Resumo

Abstract

Apostila de Algoritmos de Estimação de Distribuição

11 de fevereiro de 2011

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução e Motivação

Os Algoritmos de Estimação de Distribuição derivam dos Algoritmos Genéticos, ramo da Computação Bioinspirada que visa simular a evolução e seleção natural teorizadas por Charles Darwin em seu Livro 'A Origem das Espécies" [3]. De maneira resumida, a teoria de Darwin diz que uma espécie evolui baseada na capacidade de seus indivíduos sobreviverem ao meio e se reproduzirem. Os indivíduos com maior capacidade de sobrevivência têm mais chance de chegar à idade reprodutiva e perpetuar seus genes, fazendo com que a próxima geração seja em média mais parecida com os melhores indivíduos da atual e, portanto, com mais indivíduos adaptados ao meio.

Para que se possa modelar problemas computacionais baseados na teoria da evolução, alguns paralelos devem ser traçados entre as estruturas computacionais e eventos naturais. O algoritmo baseia-se na criação de uma população e evolução da mesma através de várias gerações com cruzamentos e mutações.

Primeiro, imaginemos o funcionamento da seleção natural. Inicialmente há uma população cujos indivíduos são bem distintos entre si, devido a variação de seu material genético. O ambiente traz riscos e dificuldades a todos esses indivíduos, fazendo com que muitos deles não cheguem à idade reprodutiva.

Pode-se assumir que os sobreviventes têm características genéticas que os ajudaram a sobreviver. Portanto, quando os indivíduos chegam à idade reprodutiva, a população está refinada, ou seja, foi reduzida aos indivíduos com mais chance de sobrevivência daquela geração. Para efeitos didáticos, assumiremos que cada cromossomo de cada indivíduo tem 'genes positivos", que os ajudam a sobreviver e "genes negativos", que não os ajudam ou até

atrapalham sua sobrevivência.

Os sobreviventes de cada geração irão se reproduzir, criando uma nova geração cujo material genético é uma combinação do material da anterior. Também deve-se levar em conta que existe uma chance de que os novos indivíduos sofram alguma mutação (erros de cópia de material genético ou mudanças causadas por fatores externos como radiação, mutagênicos químicos ou vírus). Cada indivíduo da nova geração recebe metade do material genético de cada um de seus pais de maneira aleatória. Essa combinação dos genes dos pais não resulta apenas na seleção das melhores características, mas também na criação de novas características vindas de diferentes combinações de genes, sendo que essas novas características podem ser benéficas ou não.

Com isso, a nova geração passará pela mesma seleção natural, porém os indivíduos selecionados terão mais "genes positivos"que a geração anterior e esse processo será repetido. No ambiente natural, nunca se chega a uma população "perfeita", devido a vários fatores externos, porém, imaginado um ambiente ideal e controlado, eventualmente toda a população estaria completamente adaptada ao meio.

1.1 Algoritmos Genéticos

Baseado no processo de seleção natural descrito na Seção anterior, nesta Seção serão apresentadas as estruturas e procedimentos dos Algoritmos Genéticos traçando-se paralelos com o ambiente natural.

A princípio deve-se notar que os Algoritmos Genéticos são essencialmente algoritmos de otimização, ou seja, procuram encontrar uma combinação de variáveis que resultará em um valor ótimo para uma dada função. Fazendo uma analogia à seleção natural em um ambiente natural, pode se dizer que o problema tratado é o da sobrevivência de uma população em um meio ambiente, onde a solução é representada por um indivíduo apto a sobrevier no ambiente em questão e a função a ser otimizada (chamada de função de *fitness*) mede a capacidade de sobrevivência dos indivíduos nesse ambiente.

O indivíduo é caracterizado por seus genes. Cada uma das variáveis de entrada da função de *fitness* será considerada um gene e chamaremos o conjunto de genes do indivíduo de cromossomo. Para emular a seleção natural mecanismos são implementados para que os indivíduos com maior valor de *fitness* tenham maior chance de se reproduzir.

A geração de novos indivíduos nos Algoritmos Genéticos é modelada

através de operadores de reprodução como *crossover* e mutação. No *crossover*, indivíduos sobreviventes são agrupados em casais que geram descendentes através da combinação de seus genes. Cada gene dos descendentes tem uma pequena chance de ser alterado pela mutação.

Os indivíduos da nova geração são testados novamente pela função de *fitness* e o procedimento é repetido até que um critério de parada pré-estabelecido seja atingido. Um exemplo de critério de parada é a verificação de que o valor ótimo da função de *fitness* foi encontrado.

1.1.1 Funcionamento do Algoritmo Genético

Nas próximas seções, serão detalhadas as etapas de um Algoritmo Genético clássico, cujo fluxograma é ilustrado na Figura 1.1. Ao mesmo tempo, um Algoritmo Genético simples será construído, ilustrando essas etapas. Para ilustrar o funcionamento de cada etapa, será utilizado o problema *OneMax* [5] como exemplo. O problema em questão consiste em, dado um vetor de n elementos binários, maximizar o número de *1*'s. Portanto, dado um vetor binário de genes $X = \{x_1, x_2, \cdots, x_n\}$, a função de *fitness* é:

$$f(X) = \sum_{i=1}^{n} x_i$$

No exemplo será utilizada uma população de 15 indivíduos com 6 genes por cromossomo.

1.1.2 Gerar População Inicial

Geralmente a população inicial é gerada aleatoriamente, mas algumas implementações de Algoritmos Genéticos aproveitam a população inicial para incluir algumas soluções boas previamente conhecidas. Deve-se observar que o tamanho da população influencia diretamente no desempenho do AG, sendo que o tamanho ideal depende de diversos fatores, tais como espaço de busca do problema, e número de variáveis relacionadas. Na Tabela 1.1 é mostrado um exemplo de população aleatória para o problema *OneMax*.

1.1.3 Avaliar População

Nesta etapa, todos os indivíduos da geração atual são avaliados e os critérios de parada do AG são checados. É armazenado o valor da função de *fitness*

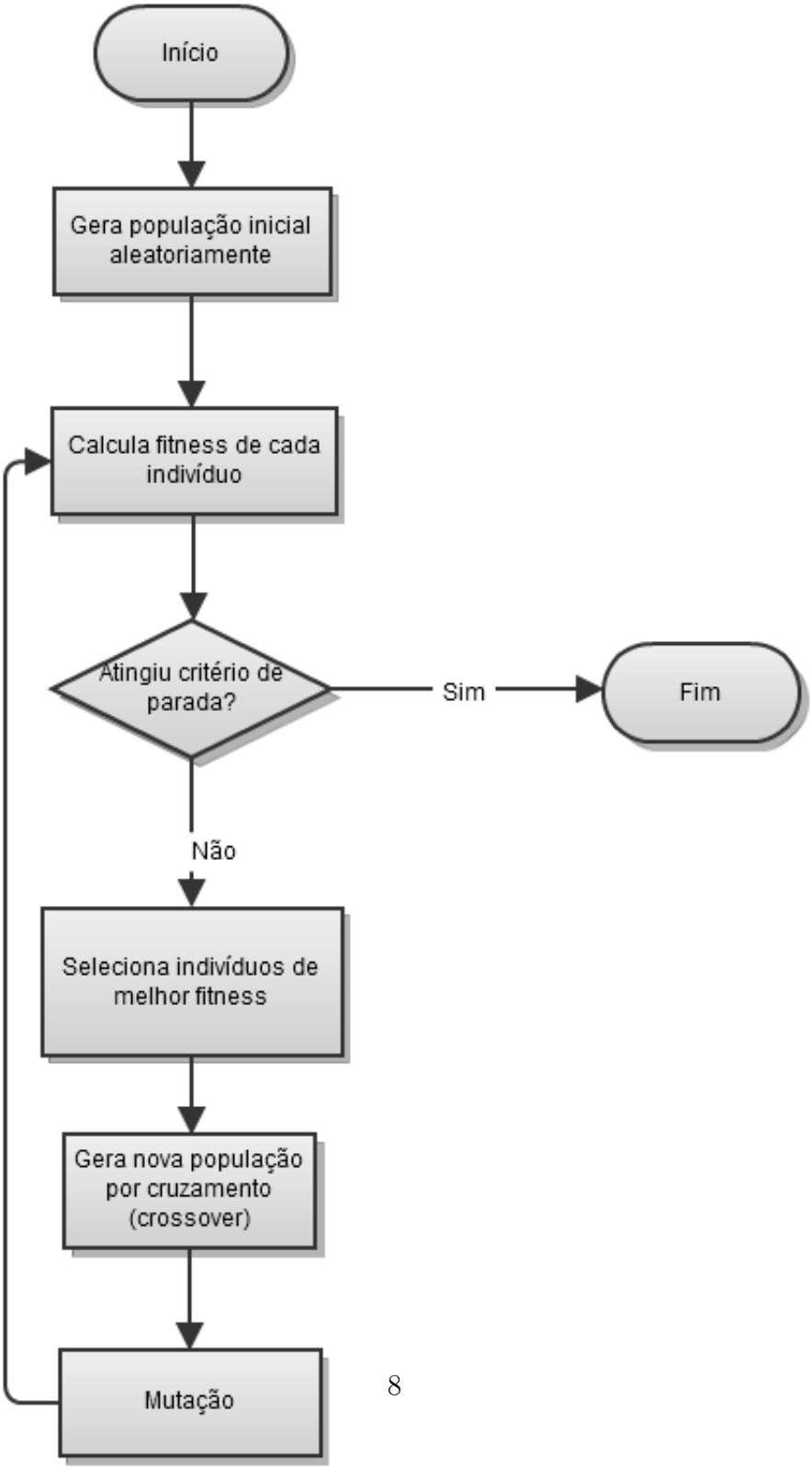


Figura 1.1: Fluxograma de um Algoritmo Genético

indivíduo	cromossomo
1	1 1 0 1 1 1
2	0 1 1 1 0 1
3	1 1 0 1 0 1
4	0 1 1 0 0 1
5	0 1 0 1 1 0
6	0 1 0 1 0 0
7	0 0 0 1 0 1
8	1 1 1 0 1 1
9	1 0 1 1 0 1
10	0 1 1 1 0 0
11	0 1 0 0 0 1
12	1 1 0 1 0 1
13	1 1 1 0 0 0
14	1 1 1 1 1 0
15	0 0 0 0 0 1

Tabela 1.1: Exemplo de população inicial

para cada um dos cromossomos. Os critérios de parada podem variar dependendo da aplicação desejada. Algumas possibilidades são: Valor ótimo atingido, número máximo de gerações atingido ou *fitness* do melhor indivíduo da população igual ao *fitness* médio da população.

