmo:

1. *crossover* aplicado como se o problema fosse da árvore geradora mínima com restrição de grau (conforme descrito na Seção 5.1.5) e a mutação considera a mudança de uma aresta de posição. Este algoritmo foi denominado de AE1;
2. *crossover* levando-se em conta as sub-árvores em comum entre os indivíduos pais (tal estratégia é similar a idéia utilizada por Lemmon *et al* [36] aplicada ao operador de mutação); e a mutação como sendo a troca de sub-árvores de posição. Este algoritmo foi chamado de AE2. Deve-se observar que para determinação das sub-árvores comuns foi necessária a implementação de algoritmos de isomorfismo de árvores [61];

¹ver Apêndice.

Figura 6.1: Exemplos de situações em que não foram possíveis gerar árvores. Os nós em negrito representam ancestrais hipotéticos, os demais dizem respeito aos nós folhas.

3. *crossover* considerando-se a poda de uma sub-árvore em T_1 , enxertando-a em T_2 e, por consequência, eliminando-se as espécies repetidas (Congdon *et al* [5] aplicou esta estratégia ao operador de *crossover* em seu trabalho). A mutação considera a troca de duas arestas de lugar. A este algoritmo foi dado o nome de AE3.

As soluções candidatas são avaliadas por meio da parcimônia de *Fitch* (ver Seção 3.5.1) onde o *fitness* considerado foi o número de mutações ocorridas em uma árvore. Tais soluções são selecionadas de acordo com o ranqueamento, sempre utilizando a estratégia elitista. O algoritmo em geral produz uma ou mais árvores com maior aptidão como resposta.

Resultados

Os resultados referentes ao desempenho do AE para o problema de *dc-MST* podem ser vistos em [14]. De forma geral, esses resultados comprovam seu comportamento linear como proposto por Raidl *et al* [55]. Além disso, tal codificação foi estendida para grafos orientados e aplicada para *One-Max Tree Problem* [15].

Os testes para filogenia consideraram seqüências de *DNA* previamente alinhadas, de diversos tamanhos considerando 5, 6, 12, 14 e 55 espécies¹ [68, 73, 74] e um outro conjunto de 186 espécies [75]. Com este último conjunto, busca-se ampliar a avaliação do algoritmo para grandes instâncias (55 e 186 seqüências).

Os resultados foram obtidos considerando-se a taxa de *crossover* como sendo a probabilidade de ocorrer a recombinação entre dois indivíduos e a taxa de mutação como sendo a probabilidade de em um indivíduo um valor no cromossomo ser mutado. A taxa de substituição corresponde a quantidade de indivíduos que vão ser trocados pelos novos indivíduos e a taxa de elitismo indica a porcentagem de indivíduos que serão preservados para a próxima geração. Todos os testes foram realizados por 20 execuções, onde foram obtidos a melhor e a média das aptidões encontradas e o melhor tempo e tempo médio de execução.

¹O conjunto de 55 espécies foi utilizado em [38].

Foram testados subconjuntos de seqüências obtidas a partir do conjunto de 55 espécies para analisar melhor os resultados em termos de tempo de computação. Foi verificado como o AE desenvolvido pode contribuir em relação a algoritmos clássicos como o método de parcimônia desenvolvido para o programa *DNAPars* do pacote (*PHYLIP*) [20] e *NJ* para construção de árvores filogenéticas.

A comparação dos AEs desenvolvidos torna-se indireta, visto que os critérios de avaliação utilizados são diferentes, um método utiliza árvore enraizada e outro não, indisponibilidade de *softwares* para efeito de comparação dos resultados e, assim por diante. A seguir são sintetizados os principais resultados dos trabalhos encontrados na literatura que aplicam AEs ao problema de filogenia.

Alguns AEs utilizaram matriz de distâncias como critério de avaliação. Matsuda [40] empregou uma representação baseada em grafos (ver Seção 4.3.2) onde a população evoluiu uma matriz de distâncias até que todos os indivíduos encontrassem a mesma aptidão utilizando um conjunto com 15 seqüências de proteínas. Os resultados foram considerados comparáveis a outros métodos como *UPGMA*, *NJ*, Máxima Verossimilhança, Máxima Parcimônia para seqüências de proteínas (*ProtPars* do pacote *PHYLIP*). Prado e Von Zuben [51, 52] desenvolveram um software utilizando duas opções de reconstrução de árvores filogenéticas: matriz de distâncias e máxima verossimilhança. A representação das soluções foi baseada em árvores de grafos (ver Seção 4.3.2). Os testes foram realizados para instâncias com até 20 seqüências de *DNA*. Em comparação com o *PHYLIP* e *PAML* [70], o AE encontrou resultados intermediários entre esses dois programas [52]. Tanto a proposta de Matsuda quanto a de Prado e Von Zuben obtiveram resultados compatíveis com o de algoritmos clássicos. Porém, consideram apenas pequenos números de seqüências.

Vários outros AEs foram desenvolvidos utilizando máxima verossimilhança como critério de avaliação. Katoh *et al* [32] aplicaram algoritmo genético utilizando máxima verossimilhança. O algoritmo além de buscar a melhor árvore, encontra também outras filogenias próximas da melhor. Os testes consideraram seqüências entre (5-24) espécies. Este AE encontra mais topologias no mesmo tempo de computação comparado ao *PHYLIP* e mostrou-se dependente do método de inicialização de população. Cotta *et al* [7] utilizaram duas estratégias em relação ao espaço de busca. Foram utilizados instâncias com 20 e 34 espécies. Não foram informados o tempo de execução do algoritmo e comparação em relação a outros softwares. Tais abordagens consideraram testes para pequenos números de seqüências.

