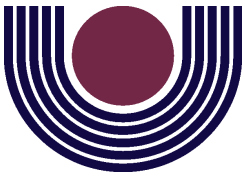


Unioeste - Universidade Estadual do Oeste do Paraná
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
Colegiado de Informática
Curso de Bacharelado em Informática

A Aplicação de um Algoritmo Genético no Processo de Clusterização

Carlos Eduardo Rodrigues Diógenes

CASCAVEL
2008

CARLOS EDUARDO RODRIGUES DIÓGENES

A APLICAÇÃO DE UM ALGORITMO GENÉTICO NO PROCESSO DE CLUSTERIZAÇÃO

Monografia apresentada como requisito parcial para obtenção do grau de Bacharel em Informática, do Centro de Ciências Exatas e Tecnológicas da Universidade Estadual do Oeste do Paraná - Campus de Cascavel

Orientador: Prof. Adair Santa Catarina

CASCAVEL
2008

CARLOS EDUARDO RODRIGUES DIÓGENES

A APLICAÇÃO DE UM ALGORITMO GENÉTICO NO PROCESSO DE CLUSTERIZAÇÃO

Monografia apresentada como requisito parcial para obtenção do Título de Bacharel em Informática, pela Universidade Estadual do Oeste do Paraná, Campus de Cascavel, aprovada pela Comissão formada pelos professores:

_____

Prof. Adair Santa Catarina (Orientador)
Colegiado de Informática, UNIOESTE

_____

Prof. Jorge Bidarra
Colegiado de Informática, UNIOESTE

_____

Prof. Josué Pereira de Castro
Colegiado de Informática, UNIOESTE

Cascavel, 8 de dezembro de 2008

“Sarcasmo é o refúgio dos fracos” – Jean Paul Sartre

AGRADECIMENTOS

Agradeço principalmente a minha esposa, que além do todo mais, foi a maior amiga que tive durante estes anos de faculdade, com toda certeza ela fez esses anos serem mais felizes, mais sensatos e muito mais doces. Agradeço as minhas filhas por ficarem me perguntando porque eu passo tanto tempo na frente do computador e me arrancarem da frente dele. Agradeço aos meus pais pela disciplina, carinho e ética. Agradeço aos meus amigos, por serem meus amigos mesmo sabendo quem eu sou, seria bastante difícil esses anos sem vocês. Agradeço ao meu orientador, que além de ser mestre, também virou um amigo.

Lista de Figuras

Lista de Abreviaturas e Siglas

AD	Algoritmo Determinístico
AG	Algoritmo Genético
BT	Busca Tabu
SA	Simulated Annealing

Sumário

Resumo

Neste trabalho aplicou-se um Algoritmo Genético (AG) na resolução de problemas de clusterização com o objetivo principal de verificar sua eficácia. A principal motivação para este estudo se deve ao fato de que problemas de clusterização possuem um grande espaço de busca e os AGs são uma boa ferramenta para explorá-los. Realizou-se então experimentos utilizando dois casos de uso. O primeiro utiliza imagens digitais e o segundo utiliza a clássica base de dados Íris. Os resultados obtidos pelo AG foram comparados com o tradicional algoritmo de clusterização K-Médias. Por fim, percebeu-se que a codificação adotada no AG permitia resolver o problema de forma determinística. Mesmo assim, constatou-se que o AG possui vantagens e desvantagens em relação ao K-Médias; outras formas de codificação permitiriam ao AG realizar boas clusterizações e de forma mais robusta. Uma das vantagens do AG é fornecer a quantidade de clusters ao final do processo. Tais conclusões sugerem que outros estudos devem ser conduzidos de forma a melhor verificar a eficácia dos AGs na resolução de tais problemas, principalmente devido a grande quantidade de recursos exigidos pelo método em termos de processamento e memória, quando aplicado a grandes base de dados.

Palavras-chave: clusterização, algoritmo genético, k-médias.

Capítulo 1

Introdução

Atualmente, quantidades significativas de dados estão disponíveis em meio eletrônico. Muitas empresas precisam gerenciar um volume crescente de informações armazenadas em bancos de dados, o que tem tornado praticamente inviável a sua interpretação por seres humanos, sem o auxílio de ferramentas automatizadas. Não obstante os dados nessas bases, a quantidade de informação disponível na web é cada vez maior, como pode ser observado pelo crescimento de servidores na figura 1.1. Métodos para resumir e extrair informações desses ambientes de dados são úteis para facilitar sua compreensão.

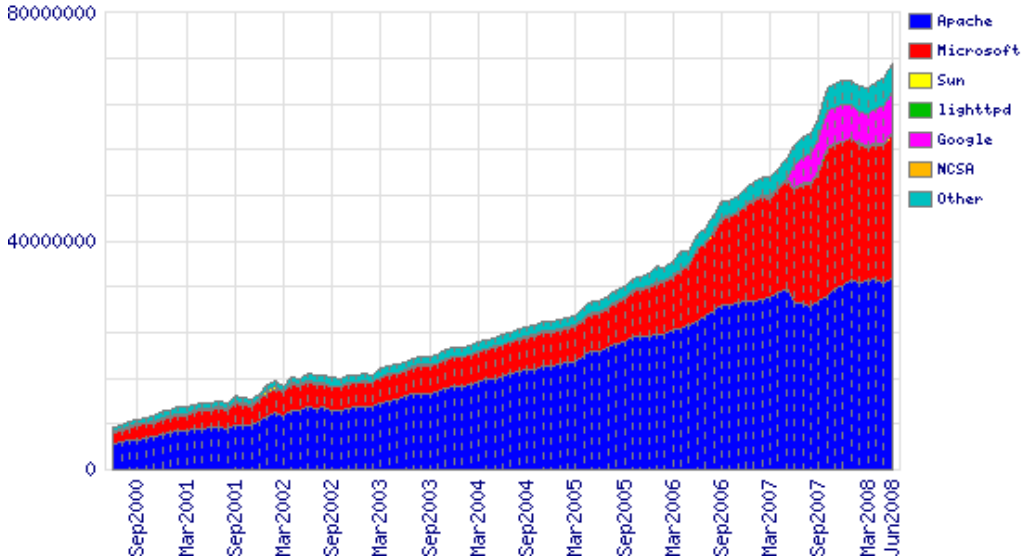


Figura 1.1: Número de servidores ativos.[21]

Uma das formas de realizar este resumo dos dados é através de métodos de clusterização. Nestes, um conjunto de dados é explorado em busca de (dis)similaridades para serem agrupa-

dos, sendo que os objetos classificados como pertencentes a um determinado grupo terão mais semelhança entre si do que em relação a objetos em outros grupos[2]. Tal operação é fundamental na tarefa de mineração de dados[19], além de já ter sido utilizada em diversas disciplinas, tais como psiquiatria, pesquisa de mercado, arqueologia, reconhecimento de padrões, engenharia e medicina[2].

Neste trabalho aplicaremos um algoritmo genético (AG) para realizar a tarefa de clusterização, comparando seus resultados ao tradicional algoritmo de clusterização K-Médias. Para isso, utilizaremos dois casos de uso. Estes são apresentados e discutidos no capítulo 4. Antes disso, nos capítulos 2 e 3, apresentamos um breve levantamento bibliográfico dos algoritmos genéticos e dos métodos de clusterização respectivamente. Por fim, no capítulo 5 apresentamos as considerações finais do trabalho e possíveis trabalhos futuros.

1.1 Justificativa

Acima já mencionamos a importância de métodos para resumir e extrair informações, bem como uma breve caracterização dos métodos de clusterização. Sendo assim, o estudo de algoritmos genéticos para realizar esta tarefa é interessante, pois como será visto no capítulo 2, tais algoritmos são eficientes para explorar grandes espaços de busca. No caso dos problemas de clusterização este espaço de busca é combinatório exponencial. O número de maneiras de se ordenar *n* objetos em *k* clusters é dado por Liu[17] *apud* [2]:

$$N(n,k) = \frac{1}{k!}\sum_{i=0}^{k}(-1)^i \left(\begin{array}{c} k \\ i \end{array} \right) (k-i)^n \tag{1.1}$$

Por exemplo, existem N (25, 5) = 2.436.684.974.110.751 formas de ordenar 25 objetos em 5 clusters. Caso o número de clusters seja desconhecido os objetos podem ser classificados em $\sum_{i=1}^{n} N(n,k)$ maneiras. Para o caso de 25 objetos isto é mais que 4×10^{18} clusters, mostrando claramente que aplicar um algoritmo de força bruta para procurar pela solução ótima é impraticável.

Cole[2] aponta alguns pontos negativos apresentados pelos algoritmos de clusterização tradicionais:

> *... algoritmos de clusterização tradicionais buscam um conjunto muito pequeno do espaço de solução. Conseqüentemente, a probabilidade de sucesso destes métodos é pequena. Algoritmos como o ligação individual são determinísticos e irão encontrar sempre a mesma solução para um dado conjunto de dados, enquanto que algoritmos tais como o K-Médias conduzem uma busca local começando por uma partição inicial. Em cada caso, a solução deve ser um ótimo 'local', o qual não é necessariamente a solução 'global'. Isto é exacerbado quando o espaço da solução é muito grande.*

Uma das condições necessárias para que o K-Médias encontre a solução ótima é conhecer, a priori, o número de clusters existentes. O AG implementado neste trabalho, desconhece esta informação; o mecanismo evolutivo, inerente ao AG, acaba por descobri-la.

O principal objetivo deste trabalho é aplicar um AG para solucionar o problema de clusterização. Além disso compararemos os resultados obtidos com o K-Médias, bem como o tempo de execução de ambos os algoritmos. Outro objetivo é disponibilizar o código-fonte e os dados utilizados nos testes para estudos futuros, facilitando a utilização da técnica em conjuntos de dados diferentes e/ou com propósitos diferentes.

1.2 Trabalhos Correlatos

A resolução de problemas de clusterização através de AGs não é uma novidade. Segundo Al-Sutan e Khan[1] *apud* [19], os algoritmos *Simulated Annelaing* (SA), Busca Tabu (BT) e AG, aplicados em problemas de clusterização, foram julgados iguais em termos de qualidade da solução; entretanto, tais soluções foram melhores que as encontradas pelo K-Médias.

O K-Médias é mais eficiente em termos de tempo de execução, sendo de 500 a 2.500 vezes mais rápido para particionar um conjunto de dados de 60 elementos em 5 clusters.

Dentre os algoritmos SA, BT e AG, o que levou menos tempo para encontrar a melhor solução foi o AG, seguido de BT e SA respectivamente.

Apesar destas constatações, o único algoritmo aplicado a grandes base de dados foi o K-Médias, pois o tempo de execução dos demais métodos é muito grande para este tipo de caso. No entanto, foi mostrado que o K-Médias converge para um ótimo local. Isto se deve a seleção inicial dos clusters na inicialização do algoritmo. Em todo caso, se uma boa partição inicial

puder ser obtida, então o K-Médias deverá funcionar bem, mesmo em problemas com grandes base de dados. Além disso, através deste estudo, foi constatado que adicionar conhecimento do domínio aos algoritmos pode melhorar sua performance.

Cole[2] também apresenta um estudo de clusterização utilizando AG. Ele chega a conclusão de que, apesar do AG realizar corretamente a tarefa, não houve vantagem em relação aos demais métodos. Os fatores limitantes da performance de um AG encontrado nesta pesquisa foram dois: o primeiro é a função objetivo, pois se esta não representar bem a estrutura dos dados, o AG não será capaz de encontrar bons clusters; o segundo é a alta complexidade do AG em si quando comparada aos métodos tradicionais, pois além de carregar a estrutura necessária para a evolução da solução este usa em sua codificação critérios de clusterização presentes nos algoritmos tradicionais, o que acaba por limitar a qualidade da solução que este pode encontrar. Além disso, ele chega a conclusão de que explorar uma parte mais ampla do espaço de busca não é vantajoso, pois a função objetivo empregada não é suficiente para encontrar melhores soluções.