O resultado do uso dos indivíduos da população inicial do exemplo apresentado como entradas para a função de *fitness* - no caso do exemplo, a função *OneMax*, apresentada anteriormente - é mostrado na Tabela 1.2. Note que o maior *fitness* da população inicial é *5* e o *fitness* médio é *3,3*. Como no caso do problema *OneMax* o valor ótimo é conhecido, é possivel utilizar esse valor como critério de parada. Quando um indivíduo da população atinge o *fitness* máximo (*6*, no caso do exemplo), ele é retornado como solução para o problema. No entanto, muitas vezes o valor ótimo não é conhecido, portanto outro critério deve ser adotado. A diferença entre o maior *fitness* da população e o *fitness* médio costuma ser um bom critério, pois quando esses valores se igualam, a população provavelmente está geneticamente uniforme, o que significa que a probabilidade de evolução é baixa, já que esta depende apenas da mutação.

indivíduo	cromossomo	*fitness*
1	1 1 0 1 1 1	5
2	0 1 1 1 0 1	4
3	1 1 0 1 0 1	4
4	0 1 1 0 0 1	3
5	0 1 0 1 1 0	3
6	0 1 0 1 0 0	2
7	0 0 0 1 0 1	2
8	1 1 1 0 1 1	5
9	1 0 1 1 0 1	4
10	0 1 1 1 0 0	3
11	0 1 0 0 0 1	2
12	1 1 0 1 0 1	4
13	1 1 1 0 0 0	3
14	1 1 1 1 1 0	5
15	0 0 0 0 0 1	1

Tabela 1.2: *Fitness* de cada indivíduo

1.1.4 Seleção

Uma vez que todos os indivíduos foram avaliados, precisa-se decidir quais deles se reproduzirão construindo a próxima geração. Existem vários métodos de seleção, tais como o truncamento (onde os indivíduos são ordenados por *fitness* e um grupo de número arbitrário com os melhores cromossomos é escolhido), ou a seleção por torneio, onde alguns pares de cromossomos são escolhidos aleatoriamente e o melhor de cada par é escolhido.

Utilizando uma seleção por torneio com quatro disputas na população inicial do exemplo, os indivíduos *1*, *4*, *6*, *8*, *9*, *11*, *13* e *15* são escolhidos. Comparando os *fitness* dois a dois, como ilustrado na Tabela 1.3, os indivíduos *1*, *8*, *9* e *13* são escolhidos para se reproduzir, criando a próxima geração.

1.1.5 Crossover

O *Crossover* é o cruzamento, o procedimento de reprodução que combinando os genes dos indivíduos selecionados, criará uma nova geração. Para se fazer isso, os indivíduos selecionados da geração atual são tomados dois a dois

disputa	indivíduo	cromossomo	*fitness*	vencedor
1	1	110111	5	1
	4	011001	3	
2	6	010100	2	8
	8	111011	5	
3	9	101101	4	9
	11	010001	2	
4	13	111000	3	13
	15	000001	1	

Tabela 1.3: Seleção por Torneio

e seus genes são combinados, formando os indivíduos da próxima geração. Uma maneira de se fazer isso é o *crossover* ponto a ponto, em que cada gene de cada indivíduo da nova geração recebe 50% de chance de vir de cada um de seus pais. Esse procedimento é repetido até que todos os indivíduos da nova geração sejam criados, sendo que cada casal selecionado é responsável por gerar uma fração igual da nova geração.

Na Figura 1.2, é mostrado um exemplo de *crossover* entre o casal formado pelos indivíduos *9* e *13*, selecionados no torneio da etapa anterior.

1.1.6 Mutação

O procedimento de mutação pode ser executado tanto durante a geração dos novos indivíduos quanto na população já gerada. A mutação resume-se a se dar uma chance, geralmente baixa, de que um gene se altere espontaneamente. Ou seja, para cada gene de cada indivíduo, existe uma probabilidade de que o mesmo se altere, o que faz que a população continue evoluindo mesmo nos casos onde sem a mutação ela teria convergido para um valor sub-ótimo. No entanto, como essa chance é muito baixa, é possível que, no caso da população ter convergido para um valor sub-ótimo, seja necessário um número impraticavelmente grande de gerações para que a mutação faça a função convergir para o valor ótimo.

A Figura 1.3 mostra um cromossomo sofrendo mutação em seu 4º gene. No caso deste exemplo, a mutação foi benéfica, ou seja, após a mutação, o *fitness* do cromossomo em questão melhorou.

A Tabela 1.4 mostra uma nova geração criada após o cruzamento dos

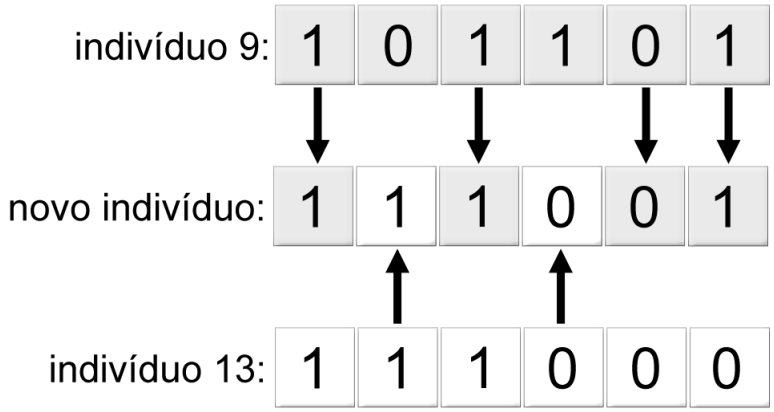


Figura 1.2: Exemplo de *crossover*

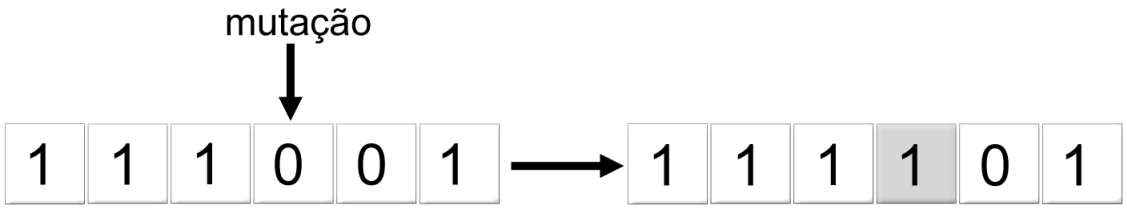


Figura 1.3: Cromossomo sofrendo mutação

indivíduos selecionados e mutação dos novos indivíduos. O valor médio de *fitness* da nova população é *4.4* e o *fitness* máximo é *6*. Como *6* é o valor ótimo da função, um dos critérios de parada é atingido e o processo encerrado.

1.2 Algoritmo Genético Compacto

Um dos problemas do Algoritmo Genético apresentado anteriormente é a grande quantidade de memória ocupada por suas populações. O *cGA* (do inglês, *Compact Genetic Algorithm*), visa resolver esse problema ao utilizar um modelo probabilístico para representar a população [6]. O funcionamento do *cGA*, apresentado na Figura 1.4, embora similar ao de um algoritmo genético tradicional, apresenta modificações necessárias para se lidar com a representação probabilística da população.

