UFRRJ

INSTITUTO DE CIÊNCIAS EXATAS

CURSO DE PÓS-GRADUAÇÃO EM MODELAGEM MATEMÁTICA E COMPUTACIONAL

DISSERTAÇÃO

O Uso De Algoritmos Genéticos Para A Solução De Problemas De Agrupamento Automático

Suzane Pereira Lima

2019

UNIVERSIDADE FEDERAL RURAL DO RIO DE JANEIRO
INSTITUTO DE CIÊNCIAS EXATAS
CURSO DE PÓS-GRADUAÇÃO EM MODELAGEM MATEMÁTICA E COMPUTACIONAL

O USO DE ALGORITMOS GENÉTICOS PARA A SOLUÇÃO DE PROBLEMAS DE AGRUPAMENTO AUTOMÁTICO

SUZANE PEREIRA LIMA

Sob a Orientação do Professor

Marcelo Dib Cruz

Dissertação submetida como requisito parcial para obtenção do grau de **Mestre**, no Curso de Pós-Graduação em Modelagem Matemática e Computacional, Área de Concentração em Modelagem Matemática e Computacional.

Seropédica, RJ
Julho de 2019

Universidade Federal Rural do Rio de Janeiro
Biblioteca Central / Seção de Processamento Técnico

Ficha catalográfica elaborada
com os dados fornecidos pelo(a) autor(a)

Lima, Suzane Pereira, 1993-
L732u O uso de algoritmos genéticos para a solução de problemas de agrupamento automático / Suzane Pereira Lima. - Seropédica, 2019.
105 f.: il.

Orientador: Marcelo Dib Cruz.
Dissertação(Mestrado). -- Universidade Federal Rural do Rio de Janeiro, Programa de Pós-Graduação em Modelagem Matemática e Computacional, 2019.

1. Otimização. 2. Heurísticas. 3. Problema de Agrupamento Automático. I. Cruz, Marcelo Dib, 1967-, orient. II Universidade Federal Rural do Rio de Janeiro. Programa de Pós-Graduação em Modelagem Matemática e Computacional III. Título.

UNIVERSIDADE FEDERAL RURAL DO RIO DE JANEIRO
INSTITUTO DE CIÊNCIAS EXATAS
CURSO DE PÓS-GRADUAÇÃO EM MODELAGEM MATEMÁTICA E COMPUTACIONAL

SUZANE PEREIRA LIMA

Dissertação submetida como requisito parcial para obtenção do grau de **Mestre**, no Curso de Pós-Graduação em Modelagem Matemática e Computacional, área de Concentração em Modelagem Matemática e Computacional.

DISSERTAÇÃO APROVADA EM 30 / 07 / 2019.

_____

Prof. Marcelo Dib Cruz. (Dr.) UFRRJ
(Orientador)

_____

Prof. Gizelle Kupac Vianna. (Dra.) UFRRJ

_____

Prof. Wagner de Souza Tassinari. (Dr.) FIOCRUZ

"Enquanto estiver vivo, sinta-se vivo.
Se sentir saudades do que fazia, volte a fazê-lo.
Não viva de fotografias amareladas...
Continue, quando todos esperam que desistas.
Não deixe que enferruje o ferro que existe em você.
Faça com que em vez de pena, tenham respeito por você.
Quando não conseguir correr através dos anos, trote.
Quando não conseguir trotar, caminhe.
Quando não conseguir caminhar, use uma bengala.
Mas nunca se detenha."
(Santa Teresa de Calcutá)

AGRADECIMENTOS

Antes de qualquer coisa, agradeço a Deus por permitir que eu passasse pelo mestrado e chegasse a conclusão deste trabalho, me dando forças física e mental para suportar a tudo até o final. Glórias e louvores sejam dadas a ti, Senhor!

Agradeço também a poderosa intercessão de Nossa Senhora, de meu anjo da guarda e de todos aqueles que lembraram de mim em suas orações.

A meus pais, por serem o meu suporte em meio a tantas barreiras, por fazerem parte de tudo isto e por darem todo apoio que preciso.

A meu namorado, Weslem, por estar do meu lado nestes tempos tão difíceis. Obrigada por todo carinho e por me ajudar a sorrir em meio a tudo isto!

A todos aqueles com quem pude contar além da conclusão da graduação: Alexsander, Egberto, Michel, Késia e Ygor. Obrigada por serem luz, de diferentes formas vocês fizeram a diferença.

Um obrigada a meu orientador, Marcelo Dib, pelo acompanhamento nestes anos de mestrado.

Agradeço à Rural, por mais uma vez me acolher como sua discente. Um obrigada, também, a todos do PPGMMC e colegas de classe. Em especial a Erylaine, com quem tive a oportunidade de conviver com maior frequência no primeiro ano, obrigada pela parceria de estudos!

Por fim, agradeço ao auxílio recebido durante estes anos. O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

RESUMO

Lima, Suzane Pereira. **O uso de Algoritmos Genéticos para a solução de Problemas de Agrupamento Automático**. 2019. 92 p. Dissertação (Mestrado em Ciências, Modelagem Matemática e Computacional), Instituto de Ciências Exatas, PPGMMC, Universidade Federal Rural do Rio de Janeiro, Seropédica, 2019.

Técnicas de agrupamento de dados consistem na organização de um conjunto de informações em grupos de acordo com as similaridades presentes em seus registros, assim propriedades comuns entre o conjunto de dados conseguem ser identificadas facilitando a sua compreensão. Nem sempre o número de grupos é um dado disponível a priori para a resolução desse processo. Quando esta informação é desconhecida tem-se o denominado Problema de Agrupamento Automático. Neste trabalho são apresentadas estratégias para a resolução deste problema, tendo como base a meta-heurística Algoritmo Genético. Sabendo que a qualidade de um agrupamento pode ser influenciada pelo modo como são gerados os grupos iniciais e pela escolha da função de avaliação das soluções, diferentes procedimentos são propostos a partir de um método já existente com o objetivo de gerar resoluções de melhor qualidade. Experimentos foram aplicados às propostas para diversos conjuntos de dados. Os resultados obtidos são comparados entre si e com outros trabalhos da literatura.

Palavras-chave: Problema de Agrupamento Automático. Heurísticas. Otimização.

ABSTRACT

Lima, Suzane Pereira. **O uso de Algoritmos Genéticos para a solução de Problemas de Agrupamento Automático**. 2019. 92 p. Dissertation (Master in Sciences, Mathematical and Computational Modeling), Instituto de Ciências Exatas, PPGMMC, Universidade Federal Rural do Rio de Janeiro, Seropédica, 2019.

The Clustering consists to organize a set of information into groups according to the similarities present in their records, so that common properties between the data can be identified for easy understanding. Sometimes the number of groups is a unavailable data for the resolution of this process. When this information is unknown a priori we have the so-called Automatic Clustering Problem. In this work are presented algorithms to solve this problem, based on the meta-heuristic Genetic Algorithm. The quality of a Clustering can be affected by the way the initial groups are generated and by the choice of the cluster validity index, different methods are proposed based on an already existing methodology with the purpose of generating better quality resolutions. Experiments were applied to proposed approaches for several data sets. The obtained results are compared with each other and with other works of the literature.

Keywords: Automatic Clustering Problem. Heuristics. Optimization.

LISTA DE FIGURAS

Figura 2.1: Exemplo de diferentes resultados de agrupamento para um mesmo conjunto de dados [11]. 5
Figura 2.2: Exemplo de representação de grupos por partições [34]. 8
Figura 2.3: Exemplo de representação de grupos por diagramas de Venn [34]. 8
Figura 2.4: Exemplo de representação de grupos por tabelas [34]. 8
Figura 2.5: Exemplo de representação de grupos por dendogramas [34]. 9
Figura 2.6: Interpretações das diferentes versões da métrica de Minkovsky [34]. 11
Figura 2.7: Exemplo de uma árvore de grupos no agrupamento hierárquico [45]. 15
Figura 2.8: Um exemplo do algoritmo *k-means* em execução [34]. 19
Figura 2.9: Exemplo de uma inicialização ruim no algoritmo *k-means* [34]. 20

Figura 3.1: Representação do ciclo de um Algoritmo Genético [20]. 34
Figura 3.2: O conjunto de dados A [28]. 35
Figura 3.3: Exemplo de codificação binária [28]. 36
Figura 3.4: Exemplo de codificação inteira baseada em rótulo [28]. 37
Figura 3.5: Exemplo de codificação inteira baseada em grafos [28]. 38
Figura 3.6: Exemplo de codificação real de comprimento fixo [28]. 39
Figura 3.7: Exemplo da Seleção por roleta [54]. 42
Figura 3.8: Exemplo da Seleção por torneio [20]. 43
Figura 3.9: Exemplo de aplicação do Cruzamento de um ponto [45]. 44
Figura 3.10: Exemplo de aplicação do Cruzamento de múltiplos pontos [45]. 44
Figura 3.11: Exemplo de aplicação do Cruzamento uniforme [45]. 44
Figura 3.12: Exemplo de aplicação da Mutação [45]. 45

Figura 4.1: Exemplo de uma solução representada numa cadeia de 7 posições [11]. . . . 47
Figura 4.2: Cálculo dos elementos pertencentes ao círculo de centro x_2 e raio $r = \alpha * d_{medio}$ [11]. 49
Figura 4.3: Exemplo de parte do funcionamento da busca Inversão Individual [11]. . . . 55
Figura 4.4: Exemplo de parte do funcionamento da busca Troca Entre Pares [11]. 56
Figura 4.5: Exemplo de parte do funcionamento da busca Reconexão por Caminhos [11]. 57

Figura 5.1: Os conjuntos de dados de A-sets [1]. 62
Figura 5.2: Os conjuntos de dados de Shape sets [1]. 62
Figura 5.3: Solução gerada para ruspini com o algoritmo AGBL6. 84
Figura 5.4: Solução gerada para R15 com o algoritmo AGBL6. 84
Figura 5.5: Solução gerada para 300p2c1 com o algoritmo AGBL6. 85
Figura 5.6: Solução gerada para 1000p6c com o algoritmo AGBL6. 85
Figura 5.7: Comparação entre os agrupamentos obtidos por AGBL6, AECBL1 e CLUES para a instância 300p2c1 [11]. 85

LISTA DE QUADROS

Quadro 3.1: Uma analogia entre os Algoritmos Genéticos e a Natureza [20]. 32

Quadro 5.1: Propostas apresentadas para a resolução do PCA.. 60
Quadro 5.2: Resultados obtidos com o uso de diferentes procedimentos para a geração de grupos iniciais. 66
Quadro 5.3: Resultados obtidos com o uso de diferentes funções de avaliação. 70
Quadro 5.4: Comparação dos resultados obtidos nas propostas deste trabalho, AGBL1 e AGBL6, com os algoritmos AECBL1, MRDBSCAN e ACO. 77
Quadro 5.5: Comparação dos resultados obtidos nas propostas deste trabalho, AGBL1 e AGBL6, com os métodos GADE e ACO. 80
Quadro 5.6: Comparação dos resultados obtidos nas propostas deste trabalho, AGBL1 e AGBL6, com o algoritmo AK-means.. 81