Capítulo 2

Algoritmos Genéticos

Nas décadas de 50 e 60 vários cientistas dedicaram sua atenção ao estudo de sistemas evolutivos, os quais evoluíam soluções para determinados problemas, principalmente de engenharia, utilizando operadores inspirados pela variação genética natural e seleção natural[20].

Na década de 60, John Holland inventou o conceito de AGs e durante esta década e a seguinte ele, seus estudantes e colegas desenvolveram a técnica. Um fato interessante da criação dos algoritmos genéticos é que ao contrário de outras estratégias evolutivas, o seu objetivo inicial não era resolver problemas específicos, mas ser um método formal pra estudar o fenômeno da adaptação como este ocorre na natureza, bem como importar suas operações para sistemas computacionais. No livro de 1975, *Adaptation in Natural and Artificial Systems*, John Holland apresentou a estrutura básica de um AG, mostrando como uma população de "cromossomos", codificados em *strings* de zeros e uns, evoluía utilizando um mecanismo de "seleção natural" e operadores de cruzamento, mutação e inversão inspirados na genética. Esta estrutura reflete muito bem os interesses do grupo ao desenvolver a técnica.

Atualmente, devido a interação entre os pesquisadores da área de computação evolutiva[20], o *framework* inicial apresentado por John Holland muitas vezes é modificado ou combinado com técnicas tradicionais de otimização. As modificações podem ser efetuadas tanto na codificação dos cromossomos quanto no conjunto de operadores utilizados. Como exemplo de técnicas de otimização que podem ser combinadas com AG temos o *simulated annealing*, *hill climbing* e *ant colony* [23][18][12][16]. Esta mudança, faz com que o termo "algoritmo genético" seja utilizado por muitos num contexto que descreve algo completamente diferente do apresentando inicialmente por Holland, porém muitos autores mantêm o termo pois as técnicas possuem grande semelhança[20].

O que tornou os AGs populares foi a inovação na forma como estes trabalhavam, ou seja, baseado em uma população e utilizando os operadores de cruzamento, mutação e inversão. Até aquele momento as estratégias evolutivas utilizavam apenas o operador de mutação.

Outro aspecto importante na popularização dos AGs foi o trabalho de Holland para fundamentá-los em uma base teórica sólida. Ele definiu a noção de *schemas*, que é uma padrão de similaridade que descreve um sub-conjunto de *strings* com similaridade em determinadas posições[20]. No caso do alfabeto utilizado para *strings* binárias, {0, 1}, é adicionado o símbolo *, também chamado de *tanto faz*, para representar um *schema*. Como exemplo, o *schema* *0000 corresponde a duas strings, sendo elas {10000, 00000}· Outro exemplo seria o *schema* *111* que descreve um sub-conjunto com quatro membros {01110, 01111, 11110, 11111}. Os símbolos fixos no *schema* são chamados de "blocos de construção".

Esta noção foi importante, pois de acordo com a teoria tradicional apresentada por ele em 1975, um AG funciona pois descobre, enfatiza, e recombina blocos de construção das soluções em uma forma altamente paralelizada. Além disso, ele definiu formas de calcular a probabilidade de um determinado "schema" sobreviver entre gerações. Com isso era possível analisar o comportamento de alguns aspectos da evolução de um AG, no entanto trabalhos ainda precisam ser feitos para provar alguns aspectos de sua teoria, bem como novas teorias devem ser criadas para fundamentar a teoria dos AGs[20].

2.1 Espaços de Busca

A busca por uma solução específica em um conjunto de possíveis soluções é uma operação bastante comum na computação, tanto que tal operação recebe o nome de "procura em um espaço de busca"[20]. Apesar de terem sido criados para estudar evolução como acontece na natureza, os AGs mostraram-se uma ferramenta eficaz para este tipo de operação, além de possuir a robustez como propriedade, isto é, consegue encontrar boas soluções, algumas vezes ótimas, em espaços de busca bastante variados.

O que faz o AG robusto em diferentes condições, é que estes não precisam levar em consideração a necessidade da existência de derivadas, o que é um requisito para os métodos baseado no cálculo. Os métodos baseados em enumeração são muito custosos quando o espaço de busca é muito grande, e este normalmente é. Os métodos de busca aleatória são ineficientes,

pois levam muito tempo para encontrar boas soluções, já que não possuem nenhuma forma de direcionamento. Nos AGs, o método de seleção direciona a busca em relação aos melhores indivíduos, os operados de cruzamento e mutação permitem encontrar indivíduos melhores e explorar amplamente o espaço de busca[8].

Devido a estas características, mais a simplicidade de implementação, os AGs ganharam popularidade entre aplicações de negócios, científicas e de engenharia que envolvem busca[8]. Finalmente, é importante ressaltar que, apesar de sua popularidade, esta é uma ferramenta de propósito geral; deste modo, quando métodos mais apropriados existirem, tais métodos devem ser utilizados, pois muito provavelmente encontrarão soluções mais rapidamente e melhores. GAs são recomendados quando nenhum método funcionou, ou quando o espaço de busca é desconhecido[24].

2.2 Implementação de um Algoritmo Genético Simples

Na literatura é possível encontrar uma grande variedade de opções que podem ser utilizadas para implementar um AG, desde a codificação do cromossomo, a implementação dos operadores genéticos até os parâmetros do AG. O leitor interessado pode consultar [20], [8] e [24] para obter mais informações, pois aqui apresentaremos apenas uma breve descrição dos elementos mais utilizados e que apresentam bons resultados.

2.2.1 Parâmetros de um Algoritmo Genético

Segundo Mitchell[20], existe uma grande discussão sobre como configurar os parâmetros de um AG, pois estes tipicamente interagem de forma não linear, sendo assim eles não podem ser otimizados um de cada vez. Isto torna difícil encontrar parâmetros ótimos, sendo assim, muitos utilizam parâmetros que deram certo anteriormente. Alguns estudos, que focaram na otimização de funções matemáticas de várias variáveis, na tentativa de estimar os melhores parâmetros de um AG são:

1. Parâmetros apresentados por De Jong e Spears[15]
   (a) Tamanho da população: 50
   (b) Número de gerações: 1.000

(c) Tipo do cruzamento: dois pontos

(d) Taxa do cruzamento: 60%

(e) Tipo da mutação: inversão de bit

(f) Taxa de mutação: 0,1%

2. Parâmetros apresentados por Grefenstette[11]

(a) Tamanho da população: 30

(b) Número de gerações: Não especificado

(c) Tipo do cruzamento: dois pontos

(d) Taxa do cruzamento: 90%

(e) Tipo da mutação: inversão de bit

(f) Taxa de mutação: 1%

De acordo com Santa Catarina[23], os parâmetros de um AG influenciam sua execução da seguinte forma:

1. Tamanho da população: afeta o desempenho global e a eficiência do algoritmo. Uma população pequena tende a diminuir a eficiência, pois o espaço de busca tem uma representação menor, já uma população grande representa uma cobertura maior do domínio de busca, além de prevenir convergências prematuras a soluções sub-ótimas.

2. Taxa de cruzamento: determina a probabilidade com a qual o cruzamento ocorrerá. Quanto maior a taxa de cruzamento, maior é a inserção de novas estruturas na população. Se esta for muito alta, muitos indivíduos serão substituídos, o que pode fazer alguns indivíduos de alta aptidão serem perdidos, já um valor baixo pode deixar a convergência muito lenta.

3. Taxa de mutação: determina a probabilidade de uma mutação ocorrer. Caso a taxa de mutação seja muito alta o algoritmo se comporta como uma busca aleatória.

Por fim, mesmo que alguns estudos indiquem parâmetros a serem utilizados, nem sempre estes são adequados a todas as classes de problemas, sendo que ainda existe muita discussão a

respeito de como configurar estes parâmetros, além de métodos para modificar os parâmetros durante a execução de um AG[20]. Isto implica que muitas vezes tais parâmetros devem ser ajustados empiricamente.

2.2.2 Métodos de Reprodução

O método de reprodução, também conhecido como método de seleção, é responsável por selecionar os indivíduos que irão sobreviver; às vezes estes indivíduos sofrerão cruzamento e/ou mutação. Esta é a versão artificial da seleção natural, pois tais métodos privilegiam os indivíduos mais aptos da população, ou seja, aqueles que possuem maior *fitness* na esperança de que os novos indivíduos tenham um *fitness* ainda maior. Segundo Mitchell[20], este método deve ser balanceado com variação das operações de cruzamento e mutação, pois uma seleção muito forte irá fazer com que a população convirja rapidamente, provavelmente para soluções sub-ótimas, já uma seleção muito fraca resultará em uma evolução muito lenta.

Vários métodos foram propostos durante os anos de pesquisa na área, porém o mais comumente utilizado e que foi introduzido pelo trabalho inicial de Holland é o método da roleta, o qual atribui para cada indivíduo uma fatia de uma "roleta giratória", sendo o tamanho da fatia proporcional ao "fitness" do indivíduo. A roleta é então girada N vezes, sendo N o número de elementos da população, e na fatia na qual ela pára condiz com o elemento que é selecionado para reprodução. Uma representação visual deste método é apresentado na figura 2.1.

Um método que pode ser adicionado aos métodos de reprodução é o elitismo, onde uma parcela dos indivíduos mais aptos da população, definida no código ou configurada pelo usuário, é copiada para a nova população sem modificações. Isto previne que bons elementos sejam perdidos por não serem selecionados para cruzamento ou sejam destruídos pelas operações de cruzamento e/ou mutação. Muitos pesquisadores constataram que o elitismo melhorou significantemente a performance do AG[20].

2.2.3 Operadores Genéticos

Os operadores genéticos são responsáveis por modificar os indivíduos, criando novas soluções. Aqui apresentamos os operadores mais utilizados: o cruzamento e a mutação; operando sobre uma *string* binária.

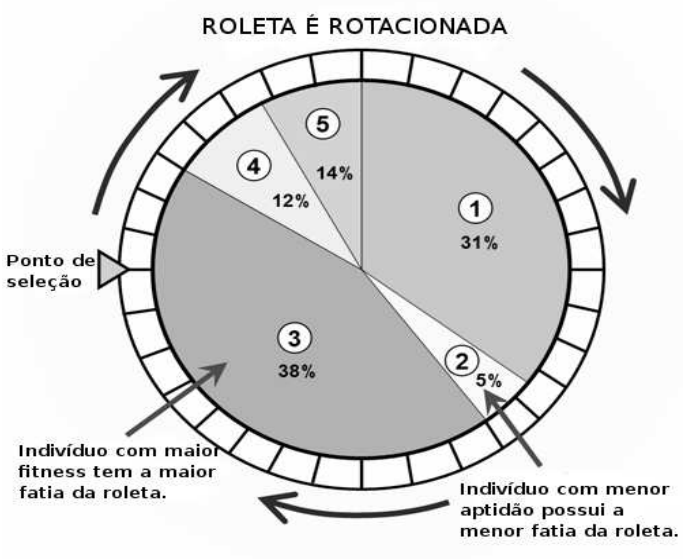


Figura 2.1: Método da roleta.[26]

1. Cruzamento: esta operação é a que distingue os AGs de outros métodos da computação evolutiva, sendo a operação de maior importância para combinar cromossomos e gerar soluções melhores[20]. Os indivíduos são selecionados com uma determinada probabilidade, combinando suas características na esperança de criar indivíduos mais aptos. A forma mais simples de cruzamento é o de um único ponto, isto é, uma posição no intervalo [1, tamanho do cromossomo - 1] é selecionado de forma aleatória. A figura 2.2 mostra um cruzamento com o ponto de corte igual a 3. Do lado esquerdo desta figura temos os cromossomos pais, que correspondem aos indivíduos selecionados. Do lado direito temos os cromossomos filhos, que recebem duas partes de cromossomo, uma de cada pai, com o tamanho de cada parte definido pelo ponto de corte sorteado.