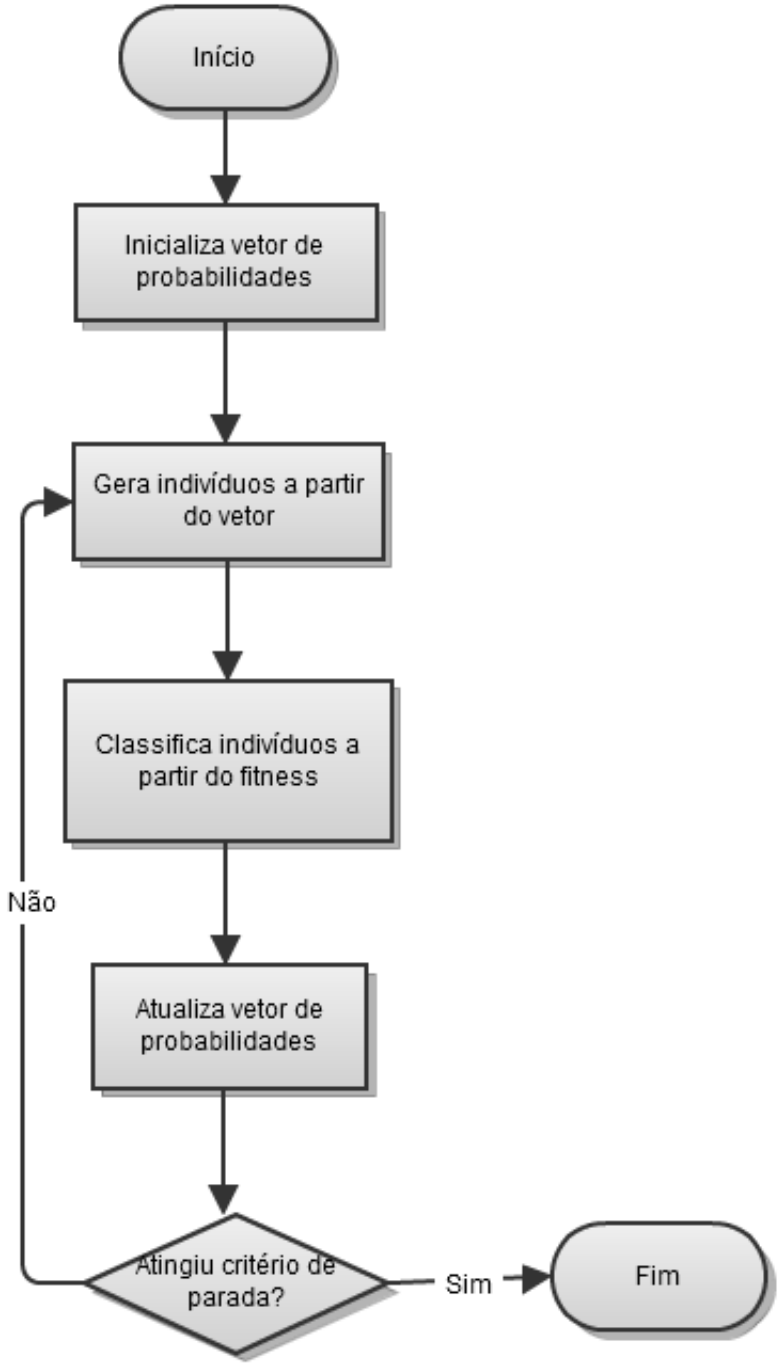


Figura 1.4: Fluxograma de um cGA

indivíduo	cromossomo	*fitness*
1	1 1 1 1 1 1	6
2	1 1 1 0 1 1	5
3	1 1 1 1 1 1	6
4	1 1 0 1 1 1	5
5	1 1 1 1 1 1	6
6	1 1 1 0 1 1	5
7	1 1 0 1 1 1	5
8	1 1 0 0 1 1	4
9	1 1 1 0 0 0	3
10	1 1 1 1 0 1	5
11	1 0 1 0 0 1	3
12	1 0 1 0 0 1	3
13	1 1 1 0 0 0	3
14	1 1 1 1 0 0	4
15	1 0 1 0 0 1	3

Tabela 1.4: Indivíduos da segunda geração e seus respectivos *Fitness*

O *cGA* também visa a otimização de uma função de *fitness*, sendo capaz de resolver os mesmos problemas que um algoritmo genético tradicional.

Para ilustrar o funcionamento do *cGA*, será utilizado novamente o exemplo do problema *OneMax*, sendo que neste caso o vetor $p[i]$ representará a probabilidade do $i-ésimo$ gene de um indivíduo de uma população ser igual a 1. Embora no *cGA* não seja gerada uma população inteira, ele utiliza um parâmetro n equivalente ao tamanho da população para atualizar os valores do vetor de probabilidades que a representa.

1.2.1 Representação e Geração da População

A população de um *cGA* é representada por um vetor de probabilidades, onde cada elemento do vetor corresponde à probabilidade do gene correspondente àquele elemento assumir um certo valor. No caso de um *cGA* para problemas com variáveis binárias, apresentado como exemplo nesta Seção, a probabilidade daquele gene assumir o valor 1. A partir deste vetor serão gerados os indivíduos a serem avaliados.

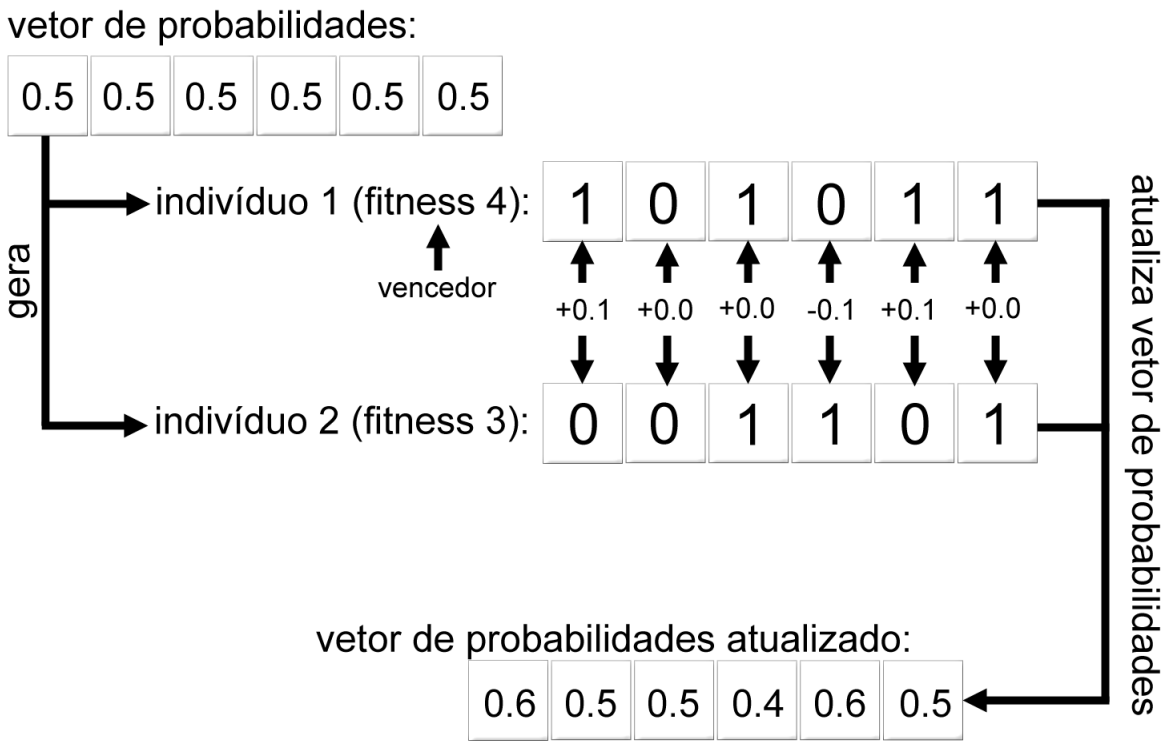


Figura 1.5: Atualização do vetor de probabilidades de um cGA

1.2.2 Geração e Avaliação de Indivíduos

Dois indivíduos são gerados a partir da probabilidade de cada gene armazenada no vetor que representa a população. Os valores de *fitness* de cada um desses indivíduos são comparados. Um a um, os genes do melhor indivíduo (maior *fitness* são comparados com os do pior). Caso esses genes sejam diferentes, o valor referente a eles no vetor de probabilidades é atualizado. Se o gene do melhor indivíduo for igual a 1, o valor do vetor de probabilidades é acrescido de $1/n$ e caso seja igual a 0, é decrescido de $1/n$, como ilustrado na Figura 1.5. Caso os genes sejam iguais, o valor referente a eles no vetor de probabilidades não é alterado.

1.2.3 Critérios de Parada

Como em um *cGA* a população é representada por um único vetor, a convergência depende apenas deste vetor. Cada posição do vetor é verificada. Caso todos elementos sejam iguais a 0 ou 1, houve convergência e o algoritmo é finalizado. Pode ser estabelecido um limiar para que a convergência seja atingida quando os elementos se aproximarem suficientemente de 0 ou 1. Outros critérios de paradas, como o número de iterações, podem ser adotados.

1.3 Por que AEDs?

Agora, dado o vetor binário $X = \{x_1, x_2, x_3\}$ consideremos um problema com a seguinte função de *fitness*:

$$f(x) = \begin{cases} 4, \text{se} \sum_{i=1}^{3} x_i = 0 \\ \\ \sum_{i=1}^{3} x_i, \text{caso contrário} \end{cases}$$

Esse problema, conhecido como *trap-3* [5] (do inglês, armadilha-3), faz parte de uma classe de problemas chamada "*deceptive problems*"(problemas enganosos). Esta classe de problemas é caracterizada por ter seu valor ótimo associado a uma combinação específica das variáveis, não podendo ser aproximado pelo ajuste individual das mesmas. Como é ilustrado no gráfico da Figura 1.6, o valor da função aumenta de acordo com número de *1*'s do cromossomo, aproximando-se de um valor sub-ótimo (*3*, no caso do exemplo),

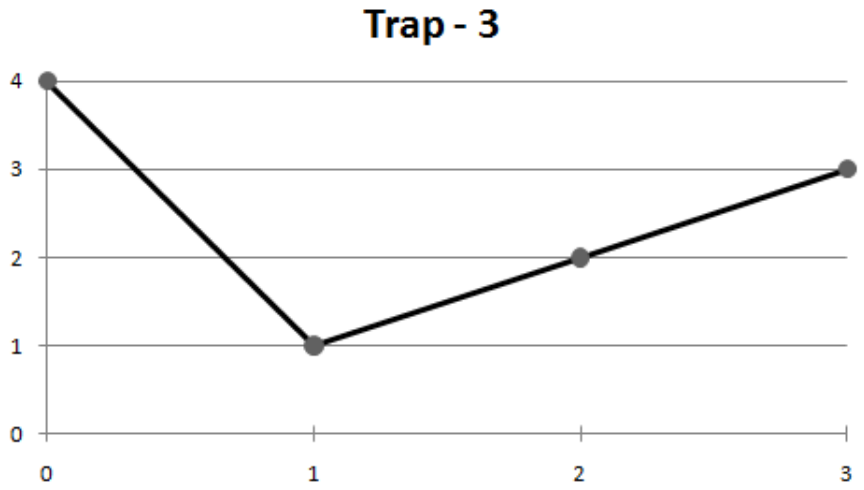


Figura 1.6: Gráfico do problema *trap-3*

enquanto seu valor ótimo *4* é atingido quando o cromossomo é formado apenas por *0*'s. Devido a esse comportamento, os *deceptive problems* enganam os algoritmos genéticos simples, pois a menos que o ótimo global apareça na população inicial, toda geração subsequente se aproximará mais do valor sub-ótimo.