Lewis [38] também desenvolveu um AG utilizando o método de máxima verossimi-

lhança. Foram realizados testes para um conjunto de 55 seqüências e estes mostraram desempenho superior ao programa *PAUP*. Lemmon *et al* [36] utilizaram meta-populações no algoritmo genético utilizando o método de máxima verossimilhança com a estratégia de árvore de consenso em seus operadores. Destacam-se a utilização de conjuntos com 55 seqüências e 500 táxons com 3000 nucleotídeos. Este último, segundo o autor, pode ser analisado em 10 horas de execução em um processador adequado. O algoritmo mostrou-se superior ao programa *PAUP*. Skourikhine [63] desenvolveu um algoritmo genético auto-adaptativo utilizando o método de máxima verossimilhança. Os testes foram realizados para um conjunto de 55 seqüências e as árvores encontradas foram equivalentes às do *PHYLIP*. O tempo de execução do algoritmo não foi informado. Para quantidade de seqüências relativamente grande (55), os AEs desenvolvidos por Lewis e Lemmon apresentaram tempos de computação inferiores ao programa *PAUP*.

Os AEs apresentados a seguir utilizaram parcimônia de Wagner como critério de avaliação para caracteres binários. Segundo Congdon *et al* [5], o AG desenvolvido (*GAPhyI*) produziu várias árvores parcimoniosas no mesmo tempo de computação em relação ao *PHYLIP* para pequenos números de seqüências. Moilanen [45] desenvolveu um programa que combina AE com busca local. Os resultados encontrados pelo programa foram razoavelmente melhores em termos de aptidão [45], quando comparado a outros softwares da literatura. O tempo dispendido foi considerado não comparável, uma vez que o AE desenvolvido encontra apenas uma árvore.

Pode-se perceber várias dificuldades quanto à obtenção de resultados e comparações nos trabalhos encontrados na literatura. Alguns autores não levam em consideração a quantidade de tempo dispendida para cada teste realizado. Outros utilizaram pequenas quantidades de seqüências impossibilitando mostrar a capacidade computacional de um AE e também incorrendo em erros do ponto de vista biológico.

O desempenho do AE proposto mostrou-se similar aos dos demais trabalhos com AEs para pequenos conjuntos de seqüências. Em geral, os trabalhos não lidaram com grande quantidade de seqüências. Assim, uma das contribuições deste trabalho é fornecer um AE viável para filogenia de grandes números de seqüências de *DNA*. Além disso, o AE proposto utiliza parcimônia de Fitch, a qual não havia sido empregada pelas demais abordagens evolutivas para filogenia.

A seguir o desempenho do AE proposto é comparado com o desempenho de algoritmos largamente utilizados para filogenia: *NJ* e *DNAPars* do pacote *PHYLIP*. Como *NJ* é um algoritmo de construção de apenas uma árvore e *DNAPars* a partir de uma árvore

tenta rearranjá-la de forma a encontrar um ótimo, não é adequado buscar maior eficiência computacional do AE proposto (que trabalha com múltiplas árvores) em relação a esses algoritmos. Os testes apresentados a seguir buscaram a obtenção de um AE capaz de contribuir na qualidade das árvores filogenéticas encontradas, sem com isso requerer tempo de computação abusivo. Por qualidade das árvores, entende-se tanto uma árvore com melhor avaliação (no caso, maior parcimônia) quanto a obtenção de um conjunto diversificado de árvores filogenéticas (com avaliações semelhantes).

Quando executado o teste para 5 seqüências (tipos de macacos e a espécie humana) de tamanho 69 [68], o AE1 atingiu o mesmo resultado encontrado pelo *DNAPars* do *PHYLIP* (aptidão igual a 17). A quantidade de árvores encontradas com melhor aptidão variou de 1 a 10 (tamanho da população). Este e outros testes exibidos na Tabela 7.1 indicaram que o AE1 rapidamente encontra árvores com aptidões próximas às encontradas pelo *PHYLIP* para conjuntos com até 14 seqüências.

Os primeiros testes com 55 seqüências mostraram um desempenho relativamente inferior dos algoritmos implementados. O *DNAPars* do pacote *PHYLIP* apresentou solução com aptidão 4669, enquanto o AE1 obteve parcimônia maiores que 5000, AE2 encontrou o valor 5855 e AE3 6058. Diante desses resultados foram realizadas alterações e testes no AE1 buscando aumentar seu desempenho. A seguir são apresentadas as principais modificações.

Primeiramente verificou-se a influência dos operadores de AE1 no processo de otimização. A Figura 7.1 apresenta o desempenho do algoritmo AE1 com inicialização aleatória, com os dois pais escolhidos aleatoriamente no processo de *crossover*. Considerando-se a influência dos operadores de *crossover* e mutação, foi possível verificar que o operador de *crossover* (melhor aptidão 5472) colabora mais na convergência do algoritmo em relação ao operador de mutação (a melhor árvore encontrada possui aptidão 6807). Conforme esperado, a combinação dos dois operadores produz melhor resultado (melhor aptidão 5207). Este resultado motivou a investigação de operadores de *crossover* ainda mais eficientes, dada a sua importância para o problema de filogenia. Nesse sentido, foi desenvolvido um operador de *crossover* que utiliza a informação de consenso entre árvores (isomorfismo de grafos) [61] para efetuar a recombinação. O AE que utiliza este operador foi denominado AE2.