2. Mutação: vista por muitos pesquisadores como uma operação secundária na evolução de um AG[20]. Ultimamente alguns trabalham têm feito a apreciação pela mutação crescer, sendo que o balanço entre cruzamento, mutação e seleção é o aspecto mais importante de um AG, no entanto este é um campo que ainda precisa ser investigado para que todas estas interações sejam esclarecidas. Finalmente, como mostra a figura 2.3 a mutação é uma operação que ocorre na cópia de um *bit*; cada *bit* copiado de uma *string* para outra pode sofrer mutação de acordo com a taxa utilizada, invertendo seu valor.

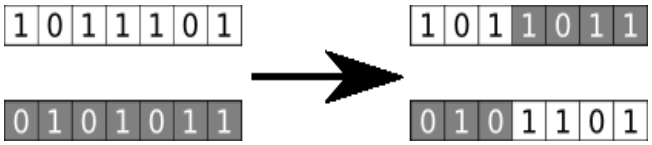


Figura 2.2: Cruzamento com ponto de corte 3.

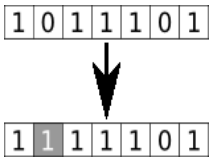


Figura 2.3: Mutação no segundo *bit*.

Na figura 2.4 é apresentado um fluxograma de um AG simples.

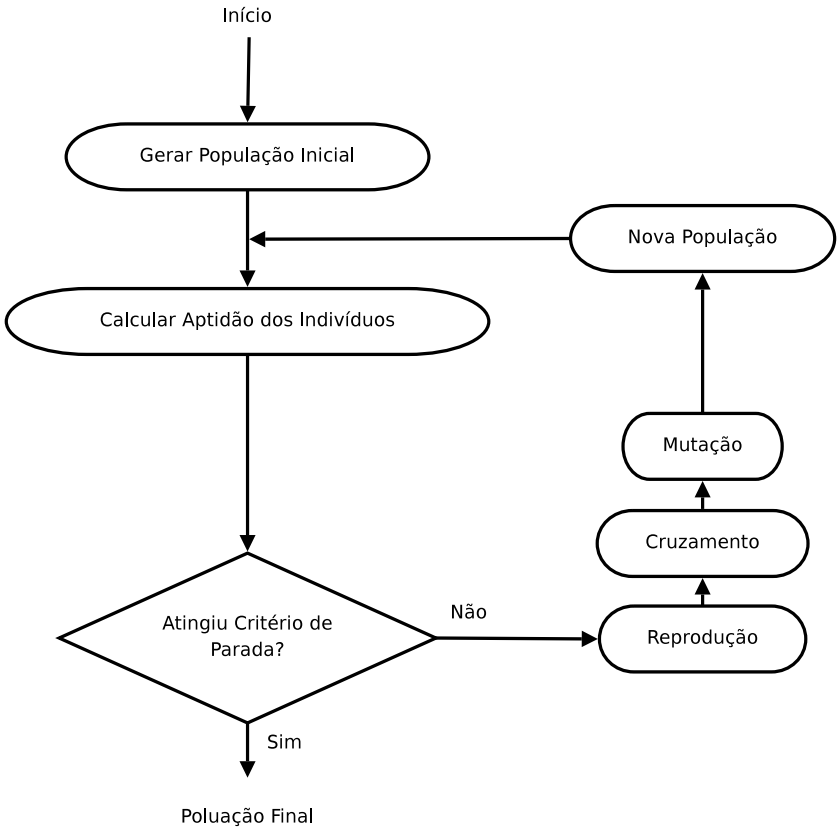


Figura 2.4: Fluxograma de um algoritmo genético simples.

Capítulo 3

Clusterização

O processo de clusterização (ou agrupamento), consiste basicamente em agrupar objetos semelhantes em um mesmo grupo. Segundo Jain e Dubes[13], classificar objetos de acordo com suas similaridades é a base de uma grande parte da ciência, além de ser um dos modos fundamentais para o entendimento e aprendizado. Everitt[6] *apud* [13] documenta algumas das seguintes definições de cluster:

- "Um cluster é um conjunto de entidades que são *similares*, e entidades de diferentes clusters não são similares."
- "Um cluster é uma agregação de pontos no espaço de busca tal que a *distância* entre qualquer dois pontos no cluster é menor que a distância entre qualquer ponto no cluster e qualquer ponto fora dele."
- "Clusters podem ser descritos como regiões conectadas de um espaço multi-dimensional contendo uma relativa *alta densidade* de pontos, separado de outra região por uma região contendo uma relativa baixa densidade de pontos."

A definição formal do problema de clusterização adotada neste trabalho é a dada por Cole[2]:

> Um conjunto de *n* objetos $X = \{X_1, X_2, ..., X_n\}$ será clusterizado. Cada $X_i \in \Re^p$ é um vetor de atributos consistindo de *p* medidas reais que descrevem o objeto. Os objetos serão clusterizados em grupos não sobrepostos $C = \{C_1, C_2, ..., C_n\}$ (*C* sendo um cluster), onde *k* é o número de clusters, $C_1 \cup C_2 \cup ... \cup C_k = X, C_i \neq \emptyset$,

e $C_i \cap C_j = \emptyset$ para $i \neq j$. Os objetos contidos em um grupo devem ser mais similares entre eles do que os objetos em qualquer outro grupo, e o valor de *k* pode ser desconhecido. Se *k* é conhecido, o problema é referido como um problema *k-clustering*.

3.1 Medidas de Similaridade

Saber o quanto um determinado objeto é similar a outro é fundamental para um processo de clusterização; assim tornam-se necessários meios para medir este parâmetro. Segundo Cole[2] a mais utilizada é a distância Euclidiana,

$$d(X_i, X_j) = [\sum_{l=1}^{p}(x_{il} - x_{jl})^2]^{1/2}, \tag{3.1}$$

a qual determina a distância em linha reta de um ponto até outro, contendo *p* atributos. Diversas outras medidas de similaridade existem [19][13][6], porém por simplicidade e por não ser o objetivo deste trabalho realizar um estudo exaustivo dos métodos de clusterização não apresentaremos outras medidas.

3.2 Métodos de Clusterização

Diversos métodos de clusterização existem[19], tais como hierárquico, particionamento, grafo-teórico, baseado em densidade, baseado em modelo, redes neurais, *grid-based*, *soft-computing*. Estes métodos de clusterização foram desenvolvidos levando em consideração diferentes princípios de indução. Neste trabalho apresentaremos apenas os algoritmos tradicionais de clusterização[2].

3.2.1 Métodos Hierárquicos

Tais métodos trabalham de forma recursiva particionando os dados de maneira “top-down” (divisivos) ou “bottom-up” (aglomerativos):

1. Aglomerativos: no início do algoritmo cada dado é um cluster que vai sendo agrupado com outro cluster a cada iteração, formando assim clusters com mais elementos. No final todos os elementos pertencem ao mesmo cluster.

2. Divisivos: no início do algoritmo existe apenas um cluster, que engloba todos os dados, e a cada iteração estes vão sendo divididos para formar novos clusters. No final cada elemento é um cluster.

O resultado de ambos os métodos é um dendrograma, como o mostrado na figura 3.1, o qual deve ser cortado no nível de similaridade desejado para obtenção do agrupamento apropriado.

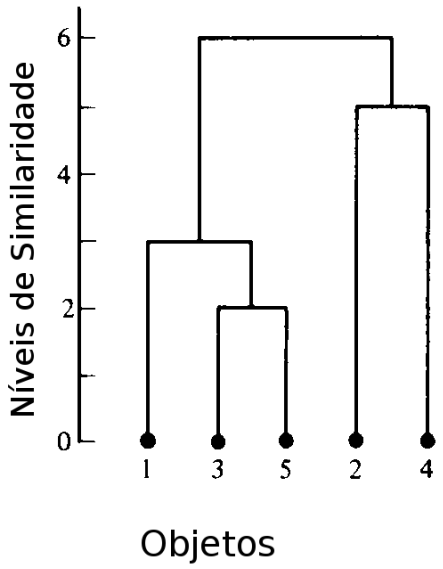


Figura 3.1: Dendograma de uma clusterização hieráquica.[14]

A aglomeração ou divisão em tais métodos podem ser feitos de formas diferentes, permitindo classificar os métodos de acordo como a medida de similaridade é calculada:

1. Ligação individual: considera a distância entre dois clusters sendo igual a menor distância de qualquer membro de um cluster em relação a qualquer membro de outro cluster.

2. Ligação completa: considera a distância entre dois clusters sendo igual a maior distância de qualquer membro de um cluster em relação a qualquer membro de outro cluster.

3. Ligação média: considera a distância entre dois clusters sendo igual a distância média entre todos os mebros de um cluster em relação a todos os membros de outro cluster.

Tais métodos apresentam alguns problemas. O ligação individual pode gerar cluster alongados, pois este acaba unificando clusters que são ligados por poucos pontos, como mostrado na

figura 3.2. Seria mais homogêneo agrupar A com C ou B com C, pois a distância média entre os elementos de A e C ou B e C é menor que a distância média entre os elementos de A e B; no entanto como o critério de de agrupamento é a menor distância entre membros de clusters distintos, o agrupamento entre A e B é realizado, unindo então os elementos menos similares, ou seja, que estão mais distantes.

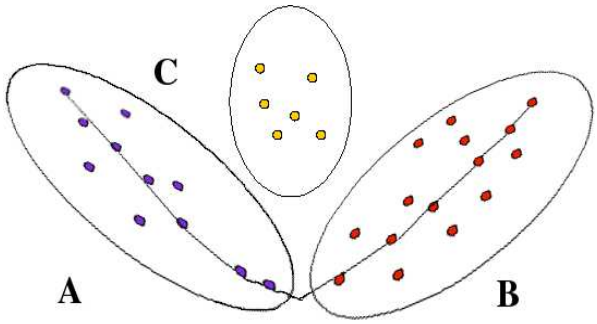


Figura 3.2: Ligação individual aglomera A e B.

O ligação média pode fazer com que um cluster alongado seja dividido e porções de clusters alongados vizinhos sejam agrupadas. O método ligação completa elimina o problema apresentado pelo ligação individual, sendo que a diferença entre eles é o modo de calcular a similaridade entre clusters, representada graficamente na figura 3.3.

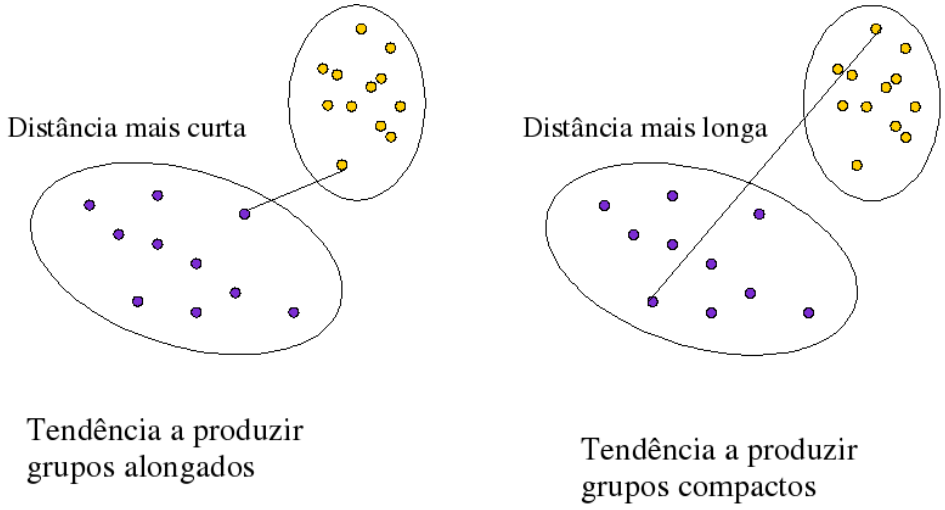


Figura 3.3: Ligação individual x Ligação completa.