Para tratar deste tipo de problema, foram criados os Algoritmos de Estimação de Distribuição. Esses algoritmos, embora mantenham parte da filosofia dos Algoritmos Genéticos, não analisam os genes individualmente. Os Algoritmos de Estimação de Distribuição visam encontrar quais variáveis são correlacionadas, ou seja, quais grupos de variáveis influenciam no resultado do problema em conjunto, o que os permite otimizar algumas funções enganosas.

Diversas estruturas diferentes, como vetores, árvores e grafos, podem representar a correlação entre as variáveis em um AED, sendo que a estrutura adotada varia de algoritmo para algoritmo. Nas próximas seções serão apresentados três AEDs distintos: O ECGA (*Extended Compact Genetic Algorithm*, ou Algoritmo Genético Compacto Estendido) [8], o BOA (*Baeysian Optimization Algorithm*, ou Algoritmo de Otimização Bayesiana) [7] e o ΦGA (*Philogenetic Algorithm*, ou Algoritmo Filogenético) [9], sendo que todos utilizam estruturas distintas entre si para modelar a correlação de variáveis.

O ECGA utiliza blocos de variáveis similares, chamados de *building blocks*

(blocos de construção). O BOA utiliza uma rede bayesiana, estrutura que permite identificação de correlações entre blocos de variáveis. Já o ΦGA utiliza uma árvore filogenética, que após uma poda identifica todos os *building blocks* permitindo que os ótimos globais sejam atingdidos em apenas uma iteração.

Capítulo 2

Algoritmo Genético Compacto Estendido (ECGA)

Para modelar as relações entre variáveis, o Algoritmo Genético Compacto Estendido (ECGA, do inglês "*Extended Compact Genetic Algorithm*") utiliza estruturas chamadas *building blocks*, que mantém as variáveis relacionadas entre si agregadas em grupos desconexos. Seu fluxograma é similar ao de um AG comum, como ilustrado na Figura 2.1. Os procedimentos de avaliação e seleção são idênticos aos do Algoritmo Genético mas os procedimentos de reprodução são bem diferentes. Ao invés de utilizar cruzamento e mutação para gerar a próxima população, o ECGA faz uma seleção dos melhores reprodutores e utiliza um modelo probabilístico baseado em uma tabela de frequências de *building blocks* para a geração da nova população. Esse procedimento será melhor detalhado nas próximas seções.

2.1 Identificando os building blocks

Os *building blocks* são estruturas que agregam as variáveis correlacionadas. Como inicialmente não se conhece nenhuma relação, a estrutura é inicializada com um bloco para cada variável. Para descobrir quais são as variáveis correlacionadas os building blocks e indivíduos selecionados são avaliados utilizando-se uma métrica chamada "Complexidade Combinada"(CC). Quanto melhor o agrupamento (ou quanto mais relacionadas forem as variáveis dentro de um grupo), menor é o valor da CC.

Para criar os *building blocks* é utilizada uma estratégia gulosa. Inicial-

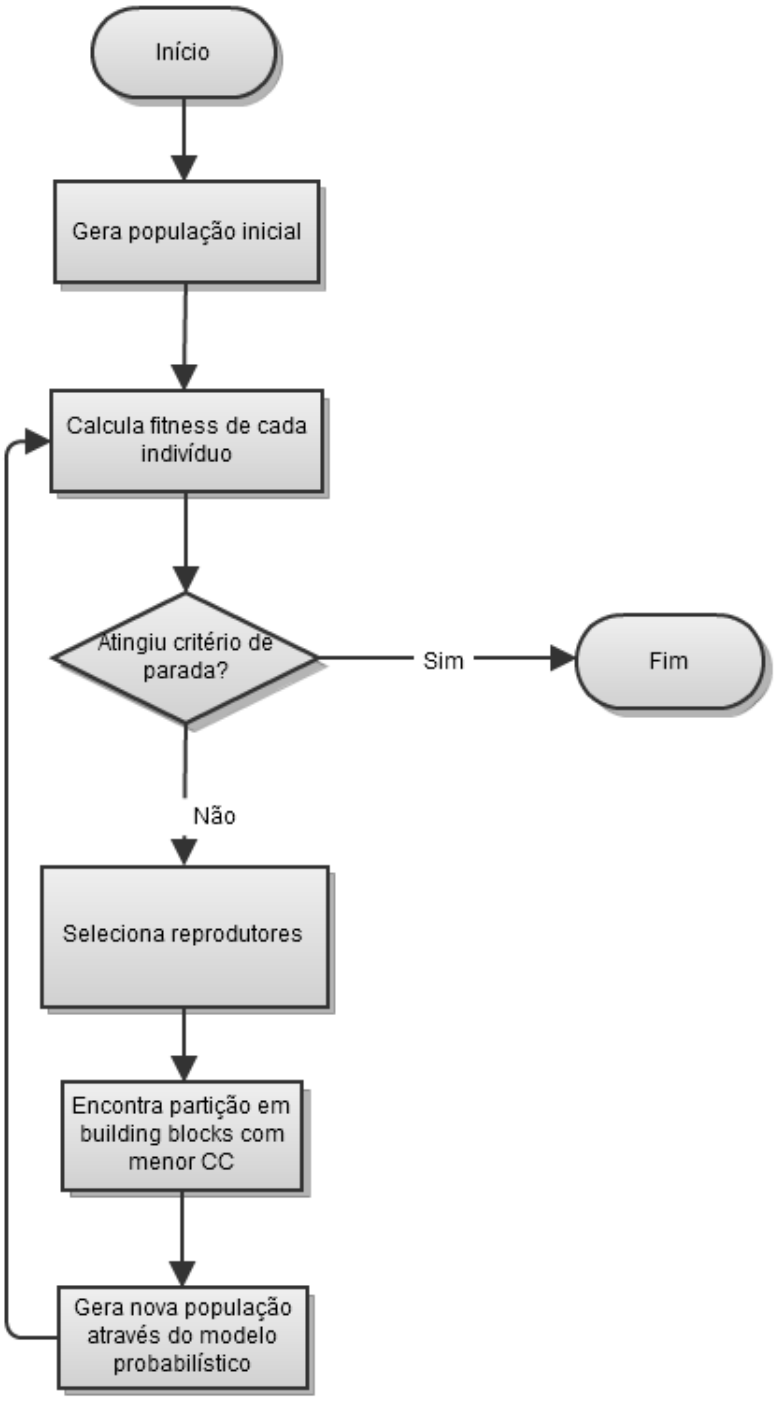


Figura 2.1: Fluxograma do ECGA

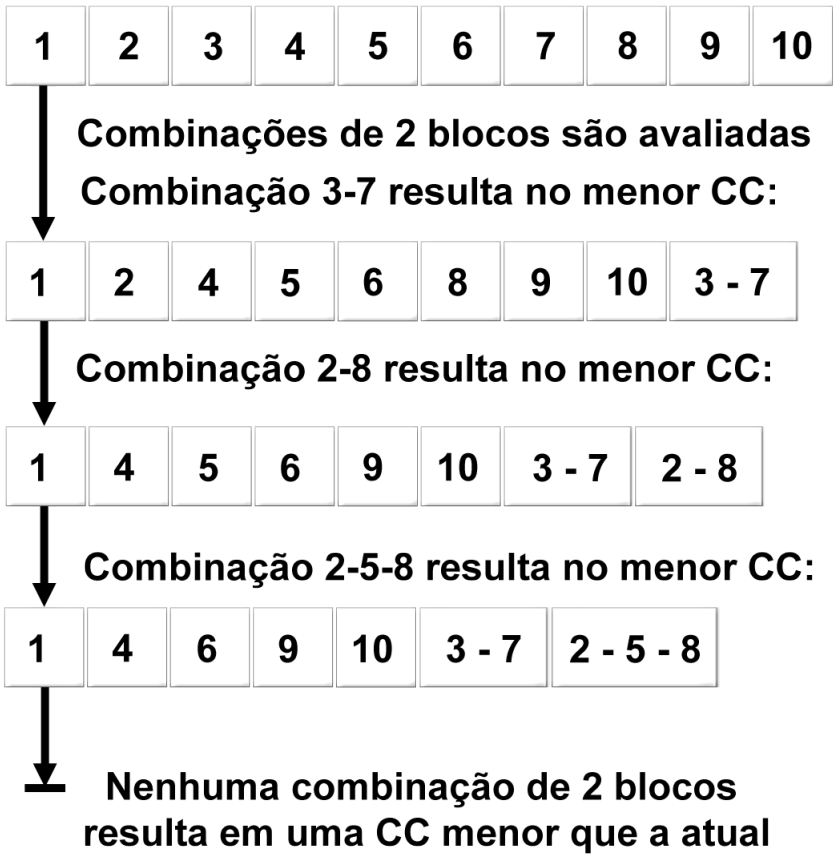


Figura 2.2: Identificação dos building blocks

mente é calculada a CC atual do modelo probabilístico. Então é calculada a CC resultante de todas as possibilidades de agrupamento de dois blocos. Caso haja alguma combinação de dois blocos que resulte em uma CC menor para o modelo, aqueles blocos são agrupados. O procedimento é repetido até que não haja combinações que resultem em uma CC menor que a atual. A Figura 2.2, na qual os números dentro de cada bloco representam a posição dos genes no cromossomo, ilustra esse procedimento.

2.2 Complexidade Combinada

A complexidade combinada, métrica adotada para avaliar a qualidade dos agrupamentos, é dada pela seguinte fórmula:

$$CC = Cm + Cd$$

onde,
Cm: Complexidade do modelo probabilístico.
Cd: Compressão dos dados.