Outro teste realizado na busca por melhoria de desempenho do AE1 utilizou a idéia de migração até 20 execuções independentes, onde quase todos os indivíduos eram gerados com exceção do melhor indivíduo que era sempre preservado para a próxima rodada. O

Figura 7.1: Desempenho da aptidão ao longo de 100 gerações para 55 seqüências utilizando inicialização aleatória.

AE1 encontrou aptidão 5534 e AE2, 6252. Portanto, os testes indicam que o método de migração não é adequado para este problema.

Buscando ainda melhorar o desempenho da proposta para grande número de seqüências, foram investigadas variações na inicialização da primeira população. Os testes envolveram populações inicializadas aleatoriamente e com *seeding*. Avaliou-se o *seeding* pela solução obtida pelo *NJ* e pelo programa *DNAPars* do pacote *PHYLIP*. De acordo com a Tabela 7.2 é possível concluir que o AE1 mostrou-se bastante dependente do procedimento de inicialização e o operador de mutação contribuiu muito pouco para a convergência do algoritmo.

Para o AE utilizando o algoritmo *NJ* como *seeding*, a matriz de distâncias utilizada pelo *NJ* foi gerada pelo programa *DNADist* do pacote *PHYLIP*. O modelo de substituição utilizado foi *F84*. A topologia gerada foi dada como entrada para o AE1.

Considerando-se o conjunto de 55 espécies, os algoritmos AE1 e AE2, ambos com *seeding* do *PHYLIP*, encontrou após 50 gerações, árvores com a mesma aptidão do *PHYLIP* (custo 4669). Para AE3 o mesmo não aconteceu. Após 100 gerações não foi encontrada nenhuma solução melhor ou com mesma aptidão que a fornecida pelo *PHYLIP* (o melhor indivíduo encontrado pelo algoritmo AE3 com base na solução do *PHYLIP* possui

aptidão 4673).

Com *NJ* como algoritmo de inicialização, a árvore passada para o AE1 possui aptidão 5067, após 100 gerações o AE1 encontrou uma árvore com aptidão 4808 em 152 segundos. Na Figura 7.2 é mostrado o desempenho do algoritmo AE1 para o caso referido.

Figura 7.2: Desempenho da aptidão ao longo de 100 gerações para 55 seqüências utilizando *NJ* como *seeding*.

Em outro teste realizado, na inicialização da população foram incluídas tanto a árvore encontrada pelo *PHYLIP* quanto a obtida pelo *NJ*. Não houve diferença significativa em relação a considerar apenas o *seeding* com a árvore obtida pelo *PHYLIP*. A quantidade de árvores encontradas com melhor aptidão e a característica de tornar a população homogênea (o intervalo entre a melhor e a pior aptidão reduziu consideravelmente) refletiu-se também para este teste.

Considerando-se o conjunto com 186 seqüências e 16608 nucleotídeos, o AE1 encontrou parcimônia mediana de aptidão 11329 para 20 execuções, enquanto o *PHYLIP* foi executado por uma semana sem finalizar com sucesso, uma vez que o programa foi abortado antes da obtenção da aptidão para a árvore mais parcimoniosa. Considerando o *seeding* do *NJ* (com aptidão 9580) o AE1 encontrou custo 9578 para a árvore mais parcimoniosa com os parâmetros escolhidos empiricamente após duas horas de execução.

Em geral, os AEs desenvolvidos encontraram uma ou mais árvores com mesma aptidão. Tal verificação foi realizada por algoritmo de isomorfismo de árvores [61].

O desempenho do AE proposto também foi avaliado experimentalmente em termos de tempo de computação. A Figura 7.3 e a Tabela 7.3 mostram o desempenho do AE1 para as seqüências criadas especialmente para as simulações. Pode-se observar um comportamento praticamente linear.

Figura 7.3: Tempo para 10, 20, 30, 40, 50 e 55 seqüências de tamanho 1314 utilizando AE1.

Os tempos de execução para os algoritmos AE2 e AE3 mostraram-se similares em relação ao AE1.

A implementação do programa foi realizada na linguagem *C++*, na plataforma Linux e conta com os recursos computacionais disponíveis (*cluster* com 12 nós: Pentium dual Xeon 2.8GHz, 1GB de RAM) do grupo de Computação Bioinspirada do ICMC (USP).

	tamanho população	número gerações	taxa <i>crossover</i>	taxa mutação	taxa substituição	taxa elitismo	AE1 melhor aptidão	AE1 aptidão média	<i>PHYLIP</i> aptidão	AE1 melhor tempo(s)	AE1 tempo médio
5 seqs/tam. 69	100	100	80%	50%	60%	10%	17	17	17	0.00	0.00
6 seqs/tam. 444	100	100	80%	50%	60%	10%	439	439	439	0.05	0.06
12 seqs/tam. 898	100	100	80%	50%	60%	10%	1183	1188.5	1163	11.92	12.1
14 seqs/tam. 232	100	100	80%	50%	60%	10%	747	753.8	747	6.90	6.95

Tabela 7.1: AE1 para conjuntos de seqüências com inicialização aleatória.