Uma vantagem dos métodos hierárquicos é que eles não geram apenas uma partição, mas múltiplas partições aninhadas, o que permite que diferentes usuários selecionem diferentes partições apenas ajustando o nível de similaridade desejado. Tomando a figura 3.1 como referência, se um determinado usuário escolher a distância 2, este obteria 4 clusters, o primeiro com o objeto 1, o segundo com os objetos 3 e 5, o terceiro com o objeto 2 e o último com o objeto 4.

Uma das principais desvantagens dos métodos hierárquicos é a inabilidade de tratar um grande volume de dados, pois sua complexidade é de $O(m^2)$, onde m é o número de objetos, além disso clusterizar um grande número de objetos utilizando tais algoritmos causa um custo elevado de E/S[19].

3.2.2 Métodos de Particionamento

Os métodos de particionamento caracterizam-se por necessitarem de um número definido pelo usuário para o número de clusters. A partir disso, os métodos funcionam movendo os dados entre os *k* clusters. Uma desvantagem deste processo é que o número de clusters deve ser conhecido, ou então uma enumeração deve ser feita com diversas valores diferentes para *k* e o algoritmo deve ser executado utilizando cada um desses parâmetros. Como tal processo é inviável devido ao custo computacional, diversos métodos foram propostos para determinar *k*. Tais métodos são normalmente heurísticas, envolvendo o cálculo do critério de clusterização para diferentes valores de *k*. Mais detalhes sobre tais métodos podem ser encontradas em [19] e [10].

Os algoritmos de minimização de erros, por sua simplicidade e eficiência em tempo de execução são os métodos mais utilizados neste processo, sendo o K-Médias o mais popular destes. Mais detalhes de tais métodos e descrição de outros: grafo-teórico, baseado em densidade, baseado em modelos, *soft-computing*, etc podem ser encontrados em [19] e [10].

A idéia básica destes algoritmos é minimizar um determinado critério de erro, sendo a soma dos erros quadráticos a métrica mais empregada. Para realizar este cômputo é utilizada a distância Euclidiana.

No algoritmo K-Médias, define-se a posição dos *k* clusters, então os elementos mais próximos de cada cluster tornam-se parte deste, assim o centro de cada cluster é recalculado como

a média de todas as instância pertencentes a ele[1]. A seguir são apresentados os passos do algoritmo:

1. Inicie *k* clusters em posições arbitrárias.
2. Para cada objeto, mova-o para o cluster no qual reduz o critério de erro ao máximo.
3. Compute o centro de cada cluster como sendo a média das posições de seus elementos.
4. Repita o segundo passo até um determinado número de interações ou até que os objetos não reduzam mais o critério de erro.

A complexidade do K-Médias, uma das razões de sua popularidade, em *T* iterações em um conjunto de *m* objetos, cada um contendo *N* atributos é de $O(T * K * m * N)$. Outra razão de sua popularidade é a facilidade de implementação, fácil interpretação, velocidade de convergência e adaptação a dados esparsos [3] *apud* [19].

O maior problema dos métodos de particionamento é a determinação da quantidade de partições iniciais, bem como sua distribuição. Os algoritmos são bastante sensíveis a esta configuração inicial, o que pode fazer a diferença entre uma convergência a um mínimo local e um ótimo global. As figuras 3.4 e 3.5 mostram este problema, onde os clusters finais possuem diferentes objetos devido a uma inicialização diferente.

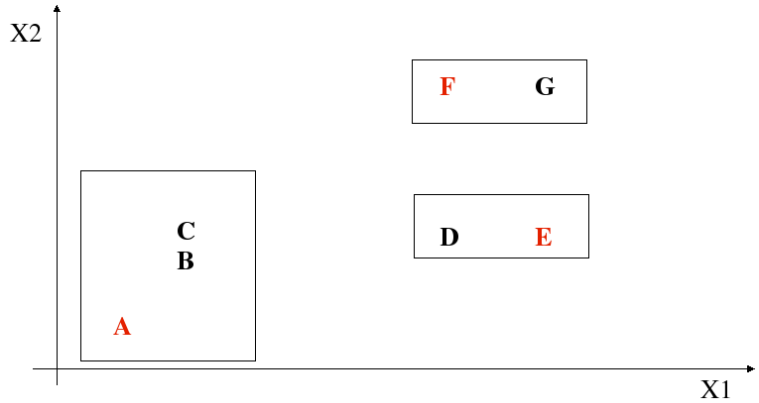


Figura 3.4: Inicialização com A, F e E.[4]

[1]Um applet com a implementação da técnica pode ser encontrado em http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/AppletKM.html

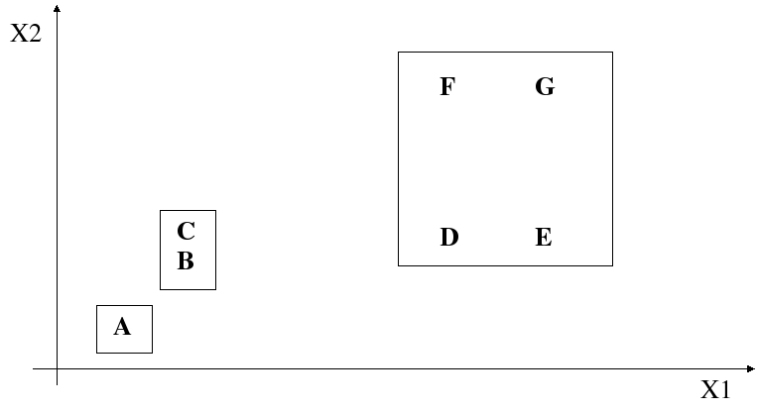


Figura 3.5: Inicialização com A, B e C.[4]

Capítulo 4

Desenvolvimento

Neste capítulo, apresentamos como o trabalho foi desenvolvido. Descrevemos como foi a codificação do AG para resolver os problemas de clusterizações adotados no estudo e como codificamos o K-Médias para resolver os mesmos problemas. Apresentamos a libgabin, a biblioteca que desenvolvemos para implementação do AG utilizado neste trabalho. Por fim, apresentamos os tempos de execução de cada algoritmo e a clusterização que cada um realizou, sendo possível realizar algumas comparações da técnica empregada com um algoritmo de clusterização clássico.

4.1 Casos de Uso

Na sequência são apresentados os dois casos de uso utilizados no trabalho, bem como a metodologia utilizada para resolvê-los.

4.1.1 Caso de Uso 1: Clusterização de uma Imagem Digital em Tons de Cinza

Para este caso de uso, entende-se que clusterização é o processo de agrupar regiões com características similares e que estejam suficientemente próximas. O dado de entrada para este processo é uma imagem digital em 256 níveis de cinza. Um exemplo é mostrado na figura 4.1. Esta imagem apresenta sub-regiões representadas em diferentes níveis de cinza com contraste suficiente para diferenciá-las.

Cada sub-região desta imagem pode representar, por exemplo, uma unidade de manejo definida após a análise de dados geo-estatísticos num processo de agricultura de precisão. A

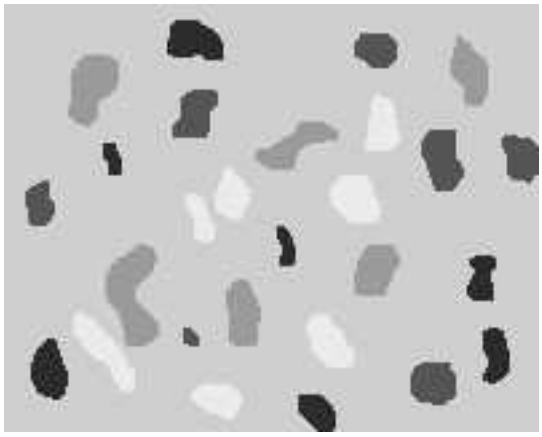


Figura 4.1: Imagem em 256 Tons de Cinza a ser Clusterizada.

cada unidade de manejo está associada, por exemplo, uma necessidade de um insumo representada pela intensidade de cinza correspondente.

Considerou-se dois fatores neste processo de clusterização: similaridade e distância. O grau de similaridade foi medido através da diferença entre os níveis de cinza; duas sub-regiões com níveis de cinza próximos apresentam necessidades similares do mesmo insumo. A distância considerada foi a menor distância euclidiana entre as bordas de duas sub-regiões.

Pré-processamento

Para realizar a clusterização da figura 4.1 aplicou-se algumas operações de pré-processamento, visando extrair da imagem digital dados necessários a execução dos algoritmos, sendo elas:

1. Reconhecimento de cada sub-região (posição e cor).

2. Determinação da menor distância entre as bordas de cada sub-região.

O reconhecimento de cada sub-região é realizado através do método de segmentação por regiões crescentes[9]. Após as regiões terem sido identificadas, verifica-se se algum dos pixels da oito-vizinhança de cada sub-região possui a cor de fundo da imagem; caso esta verificação seja verdadeira, o pixel é adicionado como pertencente a borda desta sub-região.

A cada sub-região identificada na imagem atribui-se um identificador alfa-numérico. Associado a este identificador tem-se a cor e a posição, definida pelo centróide. O centróide é a média da posição de todos os pontos da sub-região.

Para calcular a menor distância entre as bordas de duas sub-regiões percorre-se todos os pixels das bordas de ambas calculando-se a distância euclidiana entre todos os possíveis pares de pixel. A menor destas distâncias é então armazenada. No algoritmo K-Médias, utilizamos a distância entre centróides, sendo assim, o pré-processamento tornou-se menos custoso computacionalmente para este algoritmo.

Codificação Cromossômica Empregada no AG

Para entender como o cromossomo é organizado, faremos uso inicialmente de uma representação matricial, como a mostrada na figura 4.2.

	A1	B1	C1	D1	E1	F1	G1	H1	I1
A1		1	0	0	0	1	0	1	0
B1			0	1	0	0	1	0	0
C1				0	0	0	1	0	0
D1					0	1	0	0	0
E1						0	0	0	0
F1							0	1	0
G1								0	0
H1									1
I1									

Figura 4.2: Codificação de um cromossomo representando as conexões entre as sub-regiões de uma imagem.

Esta estrutura de codificação pode ser assim entendida: a sub-região rotulada como A1 está conectada às regiões B1, F1 e H1; já a região B1 está conectada com as regiões D1 e G1 e assim sucessivamente. Além disso, para determinar se determinados elementos estão no mesmo cluster, adotou-se uma regra transitiva, isto é, se A1 está ligado com B1 e B1 está ligado com D1 e G1, então considera-se A1, B1, D1 e G1 como pertencentes ao mesmo cluster.

A forma vetorial desta matriz é apresentada na figura 4.3 e esta é a representação utilizada em memória durante a execução do AG.

Figura 4.3: Representação vetorial do cromossomo apresentado na Figura 4.2

Função Objetivo

A função objetivo foi elaborada de forma a punir ligações ruins e incentivar boas ligações. A decisão entre punir ou incentivar ligações está baseada em limiares definidos pelo usuário. Há dois limiares: limiar de cor e limiar de distância; caso a diferença de cor ou a distância seja maior que o limiar especificado a ligação é punida, caso contrário ela é incentivada.

Para cada alelo igual a 1, que indica ligação entre duas sub-regiões, verifica-se se a menor distância entre suas bordas é menor que o limiar de distância; também verifica-se se a diferença entre as cores das sub-regiões é menor que o limiar de cor.

Caso ambas verificações sejam verdadeiras a ligação é incentivada, contribuindo positivamente no *fitness* do cromossomo. Caso contrário a contribuição será negativa, punindo esta ligação.