LISTA DE ABREVIATURAS E SIGLAS

AGBL	Algoritmo Genético com Busca Local
AGBL1	AGBL com GSI1 e função Índice Silhueta
AGBL2	AGBL com GSI2 e função Índice Silhueta
AGBL3	AGBL com GSI3 e função Índice Silhueta
AGBL4	AGBL com GSI4 e função Índice Silhueta
AGBL5	AGBL com GSI5 e função Índice Silhueta
AGBL6	AGBL com GSI1 e função Índice CH
AECBL1	Algoritmo Evolutivo Construtivo com Busca Local1
BEA	Bacterial Evolutionary Algorithm
C-GRASP	Continuous GRASP
CHI	Calinski–Harabasz index
CSI	CS index
DBI	Davies–Bouldin index
DI	Dunn index
GCP	Procedimento Gerar Clusters Parciais
GSI1	Procedimento Gerar Solução Inicial 1
GSI2	Procedimento Gerar Solução Inicial 2
GSI3	Procedimento Gerar Solução Inicial 3
GSI4	Procedimento Gerar Solução Inicial 4
GSI5	Procedimento Gerar Solução Inicial 5
JCPA	Procedimento Junção de Clusters Parciais Adjacentes
PC	Problema de Clusterização
PCA	Problema de Clusterização Automática
SI	Silhouette index

SUMÁRIO

1 INTRODUÇÃO 1
1.1 Motivação 1
1.2 Objetivos 3
1.3 Organização do Trabalho 3

2 O AGRUPAMENTO DE DADOS 4
2.1 Descrição do Problema 4
2.2 Termos e Notações 5
2.3 Execução de Algoritmos de Agrupamento 6
2.3.1 Agrupamento *fuzzy* 6
2.3.2 Agrupamento *hard* 6
2.4 Agrupamento x Agrupamento Automático 6
2.4.1 Problema de agrupamento 7
2.4.2 Problema de agrupamento automático 7
2.5 Representação dos Grupos 7
2.6 A Similaridade dos Dados 9
2.6.1 Funções de similaridade 10
2.7 Classificação dos Algoritmos de Agrupamento 14
2.7.1 Algoritmos hierárquicos 14
2.7.2 Algoritmos particionais 17
2.8 Homogeneidade e Separação de Grupos 20
2.8.1 Critérios para homogeneidade 20
2.8.2 Critérios para separação 21
2.9 Funções de Avaliação 22
2.10 Revisão da Literatura 24

3 HEURíSTICAS PARA A RESOLUÇÃO DE UM AGRUPAMENTO DE DADOS 29
3.1 Algoritmos Evolutivos 30
3.1.1 Algoritmo genético 31

4 OS PROCEDIMENTOS PROPOSTOS 46
4.1 Composição dos Algoritmos 46
4.1.1 Representando soluções 46
4.1.2 A geração de grupos iniciais 47
4.1.3 Algoritmo genético 51
4.1.4 Memória adaptativa 53
4.1.5 As buscas locais 54
4.2 Adotando uma Função de Avaliação 56
4.2.1 Índice silhueta 57
4.2.2 Índice CH 58

5 EXPERIMENTOS COMPUTACIONAIS 59
5.1 Materiais e Ambiente 59
5.1.1 Conjuntos de dados utilizados 60
5.2 Resultados e Discussão 65
5.2.1 Comparação entre os procedimentos para a geração de grupos iniciais 66
5.2.2 Comparação silhueta x CH 69
5.2.3 Comparação com outros trabalhos da literatura 76
5.2.4 Análise gráfica de soluções 83

6 CONSIDERAÇÕES FINAIS 86
6.1 Conclusões 86
6.2 Trabalhos Futuros 87

REFERÊNCIAS 88

1 INTRODUÇÃO

1.1 Motivação

O ato de reunir informações em conjuntos é definido como agrupamento. Esta técnica consegue distinguir propriedades comuns dentre os registros de uma base de dados, organizando-as em conjuntos visando facilitar a sua compreensão. Esse é um assunto muito abrangente e amplamente estudado, muitos trabalhos propõem algoritmos buscando a sua melhor resolução. Seu objetivo é que dados pertencentes a um mesmo grupo possuam alta similaridade entre si, já aqueles que façam parte de grupos diferentes sejam pouco semelhantes [11, 21, 29, 34, 48, 59].

Existe um variado número de casos onde é proveitoso utilizar a análise de agrupamento de dados para o estudo de um conjunto de informações em diferentes contextos e áreas de conhecimento. Utilizando tal ferramenta é possível simplificar a representação de um conjunto de dados, expressando a ideia principal de grupos de elementos. Consegue-se, também, descobrir outras características e investigar possíveis particularidades. A ampla presença do conceito de agrupamento na literatura destaca a sua importância e natureza interdisciplinar [27, 34].

O agrupamento de dados apresenta uma função importante em diversas áreas, apesar de sua grande relação com Reconhecimento de Padrões e Processamento de Imagens, por exemplo, existem numerosas pesquisas em outras ciências na literatura. Além de ser estudado na Estatística e Matemática existem aplicações específicas a campos como Bioinformática, Mineração de Dados, Categorização de Documentos, Psicologia e Geografia [11, 27, 34, 45, 48, 49].

Na área de *Marketing*, grupos diferentes podem ser encontrados nas bases de clientes. Dessa forma, o agrupamento ajuda a identificar novas informações para uso no desenvolvimento de programas de *marketing* direcionados [27, 34].

Com o agrupamento, dados reais e sintéticos de terremotos também podem ser analisados para extrair características que tornam possível a previsão de eventos que indiquem abalos sísmicos [34].

Na *World Wide Web* é possível, com o agrupamento, organizar os documentos conforme suas similaridades semânticas. Assim, tornam-se melhores os resultados gerados pelos sites de busca [11, 34].

No campo da Computação a temática do agrupamento despontou com a difusão da técnica de *data mining* (Mineração de Dados). Nesta área de mineração os problemas de agrupamento possuem o mesmo propósito que para outras áreas, com o diferencial que os conjuntos de dados trabalhados em data mining são mais numerosos e com elementos que geralmente são representados por um alto número de características [9, 11, 19, 45].

Seja nas ciências ou no mundo dos negócios, a descrição apropriada de um problema pode ser o diferencial na sua resolução. Modelos matemáticos possibilitam uma abstração do que está sendo tratado através de símbolos e expressões matemáticas. Assim, um problema consegue ser representado de forma concisa, tornando mais compreensível sua estrutura geral e possibilitando que ferramentas matemáticas e de computação sejam empregadas para análise [4, 25].

Dentro da área da Inteligência Computacional existe uma temática chamada Aprendizado de Máquina (*Machine Learning*). Seu propósito consiste na produção de sistemas com a capacidade de obter conhecimento automaticamente e, também, de técnicas computacionais relacionadas ao aprendizado. Esta a área busca produzir ferramentas computacionais referentes a aprendizado assim como sistemas que consigam obter conhecimento automaticamente [21].

A Pesquisa Operacional é uma abordagem que lida com a construção de modelos para a estruturação de processos a fim de conduzir e organizar as operações de um problema de uma determinada área. Tais modelos representam, de modo simplificado, as principais características de uma situação real. Costuma-se buscar uma melhor resolução para a questão que está sendo trabalhada, assim identificar uma solução ótima é um ponto muito relevante nesta metodologia [4, 25, 36].

Dentro da Pesquisa Operacional existe a área de Otimização. Este campo lida com conceitos, metodologias e aplicações que buscam a especificação das melhores soluções num determinado problema. Nesta área são estudadas condições de otimalidade, análise de algoritmos e experimentos computacionais. Seu intuito resume-se na descoberta de mínimos ou máximos de uma função de diferentes variáveis, onde seus valores pertencem a uma região específica de um espaço multidimensional. Para um algoritmo de otimização solucionar um problema este deve ser formulado matematicamente de modo que todos os seus aspectos relevantes sejam retratados. Devem ser especificados um conjunto de restrições a ser satisfeitas e pelo menos um objetivo a ser otimizado. O conjunto formado por todas as possíveis soluções para um problema é denominado como espaço de busca, aquelas que cumprem as restrições estabelecidas são chamadas de soluções factíveis. Assim, em meio a este espaço é definida uma função objetivo a ser otimizada de acordo com o conjunto de soluções factíveis [17, 39].

Diferentes algoritmos podem ser empregados para alcançar uma solução ótima em modelos de Pesquisa Operacional. Porém, algumas vezes tal solução não é viável de ser obtida devido às limitações de algumas estratégias e às particularidades dos problemas reais, fazendo com que a resolução destes seja de elevada complexidade. Assim, para que seja possível a obtenção de soluções aceitáveis torna-se necessária a utilização de métodos heurísticos. Estes métodos conseguem alcançar resultados próximos do ótimo num tempo de processamento considerado admissível e são aplicados a problemas específicos. Existem heurísticas que são genéricas e conseguem ser aplicadas a diversos problemas, estas são denominadas como meta-heurísticas. Os problemas de agrupamento frequentemente fazem uso destas técnicas para a obtenção de seus resultados. Dentre

várias meta-heurísticas existentes podem ser mencionados os Algoritmos Genéticos [7, 25, 46].

Na resolução de um agrupamento de dados existem dois modos distintos de se tratar o problema. Quando seu número de grupos é uma informação conhecida a priori tem-se um agrupamento clássico, quando esta quantidade é estimada no decorrer da busca pela solução do problema a metodologia é denominada como agrupamento automático [7, 46, 51].

Existe na literatura o algoritmo AECBL1, que aborda o Problema de Agrupamento Automático através de um Algoritmo Genético [11]. Porém algumas questões interferem diretamente na qualidade da solução de agrupamento e o presente trabalho tem como objetivo a melhoria de dois pontos: o procedimento inicial e a função objetivo adotada.

1.2 Objetivos

Este trabalho tem como objetivo aprimorar o algoritmo AECBL1 através de novas abordagens em diferentes partes do algoritmo, ou seja, por meio de modificações na parte inicial do método e com a inclusão da função de avaliação Índice CH [11].