A complexidade do modelo probabilístico (Cm) é dada por:

$$Cm = \sum_{i=1}^{N_{bb}} 2^{l_{bb,i}} * log_2 N$$

onde,
N: Tamanho da população.
N_{bb}: Número de *building blocks*.
$2^{l_{bb,i}}$: Quantidade de genes em cada *building block*.

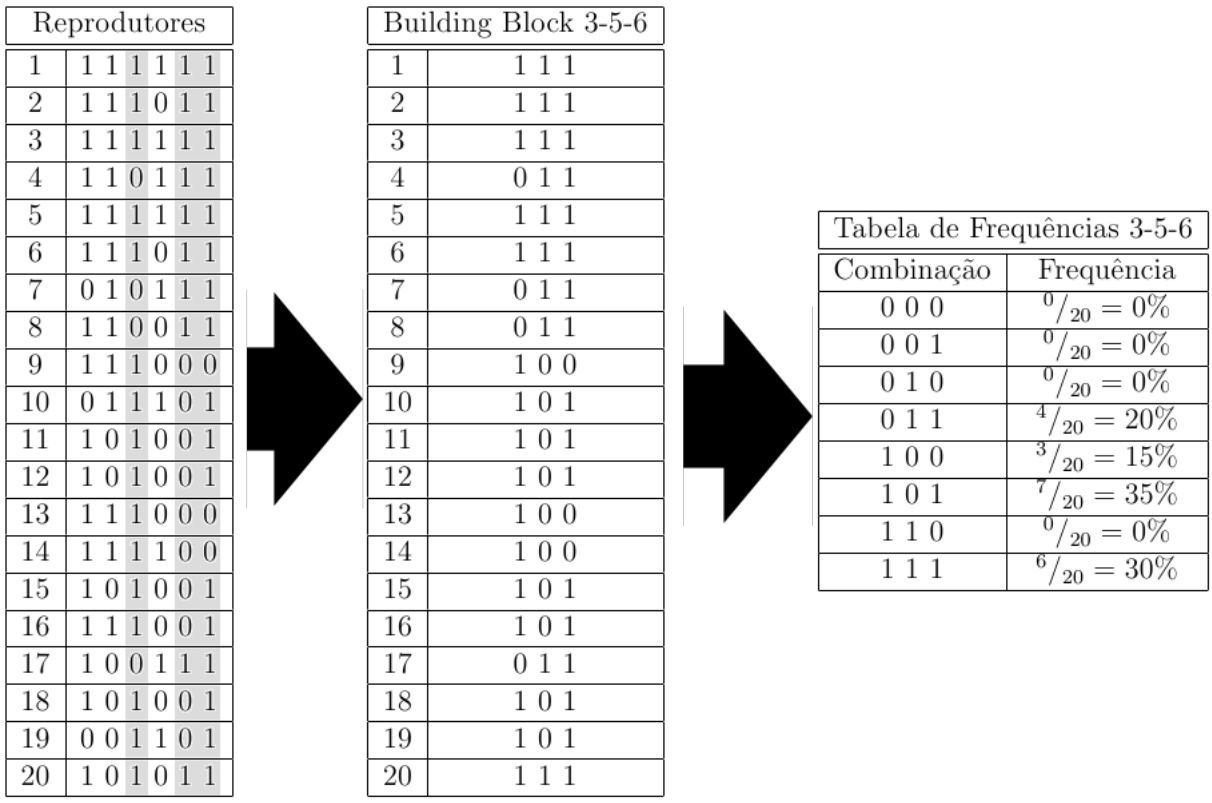
Já a compressão dos dados (Cd), cuja minimização resulta em um modelo mais preciso, com menor entropia, é dada por:

$$Cd = N * \sum_{i=1}^{2^{l_{bb,i}}} P(i) * log_2 P(i)$$

onde,
N: Tamanho da população.
$2^{l_{bb,i}}$: Quantidade de genes em cada *building block*.
$P(i)$: Frequência do gene na população.

2.3 Tabela de frequências

Uma vez encontrados os *building blocks*, é criada uma tabela de frequências das possíveis combinações de genes de cada bloco. Para o preenchimento da tabela são analisados os building blocks dos indivíduos selecionados para a reprodução e a frequência de cada combinação de genes possível naquele



Reprodutores	
1	1 1 1 1 1 1
2	1 1 1 0 1 1
3	1 1 1 1 1 1
4	1 1 0 1 1 1
5	1 1 1 1 1 1
6	1 1 1 0 1 1
7	0 1 0 1 1 1
8	1 1 0 0 1 1
9	1 1 1 0 0 0
10	0 1 1 1 0 1
11	1 0 1 0 0 1
12	1 0 1 0 0 1
13	1 1 1 0 0 0
14	1 1 1 1 0 0
15	1 0 1 0 0 1
16	1 1 1 0 0 1
17	1 0 0 1 1 1
18	1 0 1 0 0 1
19	0 0 1 1 0 1
20	1 0 1 0 1 1

Building Block 3-5-6	
1	1 1 1
2	1 1 1
3	1 1 1
4	0 1 1
5	1 1 1
6	1 1 1
7	0 1 1
8	0 1 1
9	1 0 0
10	1 0 1
11	1 0 1
12	1 0 1
13	1 0 0
14	1 0 0
15	1 0 1
16	1 0 1
17	0 1 1
18	1 0 1
19	1 0 1
20	1 1 1

Tabela de Frequências 3-5-6	
Combinação	Frequência
0 0 0	$^{0}/_{20} = 0\%$
0 0 1	$^{0}/_{20} = 0\%$
0 1 0	$^{0}/_{20} = 0\%$
0 1 1	$^{4}/_{20} = 20\%$
1 0 0	$^{3}/_{20} = 15\%$
1 0 1	$^{7}/_{20} = 35\%$
1 1 0	$^{0}/_{20} = 0\%$
1 1 1	$^{6}/_{20} = 30\%$

Figura 2.3: Preenchimento da Tabela de frequências

bloco é armazenada. A Figura 2.3 ilustra o preenchimento da tabela de frequências para um *building block*. É preenchida uma tabela deste tipo para cada um dos *building blocks* obtidos.

2.4 Geração da Nova População

Cada indivíduo da nova população é gerado utilizando as tabelas de frequência dos *building blocks* obtidas anteriormente. Ao invés do indivíduo ser gerado gene a gene, como no Algoritmo Genético ele é gerado bloco a bloco, utilizando as frequências para determinar a probabilidade da respectiva combinação de genes aparecer em seu cromossomo, de maneira similar à geração de novos indivíduos no *cGA*. Deve-se ressaltar, no entanto, que enquanto no *cGA* apenas dois indivíduos são gerados, no ECGA é gerada uma população

inteira. Uma vez gerada a nova população, são checados os critérios de parada e, caso nenhum tenha sido atingido, todo o procedimento é repetido.

2.5 Critérios de Parada

Os critérios de parada do ECGA são os mesmos de um algoritmo genético comum. Dentre os critérios de parada que podem ser adotados estão a presença de um indivíduo com valor de *fitness* ótimo na população, módulo da diferença entre o *fitness* médio e máximo de uma população menor do que um valor pré-estabelecido, número máximo de iterações atingidos.

Capítulo 3

Algoritmo de Otimização Bayesiana (BOA)

O Algoritmo de Otimização Bayesiana (BOA, do inglês "*Bayesian Optimization Algorithm*", cujo fluxograma esta ilustrado na Figura 3.1, utiliza redes bayesianas como modelo probabilístico para modelar as correlações entre variáveis. Essa estrutura tem como vantagem a possibilidade de se encontrar interseções entre os grupos de variáveis.

O BOA gera uma população inicial e classifica seus indivíduos quanto ao valor de *fitness*. A metade da população com os melhores valores de *fitness* são selecionados como reprodutores. A partir dos reprodutores, é criada a rede bayesiana que modela as correlações entre os genes e são preenchidas as Tabelas de probabilidades condicionadas, que por fim são utilizadas para a criação da nova geração. O procedimento é repetido até que um critério de parada seja atingido.

3.1 Criação da Rede Bayesiana

Baseadas em grafos acíclicos, as redes bayesianas representam variáveis como nós e relações entre as variáveis como arestas. Para a implementação do BOA, são utilizados grafos direcionados na modelagem das redes. Depois de construídas, essas junto com as tabelas de frequência modelarão a probabilidade condicionada das variáveis correlacionadas. A Figura 3.2 ilustra como uma rede bayesiana direcionada pode ser utilizada para modelar a relação entre variáveis.