A contribuição positiva é estimada pela equação 4.1, enquanto a contribuição negativa é estimada pela equação 4.2.

$$\frac{1}{|CB_i - CB_j| + 1} + \frac{1}{d(B_i, B_j)}, i \neq j \tag{4.1}$$

$$\left(1 - \frac{1}{|CB_i - CB_j| + 1}\right) + \left(1 - \frac{1}{d(B_i, B_j)}\right), i \neq j \tag{4.2}$$

sendo que:

- CB_i e CB_j: cor da sub-região i e cor da sub-região j.
- $d(B_i, B_j)$: distância euclidiana entre as sub-regiões i e j.

Em suma, o grau de punição ou incentivo é influenciado pela diferença da cor e pela distância, sendo que ligações de regiões próximas e de cores *iguais* terão *fitness* maior que as ligações de regiões próximas porém com cores *não iguais*.

O *fitness* do cromossomo é calculado através da soma de todas as contribuições obtidas através das equações 4.1 e 4.2. Deste modo, a função objetivo pode retornar valores negativos;

para contonar este problema utilizamos o mapeamento entre o valor da função objetivo para a forma de *fitness* como apresentado no capítulo 3 de Goldberg[8].

O Algoritmo K-Médias

Assim como para aplicação do AG, no K-Médias também faz-se necessária a execução da etapa de pré-processamento para obter a posição e cor de cada sub-região. Depois, aplicou-se o algoritmo como descrito na seção 3.2.2.

O critério de erro empregado é a distância euclidiana entre as tuplas (x, y, cor) associadas à cada sub-região, sendo x e y as coordenadas de seu centróide. Uma representação gráfica de uma possível distribuição de sub-regiões pode ser vista na figura 4.4.

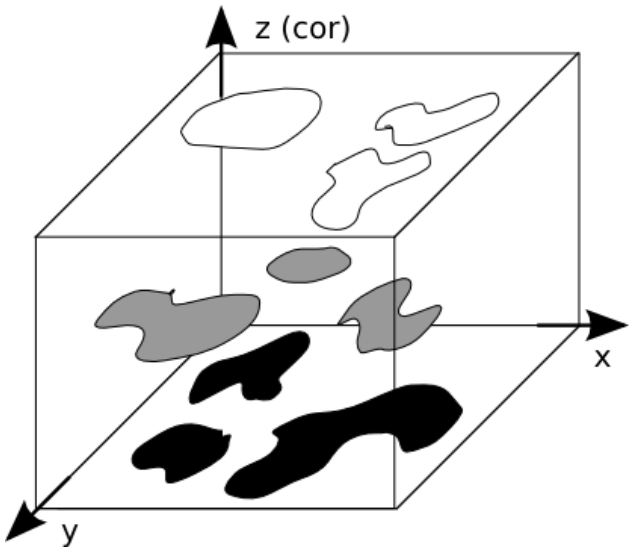


Figura 4.4: Representação das sub-regiões em um espaço tri-dimensional.

No plano XY temos o centróide da sub-região e no eixo Z temos sua cor. A seleção das sementes é feita por escolha aleatória sem reposição.

4.1.2 Caso de Uso 2: Clusterização da base Íris

A base de dados Íris publicada por Fisher (1936)[7] tem sido utilizada amplamente para testes de novos algoritmos de clusterização[25]. Nesta estão contidas a largura e altura da pétala e da sépala, bem como outros valores derivados destes. Ao todo são 150 amostras de 3 espécies diferentes, sendo 50 amostras de cada espécie. A tabela 4.1 apresenta algumas das entradas

encontradas em `http://www.statlab.uni-heidelberg.de/data/iris/`.

Altura da Sépala	Largua da Sépala	Altura da Pétala	Largura da Pétala	Espécie
5.1	3.5	1.4	0.2	"setosa"
4.9	3	1.4	0.2	"setosa"
4.7	3.2	1.3	0.2	"setosa"
4.6	3.1	1.5	0.2	"setosa"
7	3.2	4.7	1.4	"versicolor"
6.4	3.2	4.5	1.5	"versicolor"
6.9	3.1	4.9	1.5	"versicolor"
5.5	2.3	4	1.3	"versicolor"
6.3	3.3	6	2.5	"virginica"
5.8	2.7	5.1	1.9	"virginica"
7.1	3	5.9	2.1	"virginica"
6.3	2.9	5.6	1.8	"virginica"

Tabela 4.1: Exemplares da base de dados Íris.

Neste caso de uso, não foi necessário nenhum pré-processamento, apenas carregar em memória os dados a serem clusterizados.

As informações utilizadas para se realizar a clusterização foram a área da pétala e a área da sépala, tanto para o AG quanto para o K-Médias, calculadas a partir da multiplicação dos valores de largura e altura.

Codificação Cromossômica Empregada no AG

Neste problema a codificação adotada foi a mesma apresentada no caso de uso 1 e exemplificado pelas figuras 4.2 e 4.3.

Função Objetivo

Assim como no caso de uso 1 a função objetivo também foi codificada de forma a punir ligações ruins e incentivar boas ligações. A decisão de punir ou incentivar ligações também é baseado em dois limiares: limiar de área da pétala e limiar de área da sépala. Quando o valor absoluto da diferença entre a área das sépalas ou entre a área das pétalas é maior que os limiares definidos, a contribuição da ligação é negativa, caso contrário positiva. A contribuição negativa é dada por:

$$5 * \left(1 - \frac{1}{|AS_i - AS_j| + 1}\right) + 5 * \left(1 - \frac{1}{|AP_i - AP_j| + 1}\right), i \neq j \tag{4.3}$$

e a contribuição positiva é dada por:

$$\frac{1}{|AS_i - AS_j| + 1} + \frac{1}{|AP_i - AP_j| + 1}, i \neq j \tag{4.4}$$

sendo que:

- AS_i e AS_j: área da sépala das plantas i e j.
- AP_i e AP_j: área da pétala das plantas i e j.

Através dos testes, verificou-se empiricamente que multiplicar a contribuição negativa por 5 ajudava o algoritmo convergir mais rapidamente, pois muitas vezes a solução encontrada continha ligações ruins, pois a contribuição negativa não era suficiente para dar a este indivíduo um *fitness* menor do que a de um indivíduo que continha apenas ligações boas.

O *fitness* do cromossomo é calculado através da soma de todas as contribuições obtidas através das equações 4.4 e 4.3. Deste modo, a função objetivo pode retornar valores negativos; para contonar este problema utilizamos o mapeamento entre o valor da função objetivo para a forma de *fitness* como apresentado no capítulo 3 de Goldberg[8].

O Algoritmo K-Médias

Neste caso de uso também foi aplicado o K-Médias de acordo com o descrito na seção 3.2.2. O critério de erro adotado foi a distância euclidiana, porém num plano bi-dimensional, como pode ser visto na figura 4.5. As sementes também foram escolhidas de forma aleatória e sem reposição.

4.2 Questões de Implementação

Para cada caso de uso, tanto a implementação utilizando AG quanto a que utiliza K-Médias compartilham parte do código. No caso de uso 1, todo o pré-processamento é utilizado em ambos os algoritmos. No caso de uso 2, a parte de código compartilhada é relativa ao carregamento

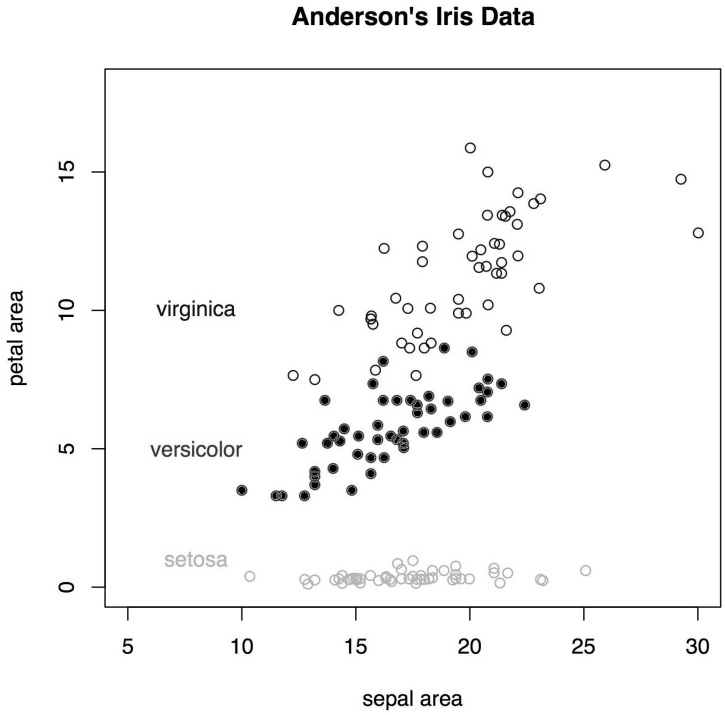


Figura 4.5: Gráfico das 3 espécies com os valores de x e y sendo a área da sépala e da pétala.

da base Iris para memória. Portanto, em abas as implementações têm-se uma fator de tempo constante até a execução do método de clusterização.

Para dar suporte à implementação do AG, desenvolvemos uma biblioteca que suporta basicamente o *framework* apresentado inicialmente por John Holland denominada Binary Genetic Algorithm Library[1]. A linguagem escolhida foi C e através da sua utilização é possível codificar um AG em poucos passos, como pode ser visto no código 4.1, que implementa um exemplo básico[20], a otimização de strings contendo valores 1.

Código 4.1: Demonstração do uso da biblioteca libgabin

```
1  #include <gabin.h>
2
3  float
4  objective (char *chrom)
5  {
6          int i, j;
7          float objective = 0;
8
9          for (i = 0; i < CHROM_SIZE; i++) {
10                 if (chrom[i] == '1') {
```

[1]https://sourceforge.net/projects/libgabin

```
11                          objective += 1;
12                  }
13          }
14
15          return objective;
16  }
17
18  int
19  main (int argc, char **argv)
20  {
21          set_chrom_size (10);
22          set_pop_size (30);
23          set_n_gen (30);
24          set_cross_type (AG_ONE_POINT);
25          set_elite_size (4);
26          set_objective_func (objective);
27          set_prob_cross (0.90);
28          set_prob_mut (0.01);
29          run_ag ();
30
31          return 0;
32  }
```

4.3 Resultados Obtidos

Nesta seção são apresentados alguns dados relacionados às implementações realizadas para ambos os casos de uso. Apresentamos resultados do tempo de execução e da qualidade da solução.

O tempo de execução é apenas informativo, pois os algoritmos são distintos e realizam operações de pré-processamento distintas, inviabilizando comparações. A qualidade da solução para o caso de uso 1 é subjetivo e depende da interpretação de um especialista, no caso de uso 2 a qualidade da solução é verificada em relação a base de dados original.

Nas implementações que utilizaram o K-Médias adotamos a abordagem de utilizar o número ideal de sementes, pois existem diversos métodos para determinar este número e estes nem sempre geram bons resultados[22]. O AG possui a vantagem de obter o número ideal de clusters automaticamente.

Antes de discutir os resultados é importante dizer que durante os testes e ajustes do AG percebeu-se que este poderia ser resolvido de forma determinística, isto é, para cada posição do cromossomo bastaria avaliar se o *bit* contendo valor 1 contribuiria positivamente ou negativamente para o *fitness*. Sendo assim bastaria ligar, configurar em 1, o *bit* que apresentou uma

contribuição positiva ao fitness e desligar o *bit* que apresentou uma contribuição negativa. Realizamos a implementação desta abordagem para verificar que o problema poderia ser resolvido de tal forma, assim apresentamos também seus resultados.