	tamanho população	número gerações	taxa <i>crossover</i>	taxa mutação	taxa substituição	taxa elitismo	aptidão	tempo(s)
In. Al.	100	50	80%	80%	60%	50%	5851	76
<i>seeding</i> do <i>NJ</i>	100	50	80%	80%	60%	50%	4877	72
<i>seeding</i> do <i>PHYLP</i>	100	50	80%	80%	60%	50%	4669	72
In. Al.	100	100	80%	80%	60%	50%	5462	153
In. Al. s/ <i>crossover</i>	100	100	80%	80%	60%	50%	6785	102.5
In. Al. s/mutação	100	100	80%	80%	60%	50%	5472	78
<i>seeding</i> do <i>NJ</i> s/ <i>crossover</i>	100	100	80%	80%	60%	50%	5067	101
<i>seeding</i> do <i>NJ</i> s/mutação	100	100	80%	80%	60%	50%	4978	76.2

Tabela 7.2: AE1 para 55 seqüências de tamanho 1314, onde In. Al. significa inicialização aleatória da população sem a utilização de *seeding*.

	tamanho população	número gerações	taxa <i>crossover</i>	taxa mutação	taxa substituição	taxa elitismo	AE1 melhor aptidão	AE1 aptidão média	<i>PHYLP</i> aptidão	AE1 melhor tempo(s)	AE1 tempo(s) médio
10 seqs	100	100	80%	50%	60%	10%	1230	1231	1230	23.22	23.33
20 seqs	100	100	80%	50%	60%	10%	2006	2030	2000	47.68	48.26
30 seqs	100	100	80%	50%	60%	10%	2855	2941.35	2807	74.59	75.10
40 seqs	100	100	80%	50%	60%	10%	3963	4136.35	3838	113.97	114.55
50 seqs	100	100	80%	50%	60%	10%	4742	4862	4326	146.54	149.2
55 seqs	100	100	80%	50%	60%	10%	5099	5388.3	4669	139.46	140.01

Tabela 7.3: AE1 para conjuntos de seqüências de tamanho 1314 com inicialização aleatória.

Considerações Finais

Desde os tempos de Darwin há o interesse em estudar a história da evolução das espécies. Tal história pode ser representada por uma estrutura em forma de árvore. No entanto, à medida que aumenta o número de espécies, cresce combinatorialmente o número de possíveis árvores a serem analisadas. Neste sentido, durante a realização do projeto foi estudado diversos métodos de construção de árvores filogenéticas, analisando vantagens e desvantagens de cada um.

Os métodos podem ser classificados de acordo com os dados utilizados (distância ou características), bem como a maneira de reconstrução das árvores (métodos de *clustering* ou por critério de otimização). Quando se trata de métodos com critérios de otimização, deve-se considerar um espaço de busca e uma função de avaliação que minimize ou maximize determinado critério. Os AEs têm mostrado resultados relevantes em tempo razoável de computação. As principais características dos AEs são: trabalhar com conjuntos de possíveis soluções, não necessitar de cálculos matemáticos complexos, evitar ótimos locais. Além disso, possui um conjunto de passos genéricos e adaptáveis a um grande número de problemas principalmente àqueles de alto custo computacional com grandes instâncias.

Devido ao fato de a filogenia ser vista computacionalmente também como um problema de projeto de redes foram revisadas as codificações (estruturas de dados de grafos) mais utilizadas na literatura para este tipo de problema. Nos trabalhos [51, 56, 40] foram

utilizadas codificações de árvores para o problema de filogenia considerando-se métodos de distância ou de máxima verossimilhança como critérios de avaliação.

Apesar de não ter um fundamento estatístico bem definido na literatura, estudos empíricos sugerem que métodos como *Neighbor Joining*, Máxima Parcimônia, Máxima Verossimilhança geram árvores filogenéticas adequadas quando um número suficientemente grande de nucleotídeos é utilizado. Não existe um consenso de qual método filogenético seja melhor. Cada qual possui vantagens e desvantagens. Se for possível, recomenda-se utilizar os métodos conjuntamente para tirar conclusões.

O AE proposto mostrou-se adequado para encontrar uma diversidade de soluções com a melhor parcimônia encontrada ou com parcimônias próximas à melhor. Além disso, o AE proposto é capaz de encontrar soluções apropriadas em tempos de computação razoáveis mesmo quando são utilizados grandes números de seqüências. Devo-se ressaltar a importância em utilizar grande quantidade de seqüências para a obtenção de filogenia conforme mostra [58].

Os resultados relacionados a este trabalho mostraram o quão complexo é o problema de filogenia. Foram testadas três propostas de AEs diferentes (variações para o processo de *crossover*). Para a operação de mutação verificou-se uma pequena contribuição durante o processo de convergência do algoritmo. Quando considerada a técnica de *seeding* verificou-se uma melhoria na obtenção da melhor árvore, mostrando, desta maneira, a dependência do AE em relação ao processo de inicialização do algoritmo. Os parâmetros do AE foram escolhidos empiricamente.

Poucos são os trabalhos aplicando AE que utiliza a máxima parcimônia como função de avaliação. Dentre esses, todos citados na Seção 4.3.2 utilizam a parcimônia de Wagner [18] com caracteres binários. Quando da utilização de caracteres de *DNA* aumenta-se consideravelmente a complexidade do problema.