1.3 Organização do Trabalho

Este trabalho está estruturado da seguinte forma: o Capítulo 1 apresenta a motivação para o agrupamento de informações, sua aplicabilidade em diferentes áreas e onde se enquadra a noção de tal abordagem. O segundo capítulo apresenta o agrupamento de dados, com suas principais classificações e características. O terceiro capítulo fala sobre o uso de heurísticas para a resolução de agrupamentos de dados, dando enfoque àquela utilizada como base para o trabalho: Algoritmo Genético. A metodologia dos experimentos deste trabalho é apresentada no Capítulo 4. O Capítulo 5 apresenta, compara e analisa os resultados obtidos pelos experimentos realizados. Por fim, o Capítulo 6 apresenta as conclusões finais sobre o trabalho como um todo.

2 O AGRUPAMENTO DE DADOS

2.1 Descrição do Problema

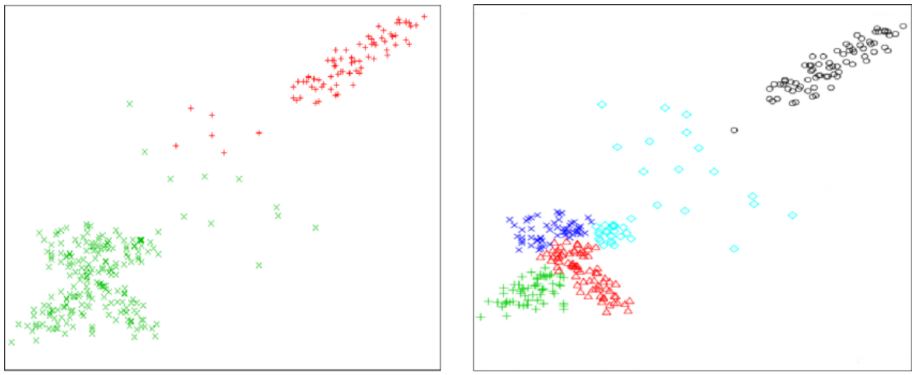
A organização de um conjunto de dados em grupos é conhecida como agrupamento de dados (*data clustering*), sendo também chamada de clusterização ou análise de grupos (*cluster analysis*). O agrupamento é um problema que lida com dados não rotulados, classificando-os em grupos definidos pelas semelhanças entre seus elementos. Sua maior dificuldade é o fato de ser uma técnica de aprendizagem não supervisionada, na maioria dos casos as características estruturais do problema são desconhecidas. No agrupamento não ocorre uma metodologia de separação por categorias, de outro modo tem-se o objetivo de organizar um conjunto de informações de forma válida e conveniente [11, 19, 26, 28, 34, 49].

No agrupamento as informações de um conjunto de dados conseguem ser representadas num modelo de forma mais simplificada. Cada membro do conjunto corresponde a uma entrada de dados, sendo também conhecido como elemento ou objeto. Tal elemento é formado por determinadas características que comumente são representadas através de um vetor de atributos (ou coordenadas). Estas podem corresponder a valores numéricos ou categóricos [9, 34, 40, 45].

O processo da formação de grupos é feito através de uma análise das características dos elementos, utilizando uma determinada medida. Assim, elementos similares entre si são alocados num mesmo grupo (*cluster*). Desse modo os grupos vão sendo formados e, cada um terá uma semelhança interna entre seus membros. Assim estes também terão uma certa diferença em relação a qualquer elemento de algum outro grupo [11, 19, 34, 45, 49].

Durante o processo de divisão dos elementos em grupos podem ser encontrados dados de entrada que não possuem semelhança alguma com nenhum dos grupos formados. Estes elementos são chamados de *outliers* e algumas de suas possíveis origens são: falhas de digitação, erros na coleta de dados e até mesmo fraudes. Lidar com tais elementos é um obstáculo adicional na busca por soluções de boa qualidade [11].

Diferentes algoritmos de agrupamento podem gerar resultados distintos para um mesmo conjunto de dados. Na Figura 2.1 é apresentado um exemplo com as resoluções obtidas por duas metodologias da literatura. Para um total de 300 elementos a estratégia o AECBL1 gerou dois grupos, enquanto o procedimento CLUES representou os dados através de cinco grupos. No desenvolvimento de técnicas de agrupamento sabe-se que são diversas as configurações de grupos que podem ser obtidas porém seu objetivo está na busca por representações estruturadas da forma mais adequada possível. Apesar da geração de uma quantidade maior de grupos tornar mais provável o aumento na semelhança interna destes uma boa separação entre grupos também precisa ser considerada ao mensurar a qualidade de um agrupamento [11, 34].



AECBL1 CLUES

Figura 2.1: Exemplo de diferentes resultados de agrupamento para um mesmo conjunto de dados [11].

2.2 Termos e Notações

A seguir são apresentadas algumas das notações usuais ao lidar com um agrupamento de dados [6, 11, 12, 28, 42].

- Conjunto de dados (ou instância): $X = \{x_1, \ldots, x_n\}$
- Tamanho do conjunto de dados: n
- Elemento i: $x_i = \{x_{i1}, \ldots, x_{im}\}$
- Número de atributos de um elemento: m
- Agrupamento: $C = \{c_1, \ldots, c_k\}$
- Quantidade de grupos do agrupamento: k
- Grupo i: c_i
- Quantidade de elementos no grupo i: $|c_i|$
- Centroide (ou referencial) do grupo i:

$$\overline{c}_i = \frac{1}{|c_i|} \sum_{i=1}^{|c_i|} x_i \tag{2.1}$$

- Centroide global do conjunto de dados:

$$\overline{X} = \frac{1}{n} \sum_{i=1}^{n} x_i \tag{2.2}$$

2.3 Execução de Algoritmos de Agrupamento

Para resolver um problema de agrupamento de dados primeiro realiza-se uma amostragem do conjunto de dados. Em seguida, são registradas as características de cada objeto deste conjunto, gerando uma matriz de dados. A partir desta matriz são calculadas as semelhanças entre os elementos, assim é definida uma matriz de similaridades [11].

De modo conjunto, o tipo de problema a ser trabalhado é indicado através da definição de restrições. Outros critérios também são estabelecidos para indicar a homogeneidade e/ou separação dos grupos [11].

Um algoritmo deve ser adotado para solucionar o problema, através de uma codificação ele é aplicado na matriz de similaridades para obter os grupos de elementos semelhantes. A partir dos grupos gerados os resultados podem ser, enfim, interpretados [11].

A execução dos algoritmos de agrupamento pode ser feita em duas formas distintas: *hard* e *fuzzy*. Estas duas serão brevemente apresentadas a seguir, porém todos os conceitos referidos neste trabalho consideram apenas a metodologia *hard* [3, 12, 28].

2.3.1 Agrupamento *fuzzy*

Nesta metodologia o agrupamento é realizado a partir do conceito de conjuntos *fuzzy*, assim cada elemento do conjunto de dados é associado a um ou mais grupos, simultaneamente, com um certo grau de associação. As relações são definidas como não-binárias de modo que, um elemento i possui um respectivo valor de pertinência $u_j \in [0, 1]$ relacionado a um grupo j [3, 12, 19, 28].

2.3.2 Agrupamento *hard*

O modo *hard* presume que a associação entre elementos e grupos é binária, logo cada elemento é associado exclusivamente a um único grupo do conjunto de dados [3, 12, 19, 28].

2.4 Agrupamento x Agrupamento Automático

Num processo de agrupamento, o número de grupos pode ser ou não conhecido na entrada do problema. Em vista disto, existem duas formas de lidar com o agrupamento de elementos. Quando o valor de k é informado tem-se o nomeado Problema de Agrupamento (*Clustering Problem*), e quando a sua descoberta faz parte do processamento da solução a metodologia é chamada

de Problema de Agrupamento Automático (*Automatic Clustering Problem*) [11, 45].

2.4.1 Problema de agrupamento

Sejam X uma base de dados e k o seu número de grupos já definido inicialmente. Nesta classe de problemas, os elementos pertencentes a esta base de dados são dispostos em exatos k grupos, de acordo com as semelhanças entre si. Logo, a partir da quantidade de grupos já conhecida a priori tem-se como resultado o agrupamento. Este é tipo de agrupamento também é denominado como Problema de K-Clusterização ou, apenas, Problema de Clusterização (PC) [11, 45].

Na literatura, grande parte dos estudos que lida com agrupamento de dados tem o número de grupos como um dado de entrada. Assim, existem diversos algoritmos para se resolver este problema, o mais conhecido chama-se k-*means* [11].

2.4.2 Problema de agrupamento automático

A partir de um conjunto de dados X, os objetos a ele pertencentes vão sendo reunidos de acordo com algum critério de semelhança. No decorrer do processamento os grupos vão sendo formados e, ao final, tem-se como resultado o agrupamento bem como o número de grupos gerados. Esta classe de agrupamento também é conhecida como Problema de Clusterização Automática (PCA) [11, 45].

Na categoria de agrupamento automático existem duas questões que são cruciais ao solucionar um problema: a definição do número ideal de grupos e a identificação correta deste conjunto de grupos. Em razão do número de grupos ser inicialmente desconhecido, este modo de agrupamento é mais complexo do que o PC pois a quantidade de possíveis configurações para solucionar o problema aumenta consideravelmente. Comparado ao Problema de Agrupamento, o número de trabalhos que lidam com o PCA é bem menor [3, 11, 28, 58].

2.5 Representação dos Grupos

Existem diferentes formas de se representar os grupos formados pelas técnicas de agrupamento. A seguir serão apresentadas as maneiras mais habituais [34].

- **Partições:** A ilustração de soluções por partições consegue ser genérica, ou seja, este formato não está associado a métodos de agrupamento específicos. A Figura 2.2 ilustra seu modo de representação [34].

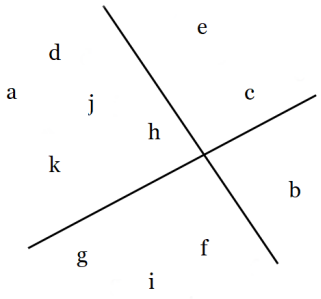


Figura 2.2: Exemplo de representação de grupos por partições [34].

- **Diagramas de Venn:** Assim como a forma particional esta representação não é relacionada a uma forma de agrupamento em particular. Sua representação é ilustrada na Figura 2.3 [34].

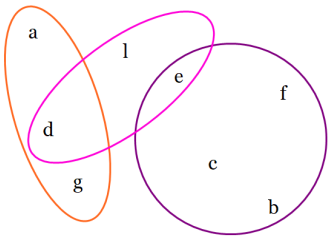


Figura 2.3: Exemplo de representação de grupos por diagramas de Venn [34].

- **Tabelas:** Este modo de representação é muito útil quando se utiliza lógica nebulosa porém também pode ser utilizado em outras situações. Sua ilustração é apresentada na Figura 2.4 [34].

	1	**2**	**3**	**4**
a	0.2	0.2	0.3	0.3
b	0.4	0.5	0.1	0.0
c	0.2	0.3	0.1	0.4

Figura 2.4: Exemplo de representação de grupos por tabelas [34].