4.3.1 Caso de Uso 1

As execuções foram realizadas nas imagens 4.6(a), 4.6(b) e 4.6(c). O equipamento utilizado nos testes[2] é apresentado a seguir:

- CPU: Intel(R) Celeron(R) D CPU 3.60GHz
- Memória RAM: 1 x 2 Gb
- Freqüência do barramento: 533 MHz
- Swap: 2 Gb
- Sistema operacional Ubuntu 8.04.1

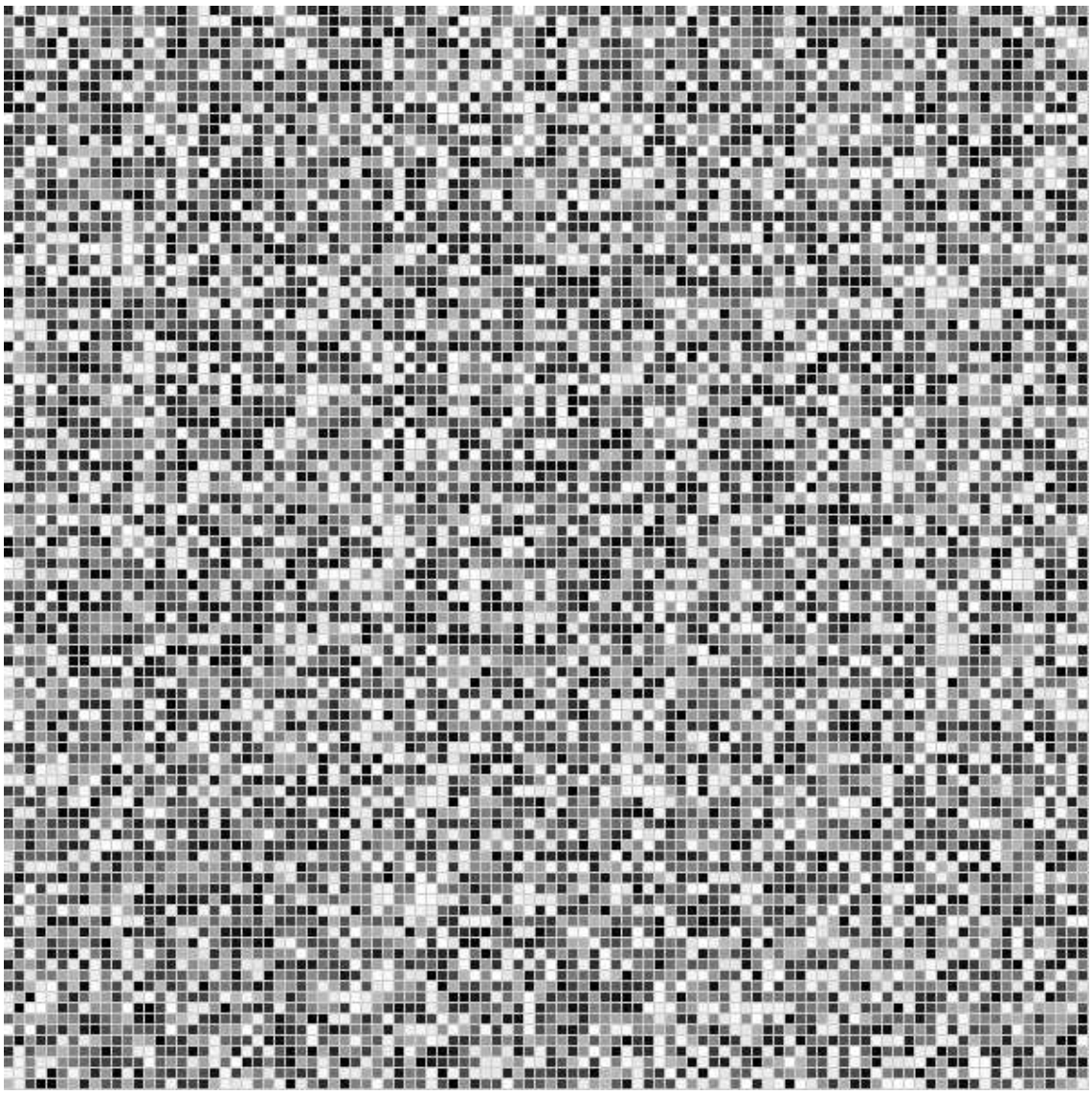
Algoritmo Genético e Algoritmo Determinístico (AD)

Tanto o AG, quanto o AD podem atingir os mesmo resultados, porém a diferença no tempo de execução é elevada. A tabela 4.2 resume os tempos de execução do AG e do AD coletados da saída do comando *time*, sendo apresentado apenas o tempo *real*, que representa o tempo desde o início do processo até o seu término. Os tempos representam uma média de 10 execuções.

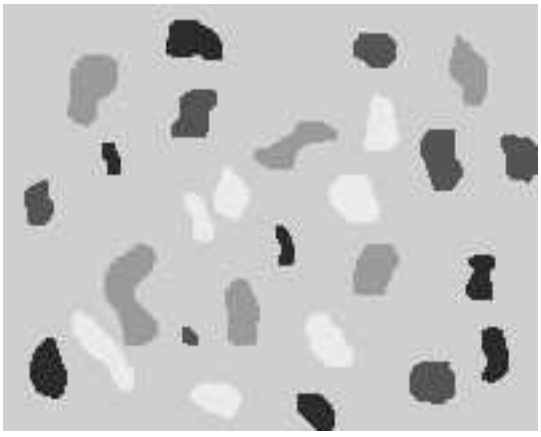
Algoritmo Genético					
Figura 4.6(a)		Figura 4.6(b)		Figura 4.6(c)	
real	XXX	*real*	0m1.900s	*real*	59m56.306s
Implementação Determinística					
Figura 4.6(a)		Figura 4.6(b)		Figura 4.6(c)	
real	53m0.325s	*real*	0m1.384s	*real*	0m0.302s

Tabela 4.2: *Tempo de execução para as figuras 4.6(a), 4.6(b) e 4.6(c)*

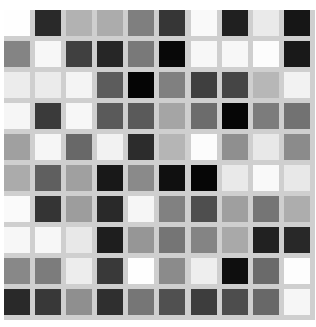
[2] `http://h10025.www1.hp.com/ewfrf/wc/document?docname=c01064741&lc=pt&dlc=pt&cc=br&lang=pt&product=3401287#`



(a) Imagem sintética com quadrados distribuídos em uma matriz 100x100.



(b) Imagem apresentada na proposta.



(c) Imagem sintética com quadrados distribuídos em uma matriz 10x10

Figura 4.6: Imagens utilizadas no caso de uso 1

Um fator relevante a ressaltar em relação a utilização do AG é o tamanho do cromossomo, o qual aumenta a medida que o volume dos dados de entrada cresce. Para as imagens utilizadas os tamanhos dos cromossomos foram 49.995.000, 325 e 4.950 posições respectivamente. Isto faz com que este AG não seja uma solução escalável, pois a medida que o tamanho do cromossomo cresce os requisitos de tempo de processamento e memória principal também crescem. No caso da figura 4.6(a) o tamanho do cromossomo impediu a execução do algoritmo devido a grande quantidade de memória necessária para representar a população, por isso não há tempos de execução para esta imagem.

Os limiares de cor e distância adotados nas figuras 4.6(a) e 4.6(c) foram de 15 e 5 respectivamente, enquanto na imagem 4.6(b) estes foram de 15 e 25 respectivamente.

A figura 4.7 apresenta o resultado da figura 4.6(a) após a aplicação do AD.

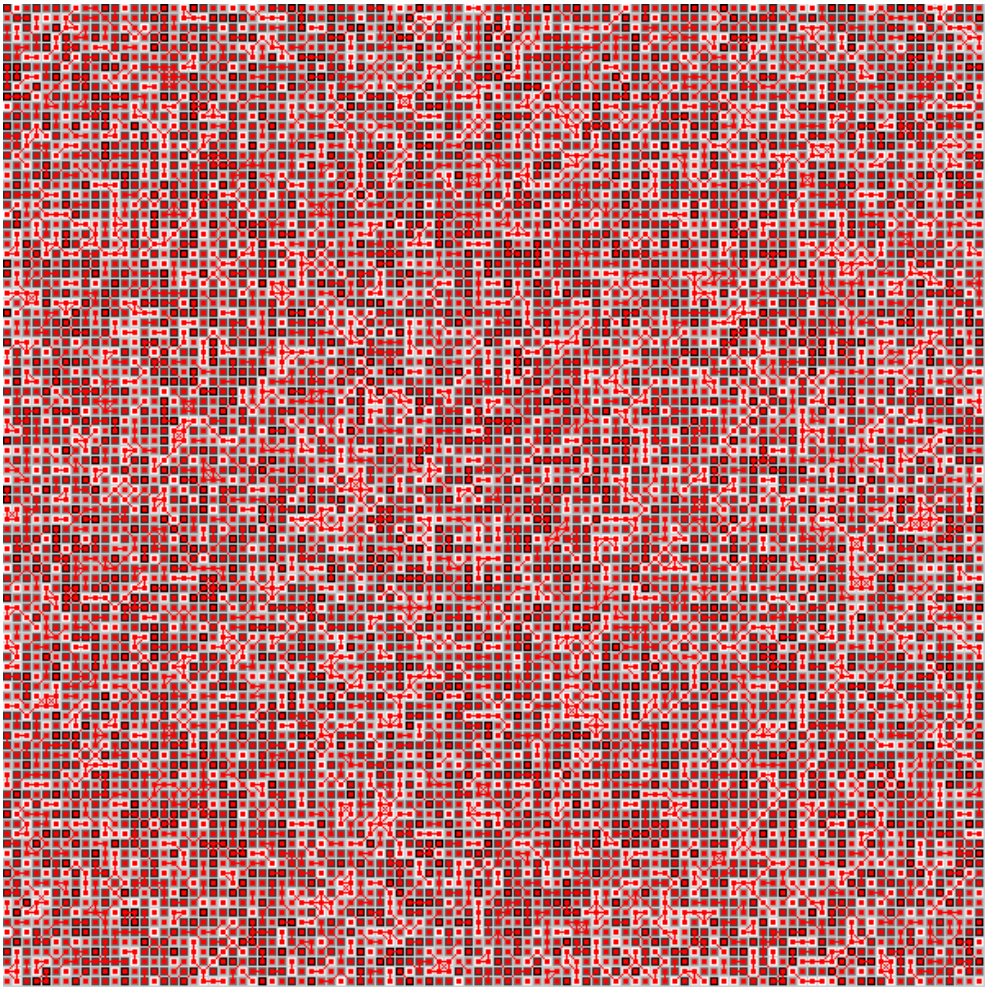
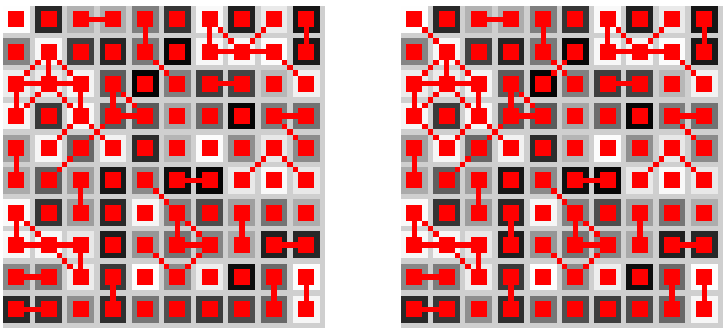


Figura 4.7: Figura 4.6(a) clusterizada.

A figura 4.8 apresenta lado a lado o resultado da imagem 4.6(c) clusterizada pelo AG e pelo algortimo determínistico. Apesar das pouquíssimas diferenças este é um exemplo onde o AG encontrou apenas uma solução sub-ótima, enquanto o AD encontra sempre a solução ótima que pode ser alcançada de acordo com a função objetivo.



(a) Figura 4.6(c) clusterizada pelo AG.