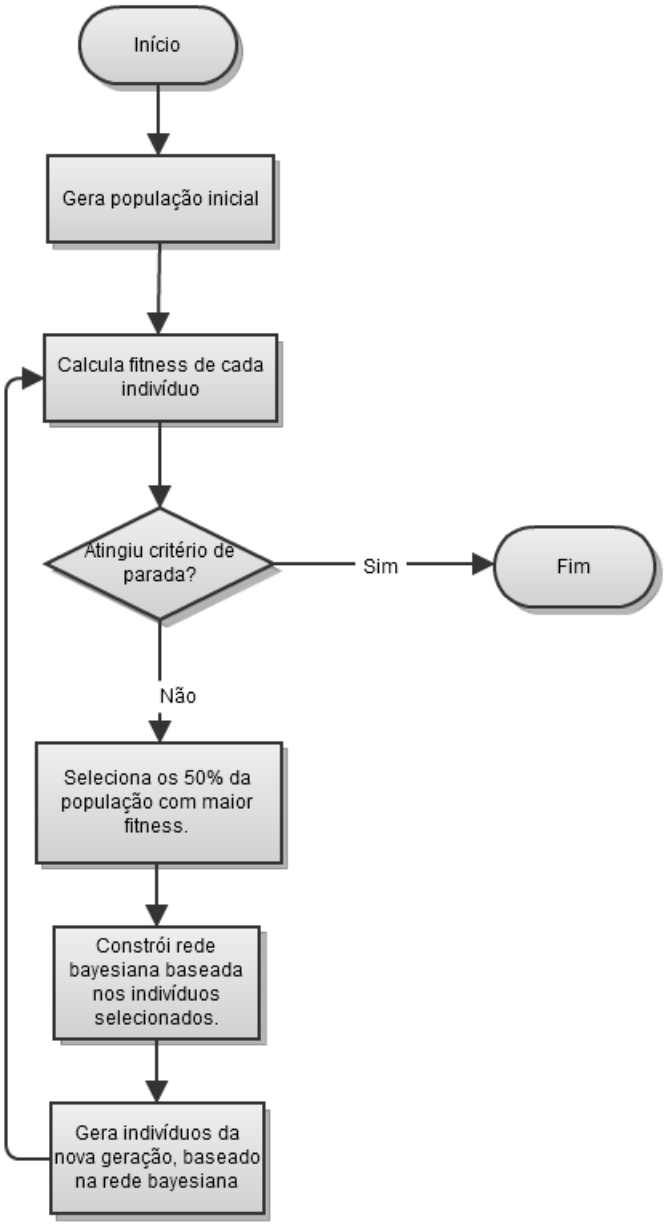


Figura 3.1: Fluxograma do BOA

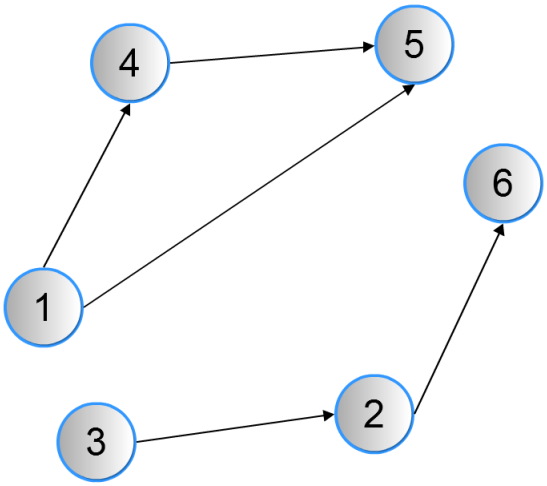


Figura 3.2: Exemplo de rede bayesiana modelando variáveis relacionadas em cromossomos de 6 genes

Para obter a rede Bayesiana, o BOA utiliza um algoritmo capaz de criar a rede a partir do conjunto de indivíduos selecionados. Um algoritmo capaz de realizar tal tarefa é o algoritmo K2 [1]. Este algoritmo admite inicialmente uma rede sem arestas e, então, utiliza uma métrica (que indica quão bem a rede bayesiana atual representa as correlações entre os genes) em conjunto com uma estratégia gulosa, para adicionar as arestas entre os nós que possuem relações. Para cada nó da rede, é buscado o conjunto de nós pais que ocasiona no melhor valor da métrica calculada. A saída do algoritmo é, para cada um dos nós da rede, um conjunto de nós pais.

A métrica utilizada pelo K2 é segue a seguinte fórmula:

$$P(B_s, D) = P(B_s) \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}!$$

onde,
$P(B_s)$: Fator utilizado para se utilizar conhecimento prévio, caso esse exista. Caso essa informação não exista, o valor adotado é 1.
n: Número de variáveis X_i.
q_i: Número de possíveis combinações para os valores dos pais de X_i.
r_i: Número de possíveis valores associados à variável X_i.
N_{ijk}: Número de casos em D onde cada variável X_i tem o valor v_{ik} e π_i

Entrada: Rede a ser preenchida com n nós, lista de arestas z disponíveis, limite de u pais por nó, conjunto π_i de pais de cada nó e conjunto D de cromossomos reprodutores.

Saída: Rede bayesiana criada a partir da identificação dos pais de cada nó.

```
Início
   Para i ←0 até n faça:
      V_atual ← g(i, π_i); //Variável V_atual recebe o valor atual dado pela métrica
      PodeContinuar ← True;
      Enquanto PodeContinuar = True e |π_i| < u, faça:
         Para todo z faça: //A inserção de cada aresta z disponível será testada
            V_novo ← g(i, π_i ∪ z); // V_novo recebe o valor da métrica com a possível nova aresta.
            Se V_novo > V_atual então:
               V_atual ← V_novo
               π_i ← π_i ∪ z
            Senão
               PodeContinuar ← False
            Fim Se
         Fim Para
      Fim Enquanto
   Fim Para
Fim
```

Tabela 3.1: Pseudo-código do algoritmo K2

(conjunto dos pais de X_i) é instanciado como w_{ij}.

Porém, devido à abordagem gulosa do método, essa métrica não precisa ser calculada por completo a cada passo. É calculado apenas o resultado da adição da nova aresta à rede, que é dado pela fórmula:

$$g(i, \pi_i) = \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}!$$

Como a utilização de fatoriais resulta em números muito grandes, deve ser utilizada a forma logarítimica da fórmula na implementação do método, de modo a evitar erros por *overflow*. A forma logarítima é obtida ao substituir $g(i, \pi_i)$ por $log(g(i, \pi_i))$, o que resulta em:

$$log(g(i, \pi_i)) = ln((r_i - 1)!) - ln((N_{ij} + r_i - 1)!) + \sum_{k=1}^{3} N_{ijk}!$$

3.2 Tabelas de probabilidades

Uma vez criada a rede bayesiana, é gerada uma tabela de probabilidade para cada um dos genes, que baseado na frequência das combinações de genes no grupo de reprodutores, armazena a probabilidade de valor de cada gene condicionada às probabilidades dos genes dos quais ele depende. Primeiramente, são preenchidas as tabelas dos genes que não dependem de nenhum outro (a natureza acíclica e direcionada das redes bayesianas garante que haja no mínimo um nó que não dependa de nenhum outro). Nesses casos, apenas a frequência do gene na população é armazenada. A Figura 3.3 ilustra como as tabelas de probabilidades de alguns genes de um grupo hipotético de reprodutores são preenchidas após a geração da rede bayesiana. Note que na figura não esta representada a rede bayesiana inteira, apenas um subgrafo conexo da mesma.

3.3 Geração da nova população

Uma vez preenchidas as tabelas de probabilidades de cada gene, as mesmas são utilizadas para a criação dos indivíduos da nova geração. O processo é similar ao do ECGA, porém, como parte dos genes têm suas probabilidades condicionadas ao valor de outros, a ordem na qual os genes são preenchidos,