- **Dendogramas:** Em geral, esta representação costuma ilustrar as soluções de algoritmos hierárquicos. Com seu uso estes algoritmos conseguem expôr a ordem em que o agrupamento foi realizado. A Figura 2.5 apresenta a sua forma de representação [34].

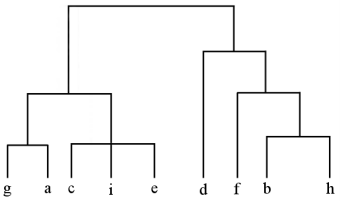


Figura 2.5: Exemplo de representação de grupos por dendogramas [34].

2.6 A Similaridade dos Dados

Os algoritmos de agrupamento geram os grupos de uma solução de acordo com uma relação de proximidade entre elementos, assim a verificação de semelhanças entre os dados é feita por meio de uma função. Diferentes medidas de similaridade podem gerar diferentes resultados de agrupamento [19, 28, 34, 45].

Apresentando um par de elementos do conjunto de dados é informado o quão parecidos estes dois são. Deste modo é possível avaliar se um mesmo grupo deve conter ambos os objetos. A interpretação do grau de semelhança depende de qual modelo de função está sendo utilizada, de similaridade ou dissimilaridade, mas o conceito de ambos é semelhante. Enquanto no primeiro caso maiores valores alcançados pela medida indicam uma maior semelhança entre dois elementos, medindo por dissimilaridade quão maior a medida maior é a diferença entre ambos. O objetivo é, com o uso dessa função, alcançar um alto grau de homogeneidade entre os elementos de cada grupo e, também, uma elevada heterogeneidade entre elementos de grupos diferentes [19, 28, 34, 45].

Todo algoritmo de agrupamento precisa ter como base alguma função verificadora de semelhanças/diferenças, caso contrário não haveria como realizar uma análise de grupos com significado. A medida aplicada deve ser específica ao problema tratado. Esta não precisa necessariamente ser representada de modo numérico por distância, porém esta forma costuma ser muito utilizada para identificar o grau de semelhança entre elementos. Trabalhando com os atributos dos elementos é possível perceber que a similaridade entre dois elementos é maior a medida que a distância entre ambos é menor. Porém em algumas situações não é possível, ou adequado, aplicar uma distância como medida. Existem problemas onde nem todos os atributos são valores escalares. Por exemplo, a condição de aprovação num concurso ou o endereço de um cliente precisariam de outras medidas que demonstrassem o grau de semelhança entre os dados [11, 19, 45].

Como sabemos, cada elemento do conjunto é representado por um vetor. Então, inicialmente, os dados são representados pelo que pode ser chamado de matriz de dados. Nesta matriz Mn_xm as linhas correspondem aos n elementos do conjunto. Já as colunas de M são os atributos relacionados às m características desses elementos. Então em M representam-se n elementos, ou

vetores, cada qual com uma dimensão de m coordenadas [11, 26, 34].

$$M = \begin{bmatrix} m_{11} & m_{12} & \dots & m_{1m} \\ m_{21} & m_{22} & \dots & m_{2m} \\ \vdots & \dots & \dots & \vdots \\ m_{n1} & m_{n2} & \cdots & m_{nm} \end{bmatrix} \tag{2.3}$$

Os relacionamentos dentro do conjunto podem ser descritos através de uma matriz de distâncias Dn_xn. Esta matriz é construída a partir da matriz de dados. Cada posição d_{ij} contém um valor numérico referente a proximidade entre dois elementos i e j, sendo este valor medido por uma função de similaridade/dissimilaridade. Considerando que a proximidade entre dois elementos independe da ordem em que se analisa a matriz D representada é simétrica [11, 26, 34].

$$D = \begin{bmatrix} 0 & & & & \\ d_{21} & 0 & & & \\ d_{31} & d_{32} & 0 & & \\ \vdots & \vdots & \vdots & \ddots & \\ d_{n1} & d_{n2} & \cdots & \cdots & 0 \end{bmatrix} \tag{2.4}$$

2.6.1 Funções de similaridade

As funções de similaridade (ou dissimilaridade) possuem as seguintes características [11, 34]:

$$d_{ij} \geq 0, \forall i, j \in X; \tag{2.5}$$

$$d_{ij} = d_{ji}, \forall i, j \in X; \tag{2.6}$$

$$d_{ij} + d_{jk} \geq d_{ik}, \forall i, j, k \in X. \tag{2.7}$$

Denomina-se como norma toda métrica que atende ao critério abaixo além dos citados anteriormente [34].

$$d(ax, ay) = |a|d(x, y) \tag{2.8}$$

As principais medidas de semelhança utilizadas nos algoritmos de agrupamento serão apresentadas a seguir.

2.6.1.1 Métrica de Minkowski

$$dist(x, y) = d_p(x, y) = \sqrt[p]{\sum_{i=1}^{m} |x_i - y_i|^p} \tag{2.9}$$

Na fórmula acima x e y são elementos do conjunto de dados, m é a dimensão destes elementos e p é um parâmetro determinado inicialmente. Conforme o valor de p aumenta, a métrica se torna mais sensível a distâncias maiores. Portanto, a decisão de qual valor definir para p está ligada ao quanto espera-se realçar as distâncias maiores. A medida atende a todas as propriedades já citadas, incluindo aquela que define uma norma, deste modo também é conhecida como norma L_p [12, 26, 27, 28, 34].

Esta medida possui alguns casos especiais de acordo com o valor de p. A Figura 2.6 ilustra a diferença de interpretação entre algumas versões desta métrica para elementos de um espaço bidimensional. A linha fina equivale ao cálculo da Distância Euclidiana, já a linha com maior espessura faz referência a Distância de Manhattan e a linha tracejada apresenta a Distância de Chebyshev [34].

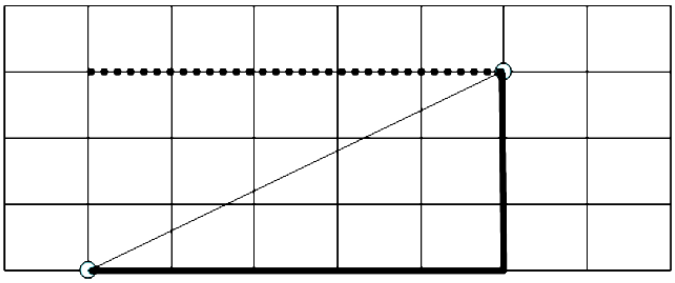


Figura 2.6: Interpretações das diferentes versões da métrica de Minkovsky [34].

Desvantagens do seu uso:

- Seu uso pode ser menos eficiente ao lidar com um alto número de características [12].

2.6.1.1.1 Distância Euclidiana

$$dist(x, y) = d_e(x, y) = \sqrt{\sum_{i=1}^{m} |x_i - y_i|^2} \tag{2.10}$$

Este é um caso específico da Métrica de Minkowski, no qual o valor de p é igual a 2. Também é denominada por norma L_2 [12, 26, 27, 28, 34].

2.6.1.1.2 Distância de Manhattan

$$dist(x, y) = \sum_{i=1}^{m} |x_i - y_i| \tag{2.11}$$

A métrica de Minkowski apresenta esta nomenclatura quando é atribuído o valor 1 a p [12, 26, 28, 34].

2.6.1.1.3 Distância de Chebyshev

$$dist(x, y) = d_\infty(x, y) = max\{|x_i - y_i|\}, onde\ x \leq i \leq m \tag{2.12}$$

Esta situação ocorre quando na métrica de Minkowski o valor de p tende a infinito. Esta medida também é chamada como norma L_∞ [26, 28, 34].

2.6.1.2 Métrica de Camberra

$$dist(x, y) = \sum_{i=1}^{m} \frac{|x_i - y_i|}{|x_i| + |y_i|} \tag{2.13}$$

Para cada par de elementos, esta métrica calcula a soma das diferenças fracionárias de seus atributos. Como visto anteriormente, cada elemento possui m atributos. Logo, o resultado final da métrica, isto é o somatório total, terá um valor dentro do intervalo $[0, m]$ [34]. Sobre os atributos sabe-se que:

- Quando o atributo k de um dos elementos é igual a 0, o k-ésimo termo terá valor 1 [34].
- Se ambos elementos possuem o atributo k com valor 0, o valor resultaria em $0/0$. Então é definido que o valor do k-ésimo termo será igual a 0 [34].
- Em outras situações o valor do termo estará dentro do intervalo $[0, 1]$ [34].

Desvantagens do seu uso:

- Quando ambos os atributos estão próximos de zero o cálculo da distância possui maior sensibilidade em meio a pequenas variações [34].
- Diferentes pares de atributos com mesma distância apresentando módulos distintos possuem diferentes influências no resultado final [34].

2.6.1.3 Métrica de Mahalanobis

Considerando Σ como a matriz de covariância[1] dos vetores que representam o par de elementos em análise, temos [27, 34, 37]:

$$dist(x, y) = \sqrt{(x - y)^T \Sigma^{-1} (x - y)} \tag{2.14}$$

Esta métrica assemelha-se a Métrica de Minkowski no caso em que p é igual a 2, ou seja, a Distância Euclidiana. Sua diferença está no fato de ser observada a correlação[2] entre os conjuntos de dados. Logo, ao utilizar esta distância, algumas limitações encontradas no uso Distância Euclidiana são corrigidas. Ela considera automaticamente a proporção dos eixos coordenados e a correlação entre as características [34].

Quando a matriz de covariâncias é a matriz identidade temos o cálculo da Distância Euclidiana. Se a matriz de covariâncias é uma matriz diagonal temos o cálculo da Distância Euclidiana normalizada, como descrito abaixo. Então, considerando m como a dimensão dos vetores e σ_i como o desvio padrão[3] relacionado ao atributo i temos [34, 37]:

$$dist(x, y) = \sqrt{\sum_{i=1}^{m} \frac{(x_i - y_i)^2}{\sigma_i^2}} \tag{2.15}$$

A distância de Mahalanobis costuma ser utilizada para a detecção de *outliers*. Com esta métrica pode-se calcular a distância entre um elemento x e um grupo qualquer. Considerando $\bar{x}_r = \{x_1, \ldots, x_p\}$ como uma média dos elementos do grupo r e Σ como a sua matriz de covariância, temos que a distância é dada por [17, 34]:

[1] É uma matriz quadrada e simétrica a qual possibilita verificar se existem dependências entre dois atributos distintos através de medidas de associação. Nela estão envolvidas medidas estatísticas de modo que, na diagonal principal da matriz estão as variâncias e fora desta diagonal se encontram as covariâncias entre os elementos i e j [17].

[2] É um coeficiente que permite verificar se existe uma relação entre atributos [17].