(b) Figura 4.6(c) clusterizada pelo método determinístico.

Figura 4.8: Figura 4.6(c) clusterizada.

K-Médias

A tabela 4.3 apresenta o tempo de execução do algoritmo K-Médias para segmentar as imagens apresentadas na figura 4.6.

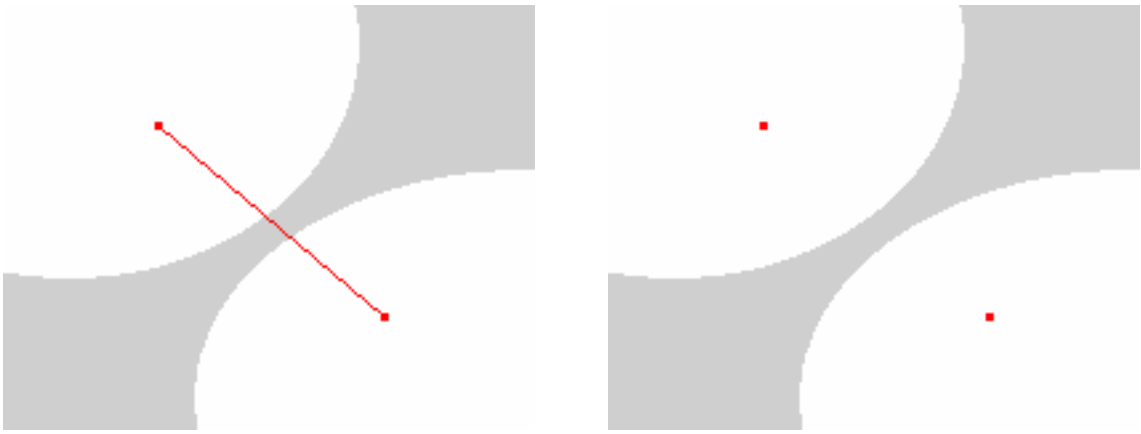
K-Médias					
Figura 4.6(a)		Figura 4.6(b)		Figura 4.6(c)	
real	5m52.491s	*real*	0m0.286s	*real*	0m0.131s

Tabela 4.3: *Tempo de execução para as figuras 4.6(a), 4.6(b) e 4.6(c)*

Como pode ser observado, o tempo de execução é bem menor que os obtidos pelo AD. Apesar dos dois algoritmos compartilharem boa parte do código de pré-processamento, na implementação determinística calculamos a distância entre as bordas de duas regiões. Tal cálculo possui custo superior à determinação da distância entre centróides, realizada no K-Médias. Uma borda contém diversos pontos que devem ser comparados com todos os outros pontos de outra borda para determinar a menor distância entre elas.

Para o AG e o AD deve-se utilizar, como critério de similaridade baseado em distância, a menor distância entre bordas, ao invés da distância entre centróides. A utilização da distância

entre centróides faz com que sub-regiões grandes, mesmo que vizinhas, não sejam agrupadas, como visto na figura 4.9(b).



(a) Clusterização baseada na distância entre bordas

(b) Clusterização baseada na distância entre centróides

Figura 4.9: Clusterização pelo AG considerando distâncias entre bordas e centróides.

Na figura 4.9(a) temos o cluster formado pelas duas sub-rgiões utilizando como critério de clusterização a menor distância entre bordas, ao passo que na figura 4.9(b) o cluster não é formado pois os centróides estão afastados.

Finalmente, apesar do K-Médias convergir mais rapidamente a solução encontrada normalmente é um ótimo local. Fato comprovado pela figura 4.11 quando comparado com a solução ótima encontrada pelo AG e o AD na figura 4.10.

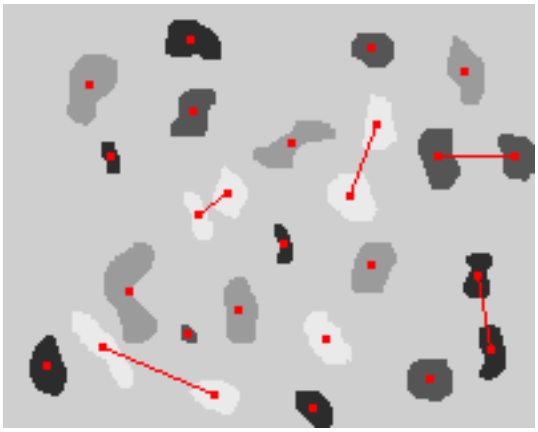
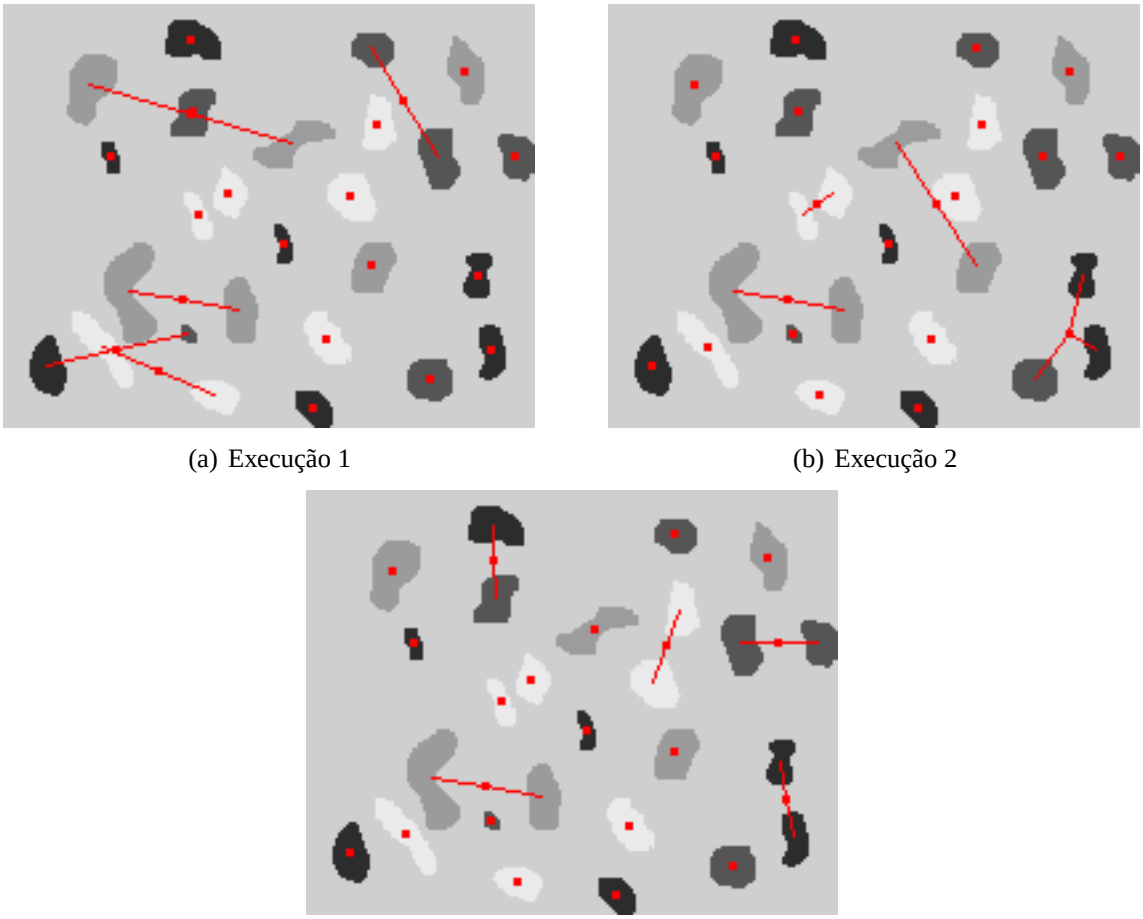


Figura 4.10: Solução ótima encontrada pelo AG e o AD.



(a) Execução 1 (b) Execução 2

(c) Execução 3

Figura 4.11: Clusterização realizada pelo K-Médias.

4.3.2 Caso de Uso 2

Nesta seção são apresentados os resultados obtidos na clusterização da base Íris para cada um dos métodos.

Algoritmo Genético e Algoritmo Determinístico

Como mencionado anteriormente, o AG resolve o problema determinando se dois elementos devem ser ligados ou não, de acordo com suas similaridades. Resgatou-se esta afirmação pois neste caso de uso, esta métrica de similaridade mostrou-se insuficiente para que o AG determinasse os clusters adequadamente.

Dados que apresentam sobreposição acabam sendo agrupados pois os elementos da fronteira são parecidos e/ou próximos, como pode ser visto na figura 4.12. O cluster acaba se alongando

devido a regra transitiva utilizada na interpretação do cromossomo, conforme apresentado na sub-seção 4.1.1.

Os limiares de área de sépala e área de pétala adotados neste caso de uso foram de 2,5 e 1,0 respectivamente.

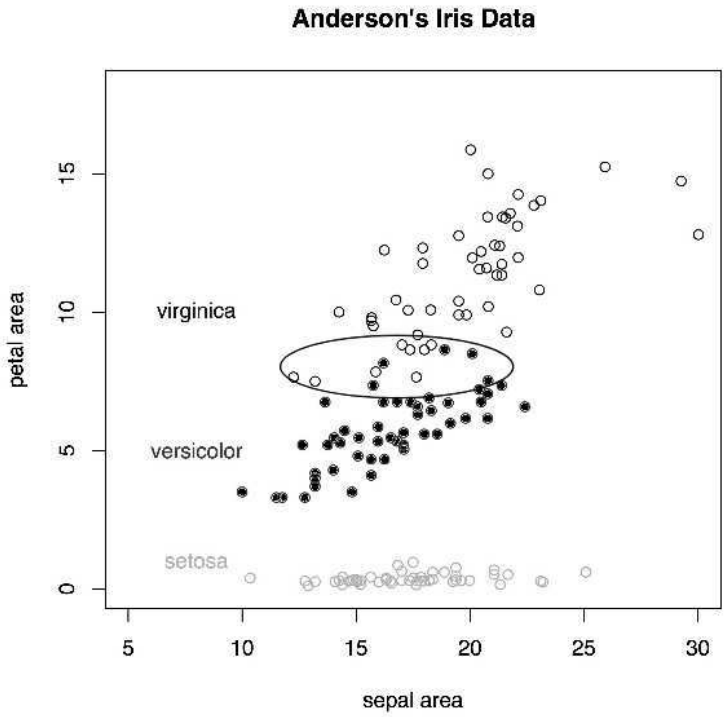


Figura 4.12: Região com sobreposição de dados.

Antes de mostrar os resultados obtidos, são apresentados os tempos de execução gastos pelo AG e pelo AD, e que podem ser verificados na tabela 4.4.

Algoritmo Genético	
real	13m9.811s
Algoritmo Determinístico	
real	0m0.029s

Tabela 4.4: *Tempo de execução do AG e AD para a base de dados Íris.*

Os métodos apresentam os mesmos resultados. Entretanto o AG utilizou mais tempo para alcançá-los. A redução do tempo de execução pode levar o AG a um ótimo local, pois o algirtmo pode não convergir.

Em ambas as implementações a espécie *I. setosa* é agrupada num único cluster, porém as espécies *I. virginica* e *I. versicolor* são agrupadas noutro cluster. Com exceção de algumas plantas que ficaram sozinhas devido aos limites utilizados. O ajuste dos limiares é bastante sensível, fazendo com que todas as espécies sejam agrupadas num único cluster caso adotem-se limiares maiores. As execções são os tês pontos no canto superior direito da figura 4.5. A representação gráfica desta solução é apresentada na figura 4.13.