Pôde-se perceber a impossibilidade de fazer comparações diretas com o que existe na literatura, devido ao fato de, por exemplo, utilização de modelos evolutivos diferentes, alguns pesquisadores trabalharem com árvores sem raiz, outros com raiz, indisponibilidade de *softwares* relacionados a AEs para filogenia e assim por diante.

Nas Seções seguintes são apresentadas as contribuições e perspectivas para uma possível continuidade para o trabalho.

8.1 Contribuições

As principais contribuições deste trabalho são citadas a seguir:

- Mostrar o poder da Computação Evolutiva na exploração do espaço de busca para este problema, a qual “evolui” várias árvores candidatas ao mesmo tempo, fornecendo uma ou mais topologias como resposta em um tempo razoável de computação à medida que aumenta o tamanho das instâncias;
- Mostrar a importância do uso de uma codificação adequada. A codificação utilizada neste trabalho, Conjunto de Arestas gera apenas soluções válidas, possui alta localidade e hereditariedade, unicidade durante o processo de codificação/decodificação;
- Exploração da representação por Conjunto de Arestas aplicados ao problema de *dc-MST* como proposto por Raidl [55] e estendido para grafos orientados e *One-Max Tree Problem*.
- Desenvolvimento dos operadores de *crossover* e mutação baseados no problema de *dc-MST*, otimização do operador de *crossover* utilizando a idéia de consenso entre árvores.
- AE para filogenia com estruturas de dados (codificação) eficientes;
- AE para filogenia utilizando parcimônia de Fitch para caracteres de *DNA*. Até então, na literatura foram encontrados trabalhos com a parcimônia de Wagner para caracteres binários;
- AE capaz de gerar um conjunto de filogenias com igual ou melhor parcimônia a partir da árvore filogenética encontrada pelo *NJ* ou *PHYLIP*;
- Algoritmo de filogenia capaz de encontrar árvores filogenéticas para um grande número de seqüências.

8.2 Perspectivas futuras

A seguir são apresentadas sugestões para a continuação da pesquisa:

- Explorar mais algumas estratégias em Computação Evolutiva, tais como: tornar o algoritmo auto-adaptativo, considerar várias populações e utilizar migração entre elas. Além disso, otimizar cada operador do AE associando busca local. Tais estratégias mostraram-se eficientes na busca pela árvore filogenética de melhor aptidão em trabalhos encontrados na literatura;
- Tornar o algoritmo mais rápido com a paralelização do AE. O AE possui a característica implícita de divisão de tarefas. A grosso modo, considerando-se uma população de soluções pode-se dividi-la e cada processador evoluir as subpopulações em paralelo;
- Aprofundar os conceitos biológicos e aplicar heurísticas relacionadas. O objetivo principal do trabalho foi voltado à parte computacional do problema. Seria interessante explorar mais as propriedades biológicas para produzir heurísticas que melhorem o desempenho do algoritmo;
- Utilizar como função de avaliação outros métodos como o de Máxima Verossimilhança testando os vários modelos evolutivos existentes;
- Implementação de interface gráfica oferecendo um programa mais amigável, tornando acessível a usuários com diversas formações. O programa foi desenvolvido na plataforma linux onde a execução é realizada via linha de comando.

Bibliografia

- [1] Alves, J. M. P. “*Caracterização e Filogenia Moleculares de Acanthamoeba*”. Tese de Doutorado, Instituto de Ciências Biomédicas-USP, 2001.
- [2] Ayala, F. J.; Valentine, J.W. “*Evolving: The Theory and Processes of Organic Evolution*”, The Benjamin/Cummings Publ. Comp., Califórnia, Capítulo 1, pp. 1-17, 1979.
- [3] Brauer, M. J.; Holder, M. T.; Dries, L. A.; Zwickl, D.J.; Lewis, P. O.; Hillis, D. M. “*Genetic Algorithms and Parallel Processing in Maximum-likelihood Phylogeny Inference*”, Molecular Biology and Evolution 19:1717-1726, 2002.
- [4] Carvalho, P. M. S.; Ferreira, L. A. F. M.; Barruncho, L. M. F. “*On Spanning-Tree Recombination in Evolutionary Large-Scale Network Problems - Application to Electrical Distribution Planning*”, IEEE Transactions on Evolutionary Computation, 5(6):623-630, 2001.
- [5] Congdon, C. B. “*Gaphyl: A Genetic Algorithms Approach to Cladistics*”, Principles and Practice of Knowledge Discovery in Databases (PKDD 2001), Freiburg, Germany, 2001. In L. DeRaedt, A. Siebes (eds.), Principles of Data Mining and Knowledge Discovery, Lecture Notes in Computer Science, 2168:67-78, Springer-Verlag, Berlin, 2001.
- [6] Cormen, T.; Leiserson, C.; Rivest, R. “*Introduction to Algorithms*”. McGraw-Hill, 1990.
- [7] Cotta, C.; Moscato, P. “*Inferring Phylogenetic Trees Using Evolutionary Algorithms*”, Parallel Problem Solving From Nature VII, J.J. Merelo et al. (eds.), Lecture Notes in Computer Science. 2439:720-729. Springer-Verlag, Berlin, 2002.