[3] Relacionado ao desvio de um conjunto de valores em torno de um valor médio. É definido como [5]: $\sigma_i = \sqrt{\frac{\sum_{j=1}^{n} (x_j - \bar{x}\ )^2}{n}}$

$$dist(x, c_r) = \sqrt{(x - \bar{x}_r\ )^T \Sigma^{-1} (x - \bar{x}_r\ )} \tag{2.16}$$

Dessa forma, seria como avaliar se um elemento pertence a um grupo não apenas no que se refere a distância ao centro dele, mas também considerando o desvio padrão do mesmo. Dessa forma, quanto maior é o valor encontrado, menor é a chance desse elemento pertencer ao grupo pois haverá um maior número de desvios padrões donde o elemento está distante do centro do grupo [34].

Desvantagens do seu uso:

- Determinar as matrizes de covariância pode ser complicado [34].
- De acordo com o número de características, a memória e o tempo de computação crescem de forma quadrática [17, 34].

2.7 Classificação dos Algoritmos de Agrupamento

Os algoritmos de agrupamento identificam diversos tipos de grupos numa base de dados, existe mais de um modo de retratar o sentido dos grupos gerados. Assim, o tipo de problema que será trabalhado precisa ser especificado. Estas classificações não fazem diferenciação entre o Problema de Agrupamento e o Problema de Agrupamento Automático. As mais populares e estudadas são duas: hierárquica e particional [11, 28, 48, 59].

2.7.1 Algoritmos hierárquicos

Nestes métodos, uma hierarquia de relacionamentos entre os elementos é formada. Uma disposição dos grupos é construída para a organização dos dados, de modo que em cada nível há um agrupamento distinto que tem como base a solução encontrada no nível predecessor. Além de se obter os grupos também consegue-se toda a estrutura de dados relacionada às informações de entrada. Desta forma, torna-se possível o acesso a subconjuntos destes elementos. Através de representações por dendogramas, por exemplo, esses algoritmos deixam clara como ocorre a ligação entre os dados, assim como a semelhança entre dois elementos quaisquer. Através desta forma de ilustrar a hierarquia pode-se perceber claramente como o agrupamento foi ordenado. Atualmente, diminuiu-se o uso desses métodos. Outros procedimentos possuem maior aceitação especialmente pelo custo no consumo de memória [9, 26, 28, 34, 48].

Considere um conjunto de partições $H = \{P_1, P_2, \ldots, P_t\}$ de X, com $t \leq n$. As partições presentes numa hierarquia são definidas de modo que, se $c_r \in P_k$, $c_s \in P_l$, com $k > l$, então

$c_r \subset c_s$ ou $c_r \cap c_s = \emptyset$ para $r, s = 1, \ldots, t$ e $r \neq s$ [11].

Nos procedimentos hierárquicos tradicionais os grupos são formados gradualmente, seja por aglomerações ou divisões dos elementos. Como consequência, é gerada uma hierarquia de grupos. Cada qual que possui mais de um elemento já pode ser tido como formado por grupos menores. O exemplo da Figura 2.7 apresenta como normalmente é a sua representação, através de uma estrutura de árvore. Duas das classificações dos algoritmos hierárquicos são denominadas como aglomerativa e divisiva [11, 28, 34, 45].

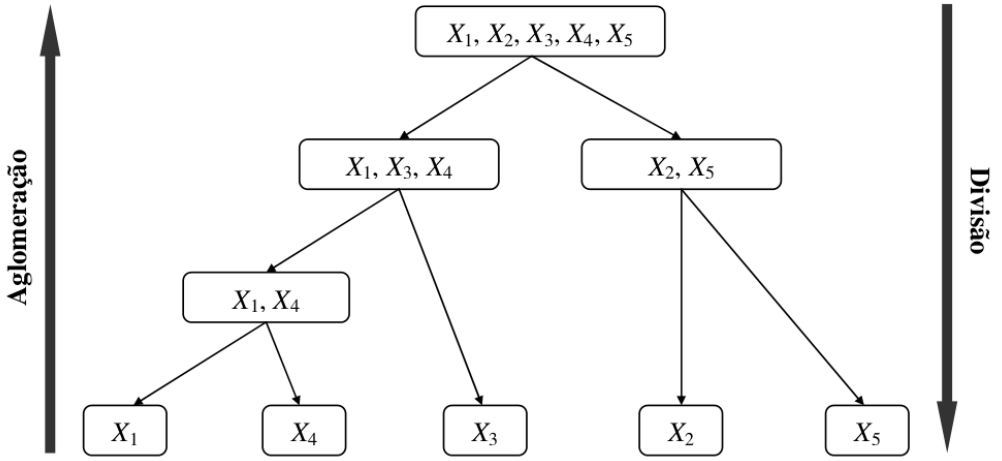


Figura 2.7: Exemplo de uma árvore de grupos no agrupamento hierárquico [45].

Nos algoritmos hierárquicos, o uso de qualquer medida de similaridade é uma tarefa simples, sendo possível aplicá-la para todo tipo de atributo, seja numérico ou categórico. Porém por estes serem algoritmos construtivos a sua maioria não visita novamente os grupos gerados no decorrer do processamento, não sendo possível, assim, aperfeiçoar as soluções. Além disto, no uso destes algoritmos tem-se a falta da obtenção de grupos específicos, já que a sua resolução apresenta somente uma hierarquia de relacionamentos. Porém, é possível definir k grupos cortando as $k - 1$ arestas mais elevadas no dendograma. Uma outra desvantagem se deve a imprecisão de seu critério de parada. A execução pode ser finalizada quando o número de grupos é alcançado, quando o problema trabalhado é do tipo K-Clusterização, ou quando algum outro critério de parada é satisfeito. A metodologia hierárquica tradicional possui características gulosas devido a ausência de melhoria durante o processo de união ou separação de dados [3, 28, 34, 45].

2.7.1.1 Algoritmo Hierárquico Aglomerativo

Os conjuntos são formados a partir dos elementos ainda isolados, sendo utilizada uma abordagem conhecida como *bottom-up*. Desta forma, o algoritmo inicia com cada elemento compondo um grupo unitário, e novos grupos são formados gradualmente a partir da junção dos já que já existem. A união dos grupos é feita segundo algum critério que mensure a proximidade dos elementos. O Algoritmo 1 apresenta o seu pseudocódigo [3, 11, 34, 45, 50, 57].

Algoritmo 1: Algoritmo hierárquico aglomerativo

```
Entrada: X
início
    Definir cada elemento do conjunto X como um grupo;
    Calcular a distância entre todos os pares de grupos;
    enquanto critério de parada não é satisfeito faça
        Unir o par de grupos mais similar em um grupo maior G;
        Calcular a distância entre o novo grupo G e os outros grupos;
    fim
    Informar a hierarquia de grupos.
fim
```

Com o uso de representações por dendogramas observa-se que, quanto mais alta for a linha de ligação de dois grupos mais tardio o agrupamento destes. Então a distância entre esses grupos é proporcional a altura da linha que faz a ligação entre eles [34].

2.7.1.2 Algoritmo Hierárquico Divisivo

O procedimento inicia com apenas um grupo, contendo todos os elementos do conjunto de dados, e separa-o em porções até alcançar elementos isolados. O agrupamento hierárquico divisivo adota uma abordagem conhecida como *top-down*, ou seja, a cada iteração novas divisões são feitas para formar grupos com tamanho ainda mais reduzido. Cada grupo que é dividido pode gerar dois ou mais grupos, esta quantidade gerada pode ser definida através de uma métrica, por exemplo. Assim, o procedimento segue recursivamente até que algum critério de parada, definido inicialmente, seja satisfeito ou que cada conjunto se torne unitário. Seu pseudocódigo é apresentado no Algoritmo 2 [11, 34, 45, 57, 58].

Algoritmo 2: Algoritmo hierárquico divisivo

Entrada: X

início
  Definir um único grupo com todos os elementos do conjunto X;
  Calcular a distância entre os elementos;
  enquanto *critério de parada não é satisfeito* **faça**
    Dividir a partição corrente em outra com grupos menores, com a maior qualidade possível, de acordo com as dissimilaridades;
    Calcular a distância entre todos os pares de grupos;
  fim
  Informar a hierarquia de grupos.
fim

O alto número de particionamentos de cada iteração faz com que este método seja muito custoso computacionalmente. Desse modo, o tempo de execução para um conjunto de dados com muitos elementos seria extremamente ruim. Além disto, a versão aglomerativa dos algoritmos hierárquicos é mais simples do que a divisiva já que esta última depende de um outro algoritmo, não-hierárquico, em seu processamento [34].

Apesar das desvantagens citadas, nos algoritmos divisivos os resultados de agrupamento tendem a ser mais próximos ao que é correto sobre as informações representadas nos dados. Isto ocorre porque o algoritmo inicia considerando o conhecimento sobre o todo, diferente da versão aglomerativa onde as decisões se baseiam em informações locais. Uma outra vantagem é que a versão divisiva não precisa dos cálculos de todas as distâncias entre os grupos a cada iteração, a execução pode ser finalizada antes de se chegar ao último nível da árvore [34].

2.7.2 Algoritmos particionais

Nesta forma de agrupamento, os dados são repartidos em k grupos disjuntos e não vazios, o valor de k pode ser ou não conhecido. Numa solução, cada elemento pertence a um único grupo e todo grupo possui no mínimo um elemento. Diferentemente da metodologia anterior que gera uma hierarquia com diferentes agrupamentos em cada nível, o algoritmo particional gera uma única divisão dos dados em grupos menores, formando subconjuntos do conjunto de dados original [11, 27, 45, 48, 59].

Grande parte das propostas na literatura tem o propósito de encontrar partições e presume que o número de grupos é um dado já conhecido previamente. Porém a maior parte dos problemas que precisam ser solucionados pelo conceito de agrupamento não conhece este valor no início [11, 45, 59].

No modo particional, cada configuração passa por uma avaliação até que seja encontrado

o agrupamento ideal. Se a verificação mostrar que a configuração dos elementos não satisfaz ao problema em estudo os elementos são realocados entre os grupos e, iterativamente, o processamento é feito até que se cumpra algum critério de parada. A ação de transferência dos elementos é denominada como otimização iterativa. Dessa forma, diferente dos algoritmos hierárquicos, a composição dos grupos pode ser aperfeiçoada no decorrer da execução [6, 11, 23, 45].