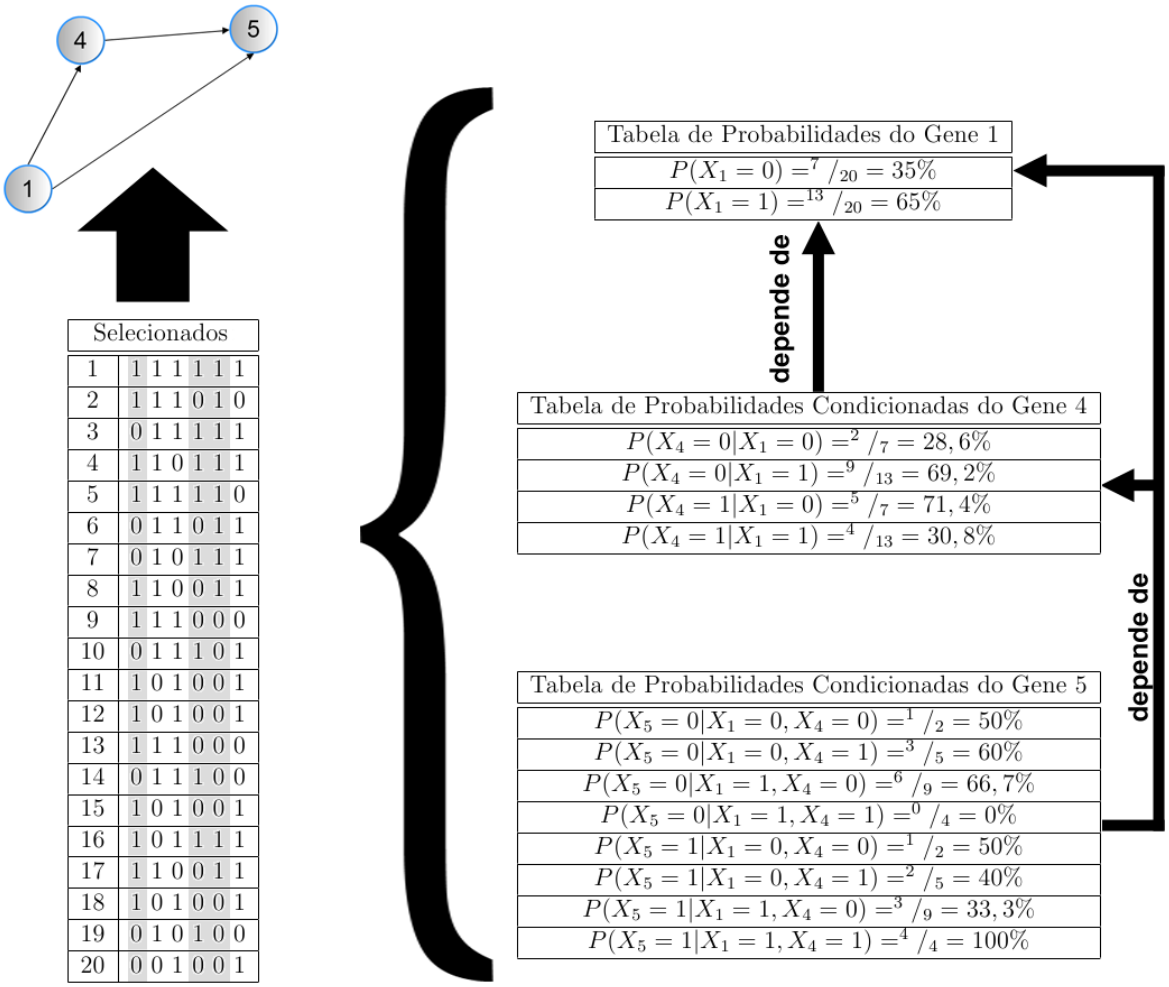


Figura 3.3: Geração de Tabelas de probabilidade a partir de redes bayesianas

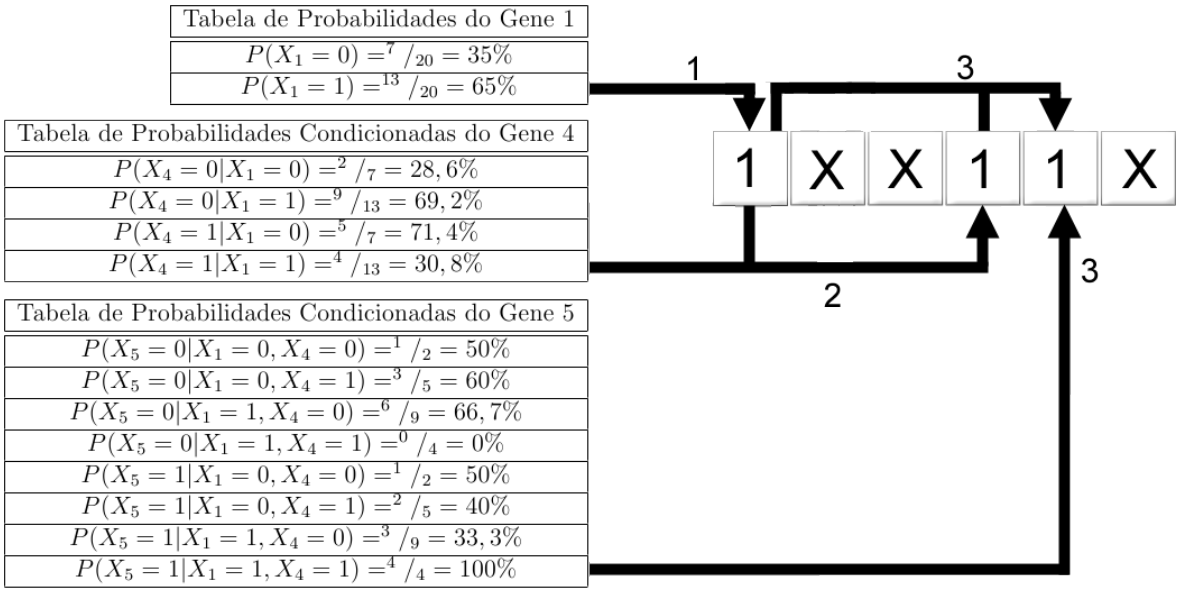


Figura 3.4: Geração de um indivíduo a partir das Tabelas de probabilidades

depende da topografia da rede bayesiana. Inicialmente são preenchidos os genes que não têm nenhuma dependência, seguidos pelos genes que já tiveram suas dependências preenchidas, até que todos os genes tenham sido preenchidos. A Figura 3.4 ilustra a sequencia na qual as tabelas geradas na Figura 3.3 são utilizadas no preenchimento do cromossomo de um novo indivíduo.

3.4 Critérios de Parada

O BOA segue a mesma linha de filosofia do Algoritmo Genético e do ECGA: Criação, avaliação por uma função de *fitness* e evolução de uma população. Portanto, os critérios de parada desses algoritmos também se aplicam ao BOA.

Capítulo 4

Algoritmo Filogenético (ΦGA)

Diferente dos algoritmos vistos anteriormente, o Algoritmo Filogenético (ΦGA, do inglês *Phylogenetic Algorithm*) [9] não é iterativo, como é ilustrado pelo fluxograma da Figura 4.1. Seu modelo probabilístico é baseado em estruturas chamadas "árvores filogenéticas" [4], que em biologia evolutiva são utilizadas para descrever as ramificações evolutivas das espécies, como ilustrado na Figura 4.2[1].

No ΦGA, são executadas a geração de população inicial e seleção de reprodutores, da mesma maneira que no AG. Uma vez selecionados, é executado um método que mede a distância entre as variáveis utilizando uma métrica chamada "*Mutual Information*", e cria uma tabela com essas distâncias. Essa tabela de distâncias é passada como entrada para um método chamado "*Neighbor Joining*", que cria uma árvore filogenética em forma de estrela, cujos ramos são os *building blocks*. Ao final da execução do *Neighbor Joining*, todas as combinações de *building blocks* e todos os máximos globais da função são encontrados. Para se obter o valor ótimo da mesma é utilizada uma busca gulosa nos *building blocks* encontrados.

4.1 Criação de matrizes de distâncias

A distância entre duas variáveis no ΦGA é dada pela fórmula:

$$d(X, Y) = H(X, Y) - I(X, Y)$$

[1]Adaptada de http://professornandao.blogspot.com/2008/05/rvores-filogeneticas.html (Acessado em 08/11/2010)

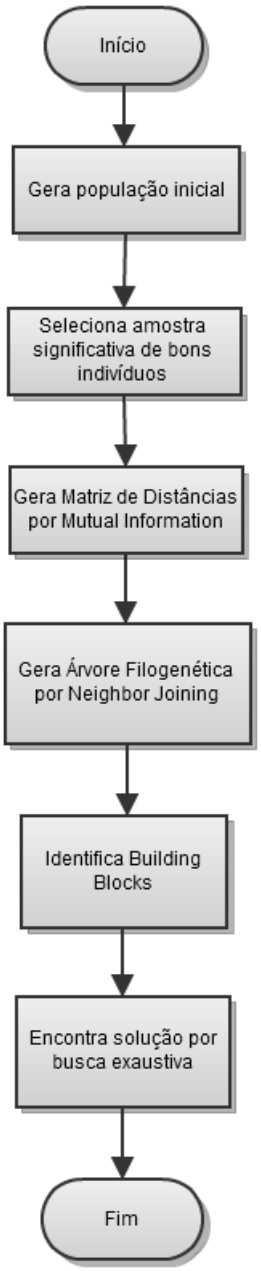


Figura 4.1: Fluxograma do ΦGA.

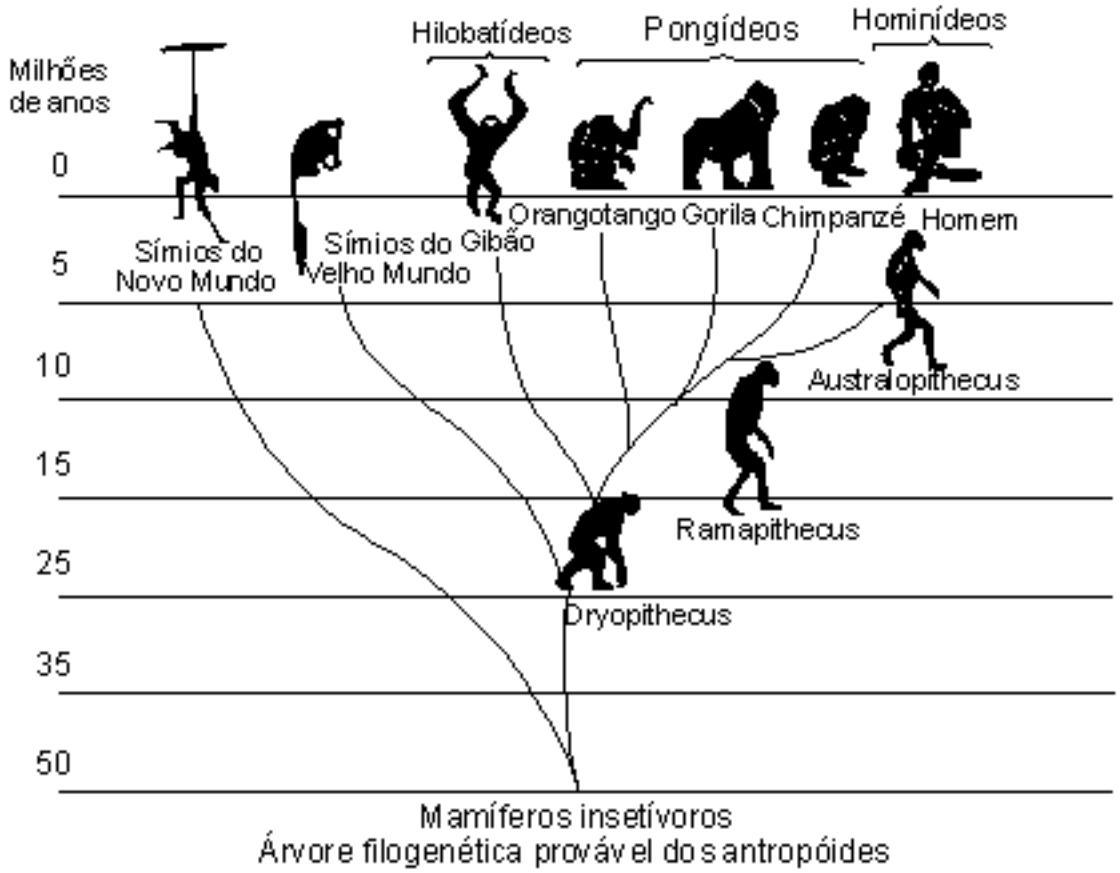


Figura 4.2: Exemplo de aplicação de árvore filogenética em biologia evolutiva.