- [8] Crandall, K. A. *"The Evolution of HIV"*. Ed. John Hopkins University Press, Baltimore, 1999.
- [9] Day, W. H. E. *"Computational Complexity of Inferring Phylogenies from Dissimilarity Matrices"*, Bulletin of Mathematical Biology, 49:461-467, 1987.
- [10] Deb, K. *"Multi-Objective Optimization using Evolutionary Algorithms"*. John Wiley, Chichester, UK, 2001.
- [11] De Jong, K.; Fogel, D. B.; Schwefel, H. -P. *"A History of Evolutionary Computation"*, Evolutionary Computation1: Basic Algorithms and Operators, Bäck. T. et al. (eds.), Capítulo 6, pp.40-58, IOP Publ., Bristol. 2000.
- [12] Delbem, A. C. B.; Carvalho, A. C. P. L. F. *"A Forest Representation for Evolutionary Algorithms Applied to Network Design"*, In Proceedings of the 2003 Genetic and Evolutionary Computation Conference, 2723:634-635, Springer-Verlag, 2003.
- [13] Delbem, A. C. B.; Carvalho, A. C. P. L. F.; Policastro, C.; Pinto, A. K. O.; Garcia, A.; Honda, K. *"Node-depth Encoding for Evolutionary Algorithms Applied to Network Design"*, Genetic and Evolutionary Computation Conference, 2004.
- [14] Delbem, A. C. B.; Policastro, C.; Carvalho, A. C. P. L. F.; Honda, K.; Ogata, A. K. O. *"Node-depth Encoding Applied to the Degree-Constrained Minimum Spanning Tree"*, Anais do VIII Simpósio Brasileiro de Redes Neurais, SBRN'04, Los Alamitos: IEEE Computer Press, 2004.
- [15] Delbem, A. C. B.; Pinto, A. K. O.; Honda, K.; Lima, T. W. *"Evolutionary Algorithm Using Node-depth Representation for Network Design"*, submetido para Evolutionary Computation, 2005.
- [16] Durbin, R.; Eddy, S.R.; Krogh, A.; Mitchison, G. *"Biological Sequence Analysis. Probabilistic Models of Proteins and Nucleic Acids"*, Cambridge University Press, 1998.
- [17] Eisen, J. A. *"Phylogenomics: Improving Functional Predictions for Uncharacterized Genes by Evolutionary Analysis"*, Genome Research 8(3):163-167, 1998.
- [18] Farris, J. S. *"Methods for Computing Wagner Trees"*, Systematic Zoology, 19:83-92, 1970.

Bibliografia

- [19] Felsenstein, J. “*Evolutionary Trees from DNA Sequences: A Maximum Likelihood Approach*” *Journal of Molecular Evolution*, 17(6):368-76, 1981.
- [20] Felsenstein, J. “*PHYLIP (Phylogeny Inference Package) version 3.5c*”, Department of Genetics, University of Washington, Seattle, 1993. Disponível em <http://evolution.genetics.washington.edu/phylip>.
- [21] Felsenstein, J. “*The number of evolutionary trees*”, *Syst. Zool.*, 27: 27-33, 1978.
- [22] Felsenstein, J. “*Phylogeny I: Parsimony and Tree Space*”, Lecture Note, 1998.
- [23] Fogel, L. J.; Owens, A. J.; Walsh, M. J. “*Artificial Intelligence through Simulated Evolution*”, Wiley, 1966.
- [24] Futuyma, D. J. “*Biologia Evolutiva*, Ed. Sociedade Brasileira de Genética, Ribeirão Preto, SP, 1992.
- [25] Gen, M.; Cheng, R. “*Genetic Algorithms and Engineering Design*”, Ashikaga Institute of Technology, Ashikaga, Japan, 1997.
- [26] Goldberg, D. E. “*Genetic Algorithms in Search, Optimization and Machine Learning*”, Addison-Wesley, 1989.
- [27] Gottlieb, J.; Julstrom, B. A.; Rothlauf, F.; Raidl, G. R. “*Prüfer Numbers: A Poor Representation of Spanning Trees for Evolutionary Search*”, In *Proceedings of the 2001 Genetic and Evolutionary Computation Conference*, 343-350. Morgan Kaufmann, 2001.
- [28] Graham, R. L.; Hell, P. “*On the History of the Minimum Spanning Tree Problem*”, *Annals of the History of Computing*, 7:43-57, 1985.
- [29] Harvey, P. H.; Brown, A. J. L.; Smith, J. M.; Nee, S. “*New Uses for New Phylogenies*”, Oxford University Press, Oxford, 1996.
- [30] Hasegawa M, Kishino H, Yano T. “*Dating of the Human-ape Splitting by a Molecular Clock of Mitochondrial DNA*”, *Journal of Molecular Evolution*, 22(2):160-74, 1985.
- [31] Jukes T. H.; Cantor C. R. “*Evolution of Protein Molecules*”. In *Mammalian Protein Metabolism*, Academic Press, New York, 1969.