Seja um conjunto $X = \{x_1, x_2, \ldots, x_n\}$ de n pontos no $\mathbb{R}^m$. Esta forma de agrupamento tem como objetivo encontrar uma partição específica $C = \{c_1, \ldots, c_k\}$ de X com k grupos não sobrepostos, com $2 \leq k \leq n-1$, onde k pode ser, ou não, conhecido a priori. Cada grupo $c_r \subset C$ é um subconjunto de elementos de X e C é o conjunto de todos os grupos gerados nesta solução. Neste procedimento deve-se cumprir as condições definidas abaixo [3, 11, 12, 28, 45]:

$$\cup_{r=1}^{k} c_r = X \tag{2.17}$$

$$c_r \neq \emptyset, para\ 1 \leq r \leq k \tag{2.18}$$

$$c_r \cap c_s = \emptyset, para\ 1 \leq r \leq k,\ 1 \leq s \leq k,\ r \neq s \tag{2.19}$$

2.7.2.1 K-means

A ideia de agrupamento particional surgiu com a publicação do algoritmo *k-means*, em 1957. Esta é uma técnica que busca solucionar o Problema de Agrupamento, sendo um dos algoritmos mais aplicados na literatura. A partir dele, os dados são arrumados de forma simples e rápida num conjunto exato de k grupos, onde k é definido inicialmente [29, 44, 48, 49].

Este método clássico realiza um agrupamento particional, minimizando, de modo iterativo, a distância de cada um dos elementos a um dos k grupos. Cada grupo possui uma figura representativa definida como centroide, este equivale a um valor médio das características relacionadas aos membros do grupo. São definidos como $\overline{c} = \{\overline{c}_1, \ldots, \overline{c}_k\}$ os centroides de cada grupo do conjunto. A cada iteração estes centroides são recalculados e o agrupamento é avaliado [11, 34, 45].

Define-se como $dist(x_i, \overline{c})$ a menor distância entre um elemento x_i e um dos centros do conjunto de grupos, ou seja, é a distância entre este ponto e o centroide mais próximo. A função definida abaixo representa a distância entre os elementos e um conjunto de k centroides, ela deve ser minimizada [34]:

$$dist(X, \bar{c}\,) = \frac{\sum_{i=1}^{n} (dist(x_i, \bar{c}\,))^2}{n} \tag{2.20}$$

No K-Means, o usuário determina inicialmente um parâmetro referente a quantidade de

grupos. Sua execução é ágil e, frequentemente, a convergência do algoritmo para uma configuração estável demora apenas algumas poucas iterações. Em tal configuração, cada elemento pertence ao grupo com o centroide mais próximo [11, 34].

Na Figura 2.8 pode-se observar um exemplo de sua execução. Na ilustração (a), os centroides foram calculados e são representados pelos círculos maiores, então cada um dos elementos foi alocado aleatoriamente num dos três grupos. Em (b) os elementos foram realocados para os grupos com maior proximidade. A ilustração (c) apresenta os centroides recalculados e a formação final dos grupos, caso esta ainda não fosse a formação final o algoritmo continuaria a sua execução. O Algoritmo 3 apresenta seu pseudocódigo [19, 27, 28, 34].

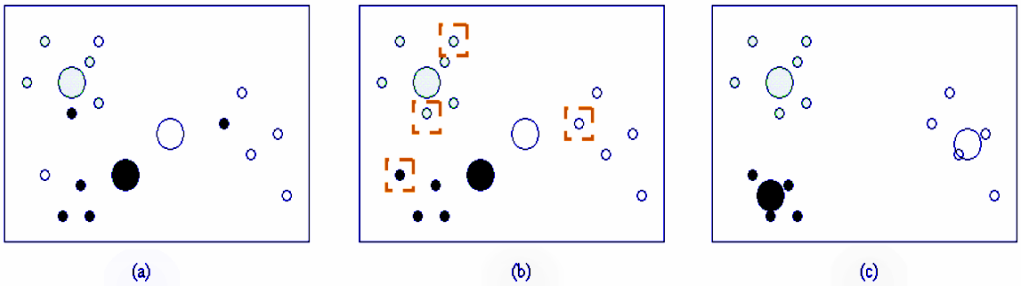


Figura 2.8: Um exemplo do algoritmo *k-means* em execução [34].

Algoritmo 3: Algoritmo k-means

Entrada: X, k

```
1 início
2     Definir um centroide para cada um dos k grupos;
3     enquanto elementos estiverem sendo realocados faça
4         Associar cada elemento de X ao centroide mais próximo;
5         Recalcular o centroide de cada grupo;
6     fim
7     Informar o particionamento gerado.
8 fim
```

Uma fragilidade do algoritmo se deve a que, muitas vezes a sua fase de inicialização conduz as soluções para mínimos locais. Em razão da ênfase acerca da homogeneidade, acaba sendo ignorada a questão de uma boa separação dos grupos. Desse modo, com uma separação dos grupos de baixa qualidade, seus centroides não são bem inicializados. A Figura 2.9 ilustra a consequência de uma inicialização ruim na execução do algoritmo. Pode ser observado que existem três grupos naturais no conjunto, porém uma inicialização ruim dos grupos levou a definição de dois deles como um único grupo representado na cor preta e o outro restante foi representado repartido em outros dois [27, 34, 48, 58].

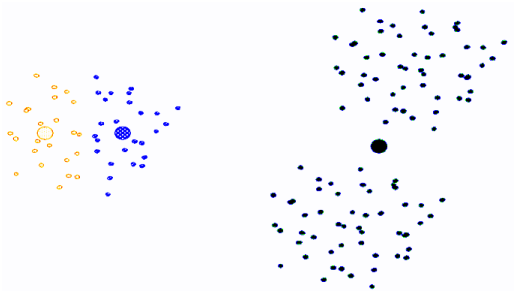


Figura 2.9: Exemplo de uma inicialização ruim no algoritmo *k-means* [34].

Uma outra questão é que, muitas vezes o número de grupos não é conhecido a priori, logo a definição dessa quantidade pode ser um problema. Assim, a escolha de poucos grupos pode resultar na união de dois grupos naturais e um grande número de grupos pode fazer com que um grupo natural seja dividido, de forma artificial, em dois [34].

2.8 Homogeneidade e Separação de Grupos

Ao obter uma solução espera-se que, em sua configuração, os grupos gerados tenham a menor semelhança possível entre si. Da mesma forma, busca-se que cada um destes possua uma alta homogeneidade interna. Existem critérios que permitem avaliar se os grupos formados são de boa qualidade ou não. Estas verificações medem o quão homogênea é a composição interna de um grupo e se a separação entre os grupos do conjunto é bem definida [11, 35].

2.8.1 Critérios para homogeneidade

Alguns dos critérios que avaliam a homogeneidade de um grupo serão apresentados abaixo.

- **Diâmetro (*Diameter*):** Representado para um grupo c_r como $d(c_r)$. Consiste na semelhança máxima entre os elementos de c_r [11, 22].

$$d(c_r) = \max\{dist(x_i, x_j)\} \ \ , onde\ x_i, x_j \in c_r. \tag{2.21}$$

- **Raio (*Radius*):** Simbolizado para um grupo c_r como $r(c_r)$. Representa o menor valor dentre todos os elementos de c_r com relação a similaridade máxima de um objeto x_i a um outro pertencente a c_r [11, 22].

$$r(c_r) = \min_{x_i \in c_r}\{\max_{x_j \in c_r}\{dist(x_i, x_j)\}\} \tag{2.22}$$

- **Estrela (*Star*):** Representado para um grupo c_r como $st(c_r)$. Equivale ao menor valor dentre todos os elementos de c_r com relação a soma de semelhanças de um elemento x_k aos outros elementos de c_r [11, 22].

$$st(c_r) = \min_{x_i \in c_r} \{ \sum_{j=1}^{|c_r|} dist(x_i, x_j) \} \tag{2.23}$$

  O cálculo também pode ser feito no modo Estrela Normalizada. A partir do resultado acima basta dividir este valor por $|c_r| - 1$ [11, 22].

- **Clique (*Clique*):** Simbolizado para um grupo c_r como $cl(c_r)$. Retrata a soma das semelhanças entre os elementos de c_r [11, 22].

$$cl(c_r) = \sum_{i=1}^{|c_r|} dist(x_i, x_j) \ \ , onde\ x_i, x_j \in c_r, \quad \forall x_j \in c_r. \tag{2.24}$$

  O critério também pode ser aplicado no modo Clique normalizado. A partir do resultado acima basta dividir este valor por $\frac{|c_r|}{(|c_r| - 1)}$ [11, 22].

Quando o conjunto de dados trabalhado pertence ao espaço Euclidiano $\mathbb{R}^p$, o critério de homogeneidade para um grupo c_r pode ser calculado considerando um centro ou centroide de c_r. Este referencial não é necessariamente um ponto do grupo. Alguns métodos que utilizam este conceito são apresentados abaixo [11, 22].

- **Soma dos Quadrados (*Sum-of-Squares*):** Representado para um grupo c_r como $ss(c_r)$ [11, 22].

$$ss(c_r) = \sum_{i=1}^{|c_r|} (d_e(x_i, \bar{c}_r\ ))^2 \tag{2.25}$$

- **Variância (*Variance*):**Simbolizado para um grupo c_r como $v(c_r)$. Equivale a divisão do resultado da Soma dos Quadrados por $|c_r|$ [11, 22].

$$v(c_r) = \frac{ss(c_r)}{|c_r|} \tag{2.26}$$

2.8.2 Critérios para separação

A seguir serão apresentados alguns dos critérios existentes para a interpretação da separação entre grupos.

- **Divisão (*Split*):** É simbolizado para um grupo c_r como $s(c_r)$. Representa a semelhança mínima entre um elemento integrante de c_r e um outro que não pertença a este grupo [11, 22].

$$s(c_r) = \min\{dist(x_i, x_j)\} \ \ , onde\ x_i \in c_r, \quad x_j \notin c_r. \tag{2.27}$$

- **Corte (*Cut*):** Representa-se para um grupo c_r como $c(c_r)$. Equivale a soma das semelhanças entre os elementos de c_r e aqueles que não pertencem ao grupo [11, 22].

$$c(c_r) = \sum_{i=1}^{|c_r|} \sum_{j=1}^{n-|c_r|} dist(x_i, x_j) \ \ , onde\ x_i \in c_r, \quad x_j \notin c_r. \tag{2.28}$$

  O critério também pode ser aplicado no modo Corte Normalizado. Após obter o resultado acima basta dividir o valor por $\frac{|c_r|}{(n - |c_r|)}$ [11, 22].

2.9 Funções de Avaliação

Uma função de avaliação faz uma análise de qualidade sobre a organização dos grupos gerados por um algoritmo de agrupamento a partir das próprias características dos dados. Geralmente é estabelecida uma relação entre a coesão interna dos grupos e a separação entre grupos para estimar a qualidade de um resultado de agrupamento. Tal prática tem sido muito reconhecida por algoritmos de agrupamento que são estruturados a partir de meta-heurísticas, onde uma função de avaliação é adotada como função objetivo a ser otimizada [28, 42].

Ao trabalhar com o Problema de Agrupamento Automático é delicada a definição de uma função de avaliação para ser adotada. Funções e algoritmos aplicados ao PC não têm um bom funcionamento quando utilizados no PCA, assim não é possível utilizar um modelo exato do Problema de Agrupamento no agrupamento automático [11]. A seguir serão apresentadas algumas funções para a avaliação de grupos que podem aplicadas ao Problema de Agrupamento Automático.