Y/X	X_1	X_2	$\cdots$	X_n
Y_1	$d(X_1, Y_1)$	$d(X_2, Y_1)$	$\cdots$	$d(X_n, Y_1)$
Y_2	$d(X_1, Y_2)$	$d(X_2, Y_2)$	$\cdots$	$d(X_n, Y_n)$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
Y_n	$d(X_1, Y_n)$	$d(X_2, Y_n)$	$\cdots$	$d(X_n, Y_n)$

Tabela 4.1: Indivíduos da segunda geração e seus respectivos *fitness*

onde $I(X, Y)$ é a métrica *Mutual Information*, dada por:

$$I(X, Y) = \sum_{i \in X} \sum_{j \in Y} p_{ij} * log \left(\frac{p_{ij}}{p_i * p_j} \right)$$

onde, X, Y: Variáveis aleatórias.
p_{ij}: Probabilidade conjunta das variáveis X e Y assumirem respectivamente os valores i e j.
p_i e p_j: Probabilidades Marginais das variáveis X e Y. Quando p_i ou p_j forem iguais a zero, assume-se que a função $I(X, Y)$ também é igual a zero, para evitar a divisão por zero.

e $H(X, Y)$ é o valor de Entropia [2], dada por:

$$H(X, Y) = -\sum_{x \in X} \sum_{y \in Y} P(x, y) log_2[P(x, y)]$$

Os valores de $I(X, Y)$ obtidos com essa métrica são as distâncias entre duas variáveis para aquele grupo de indivíduos selecionados. Esses valores são utilizados para preencher uma matriz de distâncias como a ilustrada na Tabela 4.1.

4.2 Neighbor Joining

Utilizando a matriz de distâncias D_n criada no passo anterior, o *Neighbor Joining* cria uma árvore filogenética. Inicialmente, a árvore é inicializada com um vértice artificial chamado "-1"como raiz, com todos os outros vértices ligados a ele, como é ilustrado na Figura 4.3.

Então é calculado o valor da divergência da rede para cada variável $i \in N = \{1, 2, ...n\}$, dada por:

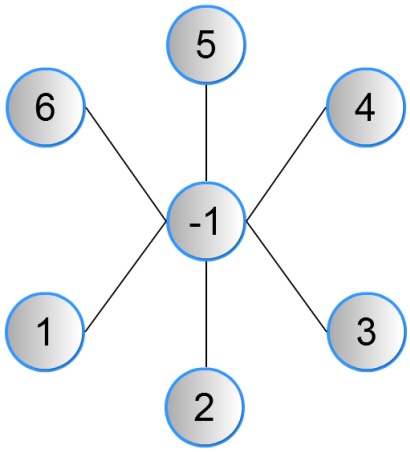


Figura 4.3: Estado inicial de uma árvore-estrela criada a partir da matriz de distâncias.

$$R_i = \sum_{j \neq 1} D_{ij}$$

e é gerada uma nova matriz $M_{i,j}$ onde:

$$M_{ij} = D_{ij} - \frac{R_i + Rj}{n-2}$$

Nessa nova matriz, é identificado o menor valor de M_{ij} e as variáveis correspondentes a esse valor são retiradas da matriz D_n original e agrupadas sob um novo nó u, cuja distância entre u e as demais variáveis é dada por:

$$D_{u,i} = \frac{D_{a,i} + D_{b,i} - D_{a,b}}{2}$$

onde,
a,b: nós retirados da árvore e agrupados sob o novo nó u.

e a distância S dos nós a e b ao nó u são dadas por:

$$S_{a,u} = \frac{D_{a,b}}{2} + \frac{R_a - R_b}{2(n-2)}$$

$$S_{b,u} = \frac{D_{a,b}}{2} + \frac{R_b - R_a}{2(n-2)}$$

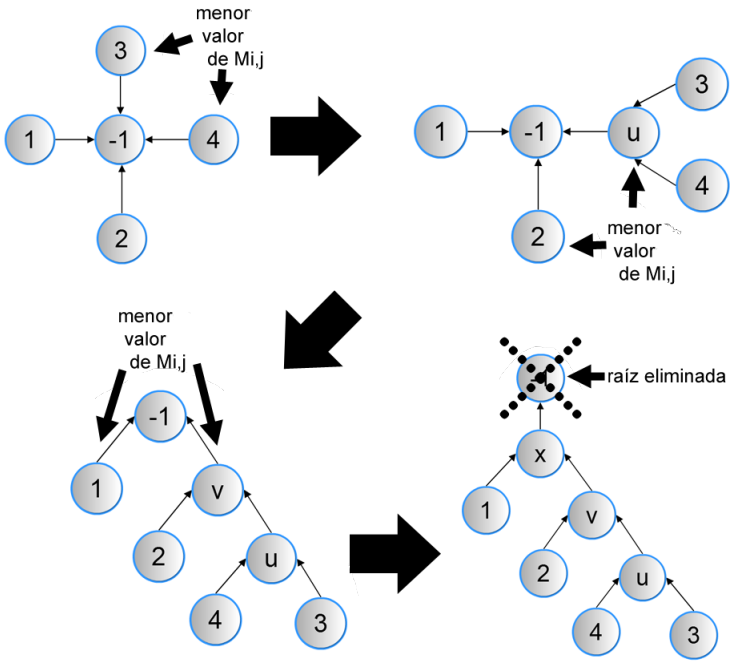


Figura 4.4: Criação de árvore filogenética por *Neighbor Joining*.

A Figura 4.4 ilustra os passos do método *Neighbor Joining* a partir uma árvore-estrela criada a partir de um cromossomo formado por 4 genes. Após cada a iteração desse método, a dimensão da matriz D é diminuída de uma unidade. Esse procedimento é repetido até que a matriz D tenha dimensão 1. Nesse momento, a raiz "-1"é retirada e a árvore filogenética está pronta.

4.3 Poda da árvore

Uma vez gerada a árvore filogenética, é necessário fazer uma poda para se identificar os *building blocks*. Dentre os vários métodos existentes para se decidir onde será feita a poda, uma opção é o *Average Thresholding* [9], onde simplesmente é feita a média dos valores das arestas da árvore e são removidas as arestas cujos valores sejam maiores que a média. Cada uma das sub-árvores resultantes da poda é um *building block*. A Figura 4.5 ilustra a poda de uma pequena árvore filogenética utilizando o método *Average Tresholding*.

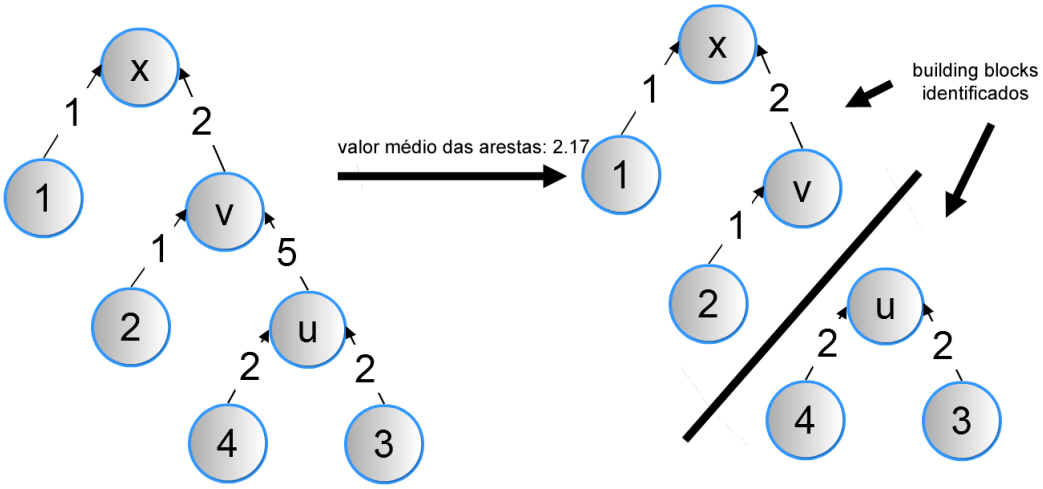


Figura 4.5: Poda de árvore filogenética por *Average Thresholding*.

4.4 Busca da solução

Uma vez identificados os *building blocks* várias abordagens podem ser utilizadas para se buscar o valor ótimo da função, como por exemplo uma busca gulosa. Nessa busca, é selecionado o cromossomo de maior valor de *fitness* da população. Então, para cada um dos *building blocks* identificados, são testadas todas as combinações de genes possíveis. O *building block* manterá a combinação de genes que obtiver o maior valor de *fitness*. Os genes do cromossomo após essa busca ser efetuada em todos os *building blocks* constituem o valor ótimo da função.

Referências Bibliográficas

[1] Gregory F. Cooper and Edward Herskovits. A bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347, October 1992.

[2] T. Cover and J. Thomas. *Elements of Isnformation Theory*. Wiley, 1991.

[3] Charles Darwin. *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life*. John Murray, 1859.

[4] J. Felsenstein. *Inferring phylogenies*. Sinauer Associates, Connecticut, 2003.

[5] David E. Goldberg. *The Design of Innovation: Lessons from and for Competent Genetic Algorithms*. Kluwer Academic Publishers, Norwell, 2002.

[6] Georges R. Harik, Fernando G. Lobo, and David E. Goldberg. The compact genetic algorithm. *IEEE Trans. Evolutionary Computation*, 3(4):287–297, 1999.

[7] Martin Pelikan, David E. Goldberg, and Erick Cantu-Paz. Boa: The bayesian optimization algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-99)*, pages 525–532. Morgan Kaufmann, 1999.

[8] Kumara Sastry and David E. Goldberg. On extended compact genetic algorithm. Technical Report 2000026, Illinois Genetic Algorithms Laboratory (IlliGAL), University of Illinois, Urbana/IL, 2000.

[9] Danilo Vasconcellos Vargas and Alexandre Cláudio Botazzo Delbem. Algoritmo filo-genético. Technical Report 350, Instituto de Ciêncais Matemáticas e de Computação, Universidade de São Paulo, São Carlos/SP, Mar 2010.