- [32] Katoh, K.; Kuma, K-I.; Miyata, T. “*Genetic Algorithm-Based Maximum-Likelihood Analysis for Molecular Phylogeny*”, Journal of Molecular Evolution, 53:477-484, 2001.
- [33] Kimura, M. “*A Simple Method for Estimating Evolutionary Rates of Base Substitutions through Comparative Studies of Nucleotide Sequences.*”, Journal of Molecular Evolution, 16(2):111-20, 1980.
- [34] Knowles, J.; Corne, D. “*A New Evolutionary Approach to the Degree-Constrained Minimum Spanning Tree Problem*”, IEEE Transactions on Evolutionary Computation, 4:125-134, 2000.
- [35] Kumar, S.; Filipski, A. “*Molecular Phylogeny Reconstruction*”, Encyclopedia of Life Sciences, Macmillan Reference Ltda, Oxford, UK, 2001.
- [36] Lemmon, A. R.; Milinkovitch, M. C. “*The Metapopulation Genetic Algorithm: An Efficient Solution for the Problem of Large Phylogeny Estimation*”, Proceedings of the National Academy of Sciences USA, 99(16):10516-10521, 2002.
- [37] Lesk, A. M. “*Alignments and Phylogenetics Trees*”, Introduction to Bioinformatics, Capítulo 4, pp.154-206, Oxford University Press, 2002.
- [38] Lewis, P. “*A Genetic Algorithm for Maximum Likelihood Phylogeny Inference Using Nucleotide Sequence Data*”, Molecular Biology and Evolution, 15:277-283, 1998.
- [39] Lipscomb, D. “*Basics of Cladistic Analysis*”, Washington, GWU, 1998.
- [40] Matsuda, H. “*Protein Phylogenetic Inference Using Maximum Likelihood with a Genetic Algorithm*”, Pacific Symposium on Biocomputing, 512-523, 1996.
- [41] Mattioli, S. R. “*Biologia Molecular e Evolução*”, Holos Editora, 2001.
- [42] Michalewicz, Z. “*Genetic Algorithms + Data Structures = Evolution Programs*”, Springer-Verlag, 1996.
- [43] Mitchell, M. “*An Introduction to Genetic Algorithms*”, MIT Press Massachusetts, 1996.
- [44] Mitchell, M.; Taylor, C. E. “*Evolutionary Computation: An Overview*”, Annual Review of Ecology and Systematics, 20:593-616, 1999.

- [45] Moilanen, A. "Searching for most Parsimonious Trees with Simulated Evolutionary Optimization", *Cladistics*, 15:39-50, 1999.
- [46] Nei, M.; Saitou, N. "The Neighbor-Joining Method: A New Method for Reconstructing Phylogenetic Trees", *Molecular Biology and Evolution*, 4:406-425, 1987.
- [47] Nei, M.; Kumar, S. "Molecular Evolution and Phylogenetics", Oxford University Press, 2000.
- [48] Page, R. D. M.; Holmes, E. C. "Molecular Evolution: A Phylogenetic Approach", Blackwell Science, 1998.
- [49] Palmer, C.; Kershenbaum, A. "Representing Trees in Genetic Algorithms", *Handbook of Evolutionary Computation*, Oxford University Press, 1997.
- [50] Palmer, C.; Kershenbaum, A. "An Approach to a Problem in Network Design Using Genetic Algorithms", *Networks*, 26:151-163, 1995.
- [51] Prado, O. G. "Computação Evolutiva Empregada na Reconstrução de Árvores Filogenéticas", *Dissertação de Mestrado*, UNICAMP, 2001.
- [52] Prado, O. G.; Von Zuben, F.J. "Reconstrução de Árvores Filogenéticas Usando Algoritmos Evolucionários", *Anais do XXII Congresso da Sociedade Brasileira de Computação - SBC'2002*, 1:405-415, Florianópolis, SC, Brasil, 2002.
- [53] Prosdocimi, F. *et al.* "Bioinformática: Manual do Usuário", *Biotecnologia: Ciência e Desenvolvimento*, 29:12-25, 2002.
- [54] Raidl, G. R.; Julstrom, B.A. "Edge-Sets: An Effective Evolutionary Coding of Spanning Trees", *Technical Report TR-186-1-01-01*, Institute of Computer Graphics and Algorithms, Vienna University of Technology, 2001.
- [55] Raidl, G. R. "An Efficient Evolutionary Algorithm for the Degree-constrained Minimum Spanning Tree Problem", In *Proceedings of the 2000 Congress on Evolutionary Computation CEC00*, pp 104-111, IEEE Press, California, USA, 2000.
- [56] Reijmers, T. H.; Wehrens, L. M.; Buydens, L. M. C. "Quality Criteria of Genetic Algorithms for Construction of Phylogenetic Trees", *Journal of Computational Chemistry*, 20(8):867-876, John Wiley & Sons, 1999.