- **Índice Dunn (*Dunn index - DI*):** Nesta avaliação busca-se maximizar o valor alcançado pela solução. Também existem outras variações deste índice, que lidam com diferentes formas de coesão e separação, sendo denominadas como índices Dunn generalizados. Nessa função, a separação é avaliada através das distâncias dos elementos com relação ao seu vizinho mais próximo e a coesão é definida pelo maior valor alcançado pelo critério Diâmetro de um grupo. A função índice Dunn possui complexidade $O(n^4)$ e é apresentada como [16, 19, 41, 42, 44]:

$$DI(C) = \min_{1 \le r \le k} \left\{ \min_{1 \le s \le k, r \ne s} \left\{ \frac{\delta(c_r, c_s)}{\max\limits_{1 \le t \le k} \Delta(c_t)} \right\} \right\} \tag{2.29}$$

onde:

$$\delta(c_r, c_s) = \min_{x_i \in c_r, x_j \in c_s} \{dist(x_i, x_j)\} \tag{2.30}$$

$$\Delta(c_r) = \max_{x_i, x_j \in c_r} \{dist(x_i, x_j)\} \tag{2.31}$$

- **Índice CH (*Calinski–Harabasz index - CHI*):** Nesta função, a coesão é avaliada por meio da soma das distâncias dos elementos do grupo em relação aos seus respectivos centroides. Já a separação é definida com o uso da soma das distâncias entre o centroide de cada grupo e o centroide global do conjunto de dados. A utilização do índice CH não é uma tarefa custosa computacionalmente, superando, de modo geral, outras funções de avaliação. A sua complexidade equivale a $O(n)$. Quando utilizado por uma meta-heurística busca-se a maximização do seu valor. A função é retratada como [8, 29, 41, 42]:

$$CH(C) = \frac{n-k}{k-1} \frac{traceB(C)}{traceW(C)} \tag{2.32}$$

onde:

$$traceB(C) = \sum_{r=1}^{k} |c_r| dist(\bar{c}_r\ , \bar{x}\ ) \tag{2.33}$$

$$traceW(C) = \sum_{r=1}^{k} \sum_{i=1}^{|c_r|} dist(x_i, \bar{c}_r\ ) \tag{2.34}$$

- **Índice DB (*Davies–Bouldin index - DBI*):** O índice DB estima a coesão através da distância média dos elementos com relação ao seu respectivo centroide. Com relação a separação, é utilizada a distância entre os centroides dos grupos. Durante o seu uso tem-se como objetivo a minimização de seus resultados. A complexidade computacional da função é $O(n^2)$ e sua formulação é apresentada a seguir [14, 19, 35, 41, 42].

$$DB(C) = \frac{\sum_{r=1}^{k} R(c_r)}{k} \tag{2.35}$$

onde:

$$R(c_r) = \max_{c_s \in C,\ c_r \neq c_s} \left\{ \frac{S(c_r) + S(c_s)}{dist(\bar{c}_r\ , \bar{c}_s)} \right\} \tag{2.36}$$

$$S(c_r) = \frac{\sum_{i=1}^{|c_r|} dist(x_i, \bar{c}_r\ )}{|c_r|} \tag{2.37}$$

- **Índice CS (*CS index - CSI*):** Este índice de validação trabalha a coesão de um grupo através do valor no critério Diâmetro. Já a separação tem como base a distância do vizinho mais próximo entre os centroides. Ao adotar esta função o problema de otimização tem como

objetivo a minimização. O índice é definido abaixo e sua complexidade é denotada como $O(n^2)$ [10, 12, 42, 44].

$$CS(C) = \frac{\sum_{r=1}^{k} a(c_r)}{\sum_{r=1}^{k} b(c_r)} \tag{2.38}$$

onde:

$$a(c_r) = \frac{\sum_{i=1}^{|c_r|} \max_{x_j \in c_r} \{dist(x_i, x_j)\}}{|c_r|} \tag{2.39}$$

$$b(c_r) = \min_{c_s \in C,\ c_r \neq c_s} \{dist(\bar{c}_r\ , \bar{c}_s\ )\} \tag{2.40}$$

- **Índice Silhueta (*Silhouette index - SI*):** Esta função avalia um agrupamento empregando simultaneamente dois critérios. Para analisar a homogeneidade, utiliza-se a distância média entre os elementos de cada grupo, semelhante a metodologia do critério Clique, apresentado anteriormente. Como critério de separação, trabalha-se com a distância média entre cada elemento e os elementos do grupo vizinho que está mais próximo, de modo similar a técnica do critério Corte, já apresentado [11, 29, 42, 52, 56].

  Os valores da função Silhueta variam dentro do intervalo $[-1, 1]$, são avaliados os elementos que estão contidos em cada grupo. A medida que os elementos estão bem posicionados os resultados alcançados são positivos e mais próximos do valor 1, ou seja, ao utilizar esta função tem-se um problema de otimização de maximização. Por outro lado, a obtenção de valores negativos indica que elementos não estão bem posicionados com relação a seu grupo. A complexidade do uso desta função equivale a $O(n^2)$. A avaliação do problema de otimização que utiliza o Índice Silhueta é denotado por [11, 29, 42, 52, 56]:

$$SI(C) = \frac{1}{n} \sum_{r=1}^{k} \sum_{i=1}^{|c_r|} \frac{b(x_i) - a(x_i)}{\max\{b(x_i), a(x_i)\}} \tag{2.41}$$

onde:

$$a(x_i) = \frac{\sum_{j=1}^{|c_r|} dist(x_i, x_j)}{|c_r| - 1}\ \ , onde\ c_r \in C, \quad x_i, x_j \in c_r, \quad x_i \neq x_j. \tag{2.42}$$

$$b(x_i) = \min_{c_s \in C,\ c_r \neq c_s} \left\{ \frac{\sum_{j=1}^{|c_s|} dist(x_i, x_j)}{|c_s|} \right\}\ \ , onde\ c_r \in C, \quad x_i \in c_r, \quad x_j \in c_s. \tag{2.43}$$

2.10 Revisão da Literatura

Diversos trabalhos da literatura trazem propostas utilizando a meta-heurística Algoritmo Genético para a resolução do PCA [28].

Em 2004 foi publicado o artigo "Clusterização em Mineração de Dados" com o propósito de auxiliar na resolução dos problemas de agrupamento trabalhados em Mineração de Dados. Além de fazer uma descrição do problema, os autores também trazem aplicações e metodologias que o resolvem de forma eficiente. Neste trabalho são enfatizados os métodos que se baseiam nos conceitos de meta-heurísticas evolutivas e na forma de agrupamento conhecida como Problema de Agrupamento Automático [45].

O trabalho que tem como título "*Automatic Clustering Using a Synergy of Genetic Algorithm and Multi-objective Differential Evolution*" foi produzido no ano de 2009 apresentando um algoritmo híbrido das técnicas Evolução Diferencial e Algoritmo Genético, denominado como GADE, para a resolução do problema de agrupamento automático com estratégia *fuzzy*. Foi proposta uma metodologia de otimização multiobjetivo, trabalhando simultaneamente com duas funções: índice XB e índice FCM. Para avaliar a qualidade final das soluções de agrupamento foram utilizadas as medidas índice Silhueta e índice Rand ajustado. A partir de dez conjuntos de dados foram realizados experimentos para a comparação com dois outros trabalhos: MOCK e NSGA II. Na proposta, a representação dos grupos foi limitada por um valor K_{max} definido pelo usuário. Em geral, o trabalho apresentou melhores resultados finais comparado às duas outras estratégias da literatura [31].

Em 2009, o artigo denominado como "*A Bacterial Evolutionary Algorithm for Automatic Data Clustering*" apresentou uma estratégia para a resolução de agrupamento automático que teve como base a técnica conhecida como Algoritmo Evolutivo Bacteriano (*Bacterial Evolutionary Algorithm* - BEA). Segundo os autores, esta foi a primeira abordagem que utiliza o BEA para a resolução de tal problema. A técnica BEA é inspirada num fenômeno biológico de evolução microbiana. Diferente dos operadores tradicionais utilizados nos algoritmos genéticos, nela são adotada duas operações para a evolução da população: mutação bacteriana e transferência gênica. Este trabalho utilizou uma outra versão para estas operações a fim de manipular os comprimentos variáveis dos cromossomos que codificam diferentes soluções de agrupamento. A avaliação de aptidão teve como base a medida CS e, inicialmente, o número de grupos definido para cada cromossomo foi gerado de modo aleatório dentro de um intervalo $[2, K_{max}]$ podendo ser definido pelo usuário. A partir de diferentes conjuntos de dados foram realizados experimentos, os resultados da proposta Agrupamento Automático com BEA (*Automatic Clustering with BEA* - ACBEA) foram comparados com os algoritmos DCPSO, GCUK e o clássico *k-means*. A proposta obteve melhores resultados em relação à maioria dos conjuntos de dados [13].

O trabalho "O Problema de Clusterização Automática", apresentado 2010, propôs diversas metodologias para a resolução do problema de agrupamento automático. Foram apresentadas heurísticas tendo como base as meta-heurísticas Algoritmos Evolutivos, GRASP e ILS. Também foram propostos métodos híbridos que aplicam modelos exatos após a execução dessas heurísticas. Além disto, foi desenvolvida uma estratégia para reduzir a cardinalidade dos dados de entrada do

problema, através de uma junção de grupos parciais. Experimentos foram realizados para alguns conjuntos de dados gerados pelo próprio autor e da literatura. Os resultados obtidos por cada metodologia foram comparados entre si, aquelas que alcançaram as melhores soluções passaram por uma comparação com uma estratégia da literatura, o algoritmo CLUES. Os métodos propostos obtiveram melhores resultados do que este último, com maior eficiência e robustez [11].

O trabalho "Proposta de um método de classificação baseado em densidade para a determinação do número ideal de grupos em problemas de clusterização" foi apresentado em 2012 e apresentou um método para a resolução do problema de agrupamento automático denominado como MRDBSCAN. Nesta proposta foi utilizado o algoritmo DBSCAN, baseado em densidade, para identificar o número de grupos do agrupamento. Tal algoritmo faz uso de certos parâmetros, na proposta do MRDBSCAN eles foram calibrados utilizando a técnica DistK. A avaliação das soluções foi feita através da função índice silhueta. Os experimentos foram realizados para 83 conjuntos de dados e seus resultados foram comparados com os seguintes algoritmos da literatura: CLUES, CLUSTERING, SAPCA, AEC-RC e AECBL1. Grande parte de seus resultados foram equivalentes ou superiores aos outros trabalhos exceto para os de um grupo específico que num sentido geral foram apenas razoáveis [55].