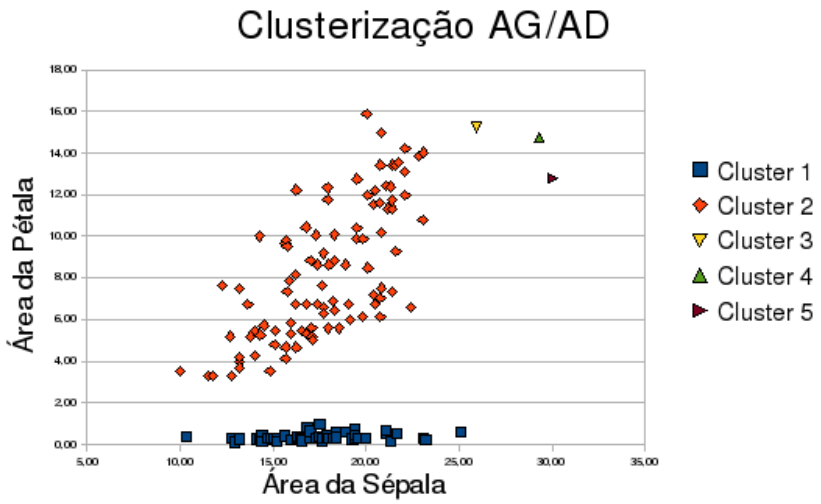


Figura 4.13: Clusterização da base Íris pelo AG.

K-Médias

Assim como no caso de uso 1, também adotou-se a convenção de utilizar o número ideal de sementes, que no caso é 3. Para este caso de uso, o K-Médias apresentou tempo de execução menor, mas também foi incapaz de separar corretamente o conjunto de dados. A tabela 4.5 apresente o tempo de execução.

K-Médias	
real	0m0.009s

Tabela 4.5: *Tempo de execução do K-Médias para base de dados Íris.*

A figura 4.14 apresenta a clusterização realizada pelo K-Médias, que apesar de separar corretamente a espécie *I. setosa* o cluster com as *I. versicolor* contem erroneamente 11 espécies

da *I. verginica* e o cluster com as *I. virginica* são classificadas erroneamente 9 espécies da *I. versicolor*.

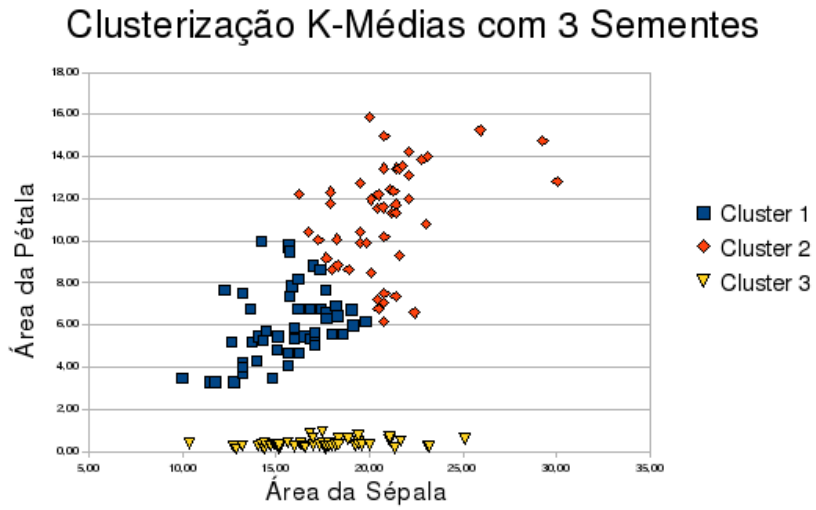


Figura 4.14: Clusterização da base Íris pelo K-Médias.

Capítulo 5

Considerações Finais

O objetivo proposto neste trabalho foi alcançado. Um Algoritmo Genético foi construído, aplicado ao processo de clusterização de dados e comparado com o algoritmo K-Médias. Porém, os resultados obtidos não foram os esperados. A princípio acreditava-se que um AG seria capaz de encontrar um bom, ou o melhor, conjunto de clusters presentes nos conjuntos de dados analisados. O que se observou é que nem sempre isso ocorre.

Acredita-se que o motivo deste comportamento inesperado deve-se à estrutura de codificação cromossômica empregada. Ela não possui, nem oferece ao AG, informações sobre os clusters que estão sendo construídos no processo evolutivo; ela apenas informa se um dado deve ser agrupado a outro dado, baseado na avaliação do critério de similaridade empregado. Os clusters são obtidos ao final do algoritmo através da observação do cromossomo com maior fitness e da regra de transitividade apresentada na seção 4.1.1.

Esta carência de informações sobre os clusters, que estão sendo construídos no processo evolutivo, também impedem a construção de métricas mais elaboradas para determinar se um dado elemento pertencerá ou não a um cluster.

Outra limitação da codificação empregada é o tamanho do cromossomo gerado. Ele cresce exponencialmente à medida que aumenta o número de dados a clusterizar. Para 100 dados o cromossomo apresenta 4.950 alelos; para 10.000 dados o cromossomo apresenta 49.995.000 alelos. Este crescimento exponencial inviabiliza a utilização desta codificação para conjuntos de dados maiores.

Apesar das limitações apresentadas pela codificação empregada no AG, este também apresenta vantagens. Permite determinar a quantidade de clusters ao final do processo, o que não acontece com o K-Médias, que deve ser aplicado diversas vezes com número de sementes difer-

entes e o resultado deve ser analisado por uma especialista para determinar sua qualidade, ou utilizar métodos heurísticos para estimar a quantidade de sementes.

Durante a avaliação do AG implementado observou-se um padrão de comportamento constante, o que possibilitou a construção de um algoritmo determinístico. O que permitiu isso foi o estabelecimento de limiares para punir ou não as ligações entre dados. Este algoritmo é muito mais eficiente e sempre encontra o resultado tido como ótimo pelo AG. O algoritmo determinístico tem comportamento linear, ou seja, avalia todas os agrupamentos dois a dois. Aqueles pares cuja métrica de avaliação de similaridade contribui positivamente para o fitness do indivíduo tem sua ligação ativada. Os pares que contribuem negativamente tem sua ligação desativada.

O algoritmo K-Médias, como já constatado na literatura, não apresenta bons resultados quando não se pode definir o número de clusters, nem quando a partição inicial é ruim. As vantagens já são conhecidas, sendo tempo de execução baixo, rapidez de convergência, baixa complexidade e facilidade de implementação.

Finalmente, quando os dados clusterizados apresentam clusters sobrepostos, isto é, que não podem ser separados linearmente, o AG empregado não consegue encontrar soluções melhores que o K-Médias, que também não encontra soluções ótimas. Isto acontece porque a função objetivo não foi capaz de distinguir os elementos que estão sobrepostos como sendo pertencentes a clusters distintos. Apesar destas constatações, acreditamos que explorar outras formas de codificação do AG podem melhorar a qualidade das soluções encontradas, mesmo que o AG desconheça o número de clusters presentes nos dados. Ao inserir esta informação no AG, acredita-se que as soluções obtidas poderiam ser ainda melhores.

5.1 Trabalhos Futuros

Alguns trabalhos que poderiam ser considerados para estender este ou para abranger outros ramos de desenvolvimento poderiam ser:

- Acrescentar a possibilidade de trabalhar com cromossomo de tamanho variável a libgabin.
- Adicionar representação real a libgabin.
- Explorar outras formas de codificação para um AG executar o processo de clusterização.

- Verificar a métrica de similaridade utilizada no AG e no K-Médias de forma a verificar sua equivalência.
- Testar outras funções de *fitness*.

Referências Bibliográficas

[1] AL-SULTAN, K. S.; KHAN, M. M. Computational experience on four algorithms for the hard clustering problem. **Pattern Recogn. Lett.**, New York, NY, USA, v.17, n.3, p.295–308, 1996.

[2] COLE, R. M. **Clustering with Genetic Algorithms**. Nedlands 6907, Australia, 1998. Dissertação de Mestrado.

[3] DHILLON, I. S.; MODHA, D. S. Concept decompositions for large sparse text data using clustering. **Machine Learning**, [S.l.], v.42, n.1, p.143–175, 2001.

[4] DE SOUTO, M. C. P. **Algoritmo de Agrupamento (Clustering): Métodos Hierárquivos e K-Médias.** `www.dimap.ufrn.br/marcilio/IA/IA2004.1/IA-alg-clustering.ppt`.

[5] ESTIVILL-CASTRO, V.; YANG, J. Fast and robust general purpose clustering algorithms. In: PACIFIC RIM INTERNATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE, 2000. **Proceedings...** Callaghan, Australia: [s.n.], 2000. p.208–218.

[6] EVERITT, B. **Cluster analysis**. Londres: Heinemenn Educational Books, 1977.

[7] FISHER, R. A. The use of multiple measurements in taxonomic problems. **Annals Eugen.**, [S.l.], v.7, p.179–188, 1936.

[8] GOLDBERG, D. E. **Genetic Algorithms in Search, Optimization and Machine Learning**. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989.

[9] GONZALEZ, R. C.; WOODS, R. E. **Digital Image Processing**. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2001.

[10] GORDON, A. D. **Classification**. Monographs in Applied Statistics and Probability. 2. ed. Chapman & Hall / CRC, 1999.

[11] GREFENSTETTE, J. Optimization of control parameters for genetic algorithms. **IEEE Trans. Syst. Man Cybern.**, Piscataway, NJ, USA, v.16, n.1, p.122–128, 1986.

[12] GWEE, B.-H.; CHANG, J. S. A hybrid genetic hill-climbing algorithm for four-coloring map problems. Amsterdam, The Netherlands, The Netherlands, v.104, p.252–261, 2003.

[13] JAIN, A. K.; DUBES, R. C. **Algorithms for clustering data**. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1988.

[14] Johnson, R. A.; Wichern, D. W., editors. **Applied multivariate statistical analysis**. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1988.

[15] JONG, K. A. D.; SPEARS, W. M. An analysis of the interacting roles of population size and crossover in genetic algorithms. In: Schwefel, H. P.; Männer, R., editors, PARALLEL PROBLEM SOLVING FROM NATURE - PROCEEDINGS OF 1ST WORKSHOP, PPSN 1, 1991. **Proceedings...** Dortmund, Germany: Springer-Verlag, Berlin, Germany, 1991. v.496, p.38–47.

[16] LEE, Z.-J. et al. Genetic algorithm with ant colony optimization (ga-aco) for multiple sequence alignment. **Appl. Soft Comput.**, Amsterdam, The Netherlands, The Netherlands, v.8, n.1, p.55–78, 2008.

[17] LIU, C. L. **Introduction to Combinatorial Mathematics**. McGraw-Hill, 1968.

[18] MAHFOUD, S. W.; GOLDBERG, D. E. Parallel recombinative simulated annealing: a genetic algorithm. **Parallel Computing**, [S.l.], v.21, n.1, p.1–28, 1995.

[19] MAIMON, O.; ROKACH, L. **Data Mining and Knowledge Discovery Handbook**. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2005.

[20] MITCHELL, M. **An introduction to genetic algorithms**. Cambridge, MA, USA: MIT Press, 1996.

[21] NETCRAFT. **Netcraft Web Server Survey**. `http://survey.netcraft.com/Reports/current/graphs.html`.

[22] RAY, S.; TURI, R. **Determination of number of clusters in k-means clustering and application in colour image segmentation**.

[23] SANTA CATARINA, A. **Algoritmos Evolutivos Híbridos Para Detecção de Casamento em Imagens Digitais**. Monografia. Instituto Nacional de Pesquisas Espaciais - Pós-Graduação em Computação Aplicada - 2004.

[24] SIVANANDAM, S. N.; DEEPA, S. N. **Introduction to Genetic Algorithms**. 1. ed. Berlin, Germany: Springer, 2008.

[25] Staff, C. S. I. I., editor. **SAS-STAT User's Guide: release 6.03 edition**. Cary, NC, USA: SAS Institute Inc., 1988.

[26] UNIVERSITY, N. **Roulette wheel selection**. `http://www.edc.ncl.ac.uk/highlight/rhjanuary2007g02.php/`.