- [57] Ridley, M. "*Evolution*", Blackwell Science, 1996.
- [58] Rokas A.; Williams B. L.; King N.; Carroll S. B. "*Genome-scale Approaches to Resolving Incongruence in Molecular Phylogenies*", *Nature*, 425(6960):798-804, 2003.
- [59] Rothlauf, F.; Goldberg, D.; Heinzl, A. "*Network Random Keys - A Tree Network Representation Scheme for Genetic and Evolutionary Algorithms*", *Evolutionary Computation*, 10(1):75-97, 2002.
- [60] Saitou, N.; Imanishi, T. "*Relative Efficiencies of the Fitch-Margoliash, Maximum-Parsimony, Maximum-Likelihood, Minimum-Evolution, and Neighbor-Joining Methods of Phylogenetic Tree Construction in Obtaining the Correct Tree*", *Molecular Biology and Evolution*, 6(5):514-525, 1989.
- [61] Setubal, J.; Meidanis, J. "*Introduction to Computational Molecular Biology*", PWS Publishing Company, Boston, 1997.
- [62] Schindler, B.; Rothlauf, F.; Pesch, H. -J. "*Evolution Strategies, Network Random Keys, and the One-Max Tree Problem*", *Applications of Evolutionary Computing, Proceedings of EvoWorkshops2002: EvoCOP, EvoIASP, EvoSTim*, 2279:141-150, Springer-Verlag, 2002.
- [63] Skourikhine, A. "*Phylogenetic Tree Reconstruction Using Self-adaptive Genetic Algorithm*", *Proceedings IEEE International Symposium on Bio-Informatics and Biomedical Engineering*, 129-134, 2000.
- [64] Sourdis, J.; Nei, M. "*Relative Efficiencies of the Maximum Parsimony and Distance-Matrix Methods in Obtaining the Correct Phylogenetic Tree*", *Molecular Biology and Evolution* 5:(3)298-311, 1988.
- [65] Tateno, Y.; Takezaki, N.; Nei, M. "*Relative Efficiencies of the Maximum-likelihood, Neighbor-joining, and Maximum-parsimony Methods when Substitution Rate Varies with Site*", *Molecular Biology and Evolution*, 11(2):261-277, 1994.
- [66] Von Zuben, F. J. "*Computação Evolutiva: Uma Abordagem Pragmática*", *Anais da 1ª Jornada de Estudos em Computação de Piracicaba e Região (1ª JECOMP)*, (1):25-45, 25 a 29 de setembro, 2000.

Bibliografia

- [67] Wareham, H. T. “*On the Computational Complexity of Inferring Evolutionary Trees*”, Technical Report 93-01, Department of Computer Science, Memorial University of Newfoundland, Canada, 1993.
- [68] Weir, B. S. “*Genetic Data Analysis II*”, Sinauer, Sunderland, MA, 1996.
- [69] Wiley, E. O.; Siegel-Causey, D.; Brooks, D. R.; Funk, V.A. “*The Compleat Cladist: A Primer of Phylogenetic Procedures*”, University of Kansas Museum of Natural History, Special Publication n^o. 19, Lawrence, Kansas, 1991.
- [70] Yang, Z. “*PAML: A Program Package for Phylogenetic Analysis by Maximum Likelihood*”, CABIOS. 13:555-556, 1997.
- [71] Zaha, A. (coordenador) “*Biologia Molecular Básica*”, Mercado Aberto, Porto Alegre, RS, 2000.
- [72] “*Dictionary of Algorithms and Data Structures*”, National Institute of Standards and Technology (NIST), 2003. (web site: <http://www.nist.gov/dads/>, acessado em 25 de fevereiro de 2004).
- [73] <http://www.bioss.sari.ac.uk/frank/newton/>, acessado em 13 de março de 2005.
- [74] <http://ftp.cse.sc.edu/bioinformatics/data/paullewisdata/>, acessado em 13 de março de 2005.
- [75] <http://www.stats.ox.ac.uk/mcvean/DTC/SEQ/?C=N&O=D>, acessado em 10 de agosto de 2005.

Bibliografia

Apêndice

Teoria dos Grafos

A seguir são descritos alguns conceitos que são utilizados no presente texto envolvendo a Teoria dos Grafos, baseados nas definições contidas em “*Dictionary of Algorithms and Data Structures*” [72]:

Um grafo G pode ser definido como um par (V, E) , onde V é um conjunto de nós (ou vértices) e E representa um conjunto de pares de nós, chamados de arestas. Se u e v são nós de um grafo e se o par $\{u, v\}$ é uma aresta denotada por e , diz-se que e conecta (ou é adjacente a) u e v . O grau de um nó é o número de arestas adjacentes a este.

Um grafo pode ser orientado ou não-orientado. Em um grafo dirigido, a ordem entre os nós de uma aresta (u, v) é importante. Esta aresta é diferente da aresta (v, u) e é representada por uma flecha de u para v . Em um grafo não orientado, a adjacência entre as arestas são simétricas, ou seja, $(u, v) = (v, u)$.

Um grafo é completo se existe uma aresta entre quaisquer dois nós de um grafo. Um grafo completo com n nós é denotado por K_n . Se o número de arestas em um grafo é muito menor do que a quantidade de arestas pertencentes ao grafo completo com mesma quantidade de nós, tal grafo é considerado esparso. Caso o conjunto de arestas em um grafo for vazio, então diz-se que o grafo é vazio.

Um subgrafo é um grafo cujos nós e arestas são subconjuntos de outro grafo.

Um caminho em um grafo é uma sequência finita de nós conectados por arestas, onde todas as arestas são distintas. Se somente os nós de início e término são iguais, então diz-se que o caminho é cíclico. Um grafo que possui um ou mais caminhos cíclicos é denominado grafo cíclico. Um grafo acíclico é um grafo que não contém ciclos.

Um grafo é conexo quando para cada dois nós sempre existe um caminho entre eles.

Uma árvore é um grafo acíclico e conexo. É comum definir um dos nós de uma árvore como sendo o nó raiz. Em geral, este nó funciona como uma referência de onde se inicia a árvore. Um nó raiz possui grau maior ou igual a 1. Se um nó possui grau 1, e não é o nó raiz então este é chamado de nó terminal (ou folha). Uma árvore é dita binária se possuir no máximo dois filhos por nó.

Um grafo formado por uma ou mais árvores é chamado de floresta.

Uma árvore geradora é um subgrafo conexo, acíclico contendo todos os nós de um grafo.