O trabalho "*A two-stage genetic algorithm for automatic clustering*" apresentou, em 2012, o algoritmo denominado como TGCA para a resolução do problema de agrupamento automático. Tal proposta consistiu num chamado Algoritmo Genético combinado a técnica k*-means*, com duas etapas. Primeiramente foi investigada uma quantidade de grupos k pertencente ao intervalo $[k_{min}, k_{max}]$, onde $k_{min} = 2$, $k_{max} = \sqrt{n}$ e n é a quantidade total de elementos. Na sequência, os melhores centros para os grupos foram buscados para encontrar a partição dos dados resultante. Utilizando a função índice CH como avaliação, os experimentos foram realizados em alguns conjuntos de dados reais e artificiais. Seus resultados foram comparados com outros trabalhos da literatura: HKM, ANJW e SGKC. O desempenho alcançado foi melhor do que as outras três propostas [24].

O algoritmo AKC-BCO foi apresentado em 2014 num estudo que tem o título "*Automatic kernel clustering with bee colony optimization algorithm*". Este trabalho apresentou uma resolução para o problema de agrupamento automático através da técnica de otimização Colônia de Abelhas. O algoritmo definiu a quantidade de grupos limitado a uma quantidade K_{max}, onde $K_{max} = \sqrt{n}$ e n é o tamanho do conjunto de dados. Os dados foram organizados a partir destas formações. Foram realizados experimentos para alguns conjuntos de dados e a avaliação foi feita através da medida *kernel* CS. Seus resultados foram comparados com os algoritmos DCPSO, DCPG e AKC-MEPSO. Por meio dos resultados obtidos, foi observado que AKC-BCO possui maior estabilidade e precisão que os outros três trabalhos. Também foi realizada uma aplicação a um problema médico do mundo real e os resultados obtidos também demostraram um melhor agrupamento do que o de outras propostas [32].

O trabalho que leva como título "*AK-means: an automatic clustering algorithm based on K-means*", apresentado em 2015, teve como objetivo superar o problema da definição inicial do número de grupos encontrado no algoritmo K-*means*. Assim, o AK-*means* propõe uma abordagem onde, inicialmente, o número de grupos k é definido como a raiz quadrada do número total de elementos. Tal decisão tem origem no fato de que este número varia entre 2 e $\sqrt{n}$. Em sequência o agrupamento automático vai sendo realizado com reinicializações consecutivas do K-*means*. Dessa forma, tem-se um algoritmo sem parâmetros para resolução do agrupamento. O processamento se mantém até o valor de k igual a 2. Em seguida o algoritmo obtém o número de grupos e a partição ótimos a partir de todos os resultados já armazenados. Tal otimalidade é medida através da função de avaliação índice CH, quanto maior o seu valor melhor o resultado do particionamento. Os experimentos foram realizados em diversos conjuntos de dados para fazer uma comparação com o algoritmo G-*means*. Eles apresentam como resultados o tempo de processamento, o número de grupos e o valor atingido pela função índice silhueta na melhor partição encontrada. Esta função é aplicada apenas na representação da resolução final, não é utilizada durante os cálculos internos do algoritmo. Foi observado que, a proposta dos autores é eficaz e supera os resultados do outro trabalho no que se refere a precisão de agrupamento e estimativa do número correto de grupos [29].

Em 2016 o trabalho "Abordagem baseada em *Continuous* GRASP para clusterização de dados" apresentou uma metodologia baseada na meta-heurística *Continuous* GRASP (C-GRASP) para solucionar o problema de k-clusterização particional. Para a proposta deste trabalho, denominada como C-GRASP-Clustering, foram realizados experimentos que avaliaram seis conjuntos de dados. Seus resultados foram comparados com outros trabalhos da literatura, sendo estes: C-GRASP original, new C-GRASP, k-*means*, BH e BPFPA. O algoritmo apresentado alcançou soluções de qualidade igual ou melhor comparado às outras metodologias [48].

O trabalho "*Automatic clustering using nature-inspired metaheuristics: A survey*", publicado em 2016, apresentou diversas questões relacionadas ao uso de meta-heurísticas inspiradas na natureza para a resolução do problema de agrupamento automático. Além de fazer uma apresentação do problema o artigo também falou sobre os principais algoritmos já propostos, termos e conceitos básicos relacionados ao agrupamento automático, estruturas utilizadas na codificação de soluções, funções objetivo existentes e medidas de proximidade de objetos [28].

O artigo apresentado em 2016 denominado como "*A multiobjective optimization based entity matching technique for bibliographic databases*" propôs uma abordagem para a resolução do problema de correspondência de entidades em bases de dados bibliográficos. A metodologia teve como objetivo a determinação do particionamento e de sua quantidade de grupos de modo automático. Através do algoritmo genético NSGA-II foi adotada uma estratégia de otimização multiobjetivo onde oito funções são adotadas, sendo estas: índice CH, índice CS, índice DB, índice Dunn, índice I, índice PS, índice Silhueta e índice XB. Segundo os autores, esta foi a

primeira abordagem multiobjetivo a buscar a resolução desse tipo de problema. Experimentos foram realizados para oito conjuntos de dados bibliográficos e seus resultados foram comparados com duas outras técnicas: DBLP e ArnetMiner. O método proposto obteve melhores resultados em várias situações [42].

Em 2017 foi apresentado o trabalho "Metaheurística inspirada no comportamento das formigas aplicada ao problema de agrupamento" que propôs um algoritmo fundamentado numa proposta baseada no comportamento de formigas para a resolução de problemas de agrupamento de dados. O algoritmo ACO proposto passou por experimentos para 68 conjuntos de dados da literatura e a função adotada para a avaliação dos agrupamentos foi a Índice Silhueta. Seus resultados apresentaram o valor silhueta e o número de grupos encontrado para cada solução e foram comparados com outro método da literatura denominado como GradeBL. A proposta alcançou resultados satisfatórios para parte das instâncias verificadas. Os autores acreditam que o algoritmo é promissor e, com relação às soluções que foram insatisfatórias, melhores resultados podem ser alcançados com a implementação de certos aprimoramentos [46].

3 HEURÍSTICAS PARA A RESOLUÇÃO DE UM AGRUPAMENTO DE DADOS

Existem diversos algoritmos que buscam uma solução ótima para problemas, porém nem sempre estas abordagens trazem êxito. Certos problemas são tão complexos de ser resolvidos que algoritmos exatos não são capazes obter uma solução ótima. Nestas situações o conceito de meta-heurísticas pode se aplicado, assim uma boa solução viável, próxima da ótima, consegue ser encontrada [25].

Define-se por meta-heurística uma estratégia estruturada que orienta a busca por um ótimo global através de informações que são coletadas no decorrer do processo. Apesar deste mecanismo não garantir que irá alcançar exatamente a solução ótima, ou uma próxima a ela, ele é propenso a se direcionar para soluções muito boas de forma rápida. Assim, seu uso se torna conveniente ao tratar de problemas grandes e complicados. Pode-se citar algumas das meta-heurísticas mais conhecidas: Algoritmos Evolutivos, GRASP, Busca Tabu e VNS [11, 25].

A meta-heurística é uma metodologia genérica que proporciona tanto a estrutura quanto orientações de estratégia gerais para o desenvolvimento de um método com finalidade mais específica, denominado heurística. Este último tem como base um conceito mais abstrato para se ajustar a um problema particular que precisa ser solucionado [11, 25].

Muitas vezes torna-se necessária a utilização de métodos heurísticos para lidar com problemas de agrupamento, especialmente para casos onde o número de grupos não é um dado conhecido a priori. A procura pelo melhor resultado no espaço de soluções viáveis é um problema combinatório de complexidade NP-difícil, ou seja, pertence a uma classe de problemas na qual o número de possíveis soluções cresce de forma exponencial à proporção do número de elementos. O uso de métodos exatos é computacionalmente inviável devido a avaliação custosa de todas as possíveis formas de realizar o agrupamento, portanto torna-se necessário o uso de técnicas especiais para encontrar resultados. Através de heurísticas é possível obter uma solução, mesmo que esta não seja a melhor possível. Assim, existem muitos algoritmos que propõem solucionar o agrupamento de dados. Eles buscam encontrar uma solução próxima da ótima em um tempo computacional razoável [11, 29, 34, 45, 48].

Inicialmente, a existência de uma grande quantidade de métodos de agrupamento disponíveis pode parecer complicado. Graças a diversidade de aplicações deste problema, não há uma heurística genérica que consiga boas soluções para todos os tipos de agrupamento. Dependendo de qual seja o objetivo pode-se aplicar, simultaneamente, diferentes técnicas e combinar seus resultados ao invés de adotar apenas uma única [25, 34, 45].

Com o aumento de novas propostas para a resolução do agrupamento automático, meta-heurísticas inspiradas na natureza têm sido adotadas na busca por soluções. Estas estratégias solucionam um problema modelando-o de forma análoga ao comportamento de certos fenômenos naturais [28].

3.1 Algoritmos Evolutivos

Modelos computacionais inspirados pela Teoria da Evolução das Espécies e pelos Princípios Básicos da Herança Genética, relatados respectivamente por Charles Darwin e Gregor Mendel, que buscam solucionar problemas de otimização são conhecidos como Algoritmos Evolucionários ou, também, Algoritmos Evolutivos [11, 44, 45].

O modelo proposto por Darwin apresenta uma população que passa por um processo de evolução natural, onde seus indivíduos conseguem se adaptar ao meio em que vivem. A partir dos processos de seleção natural, reprodução, recombinação genética e mutação, que serão apresentados posteriormente, aqueles que estão melhor ajustados alcançam uma maior expectativa de sobrevivência e a geração de descendentes. A seleção natural favorece os indivíduos que possuem uma capacidade de sobrevivência mais alta. Desta maneira, torna-se possível a melhoria da qualidade média da população, direcionando o processo de evolução ao encontro do indivíduo que é adaptado ao ambiente por completo, aquele que é considerado "ótimo" [11, 33, 44, 45].

A genética combinada a teoria da evolução possibilita que, em meio a um espaço busca extenso, com todas as combinações de sequências de DNA existentes, os seres vivos surjam e se adaptem ao meio ambiente. Estes conceitos são utilizados na evolução de uma população de indivíduos, também chamados de soluções, ao longo das iterações do algoritmo, denominadas gerações [11, 44, 45].

A Computação Evolucionária é uma área de estudo interdisciplinar. Em geral, esta metaheurística pode ter a seguinte classificação [20, 21, 45]:

- Estratégias de Evolução: Nesta abordagem não há seleção de indivíduos para que seja gerada uma próxima população. Utilizando o operador de mutação cada membro da população dá origem a um único filho [45].

- Programação Evolutiva: Máquinas de estados finitos evoluem de forma similar ao processo aplicado nas Estratégias de Evolução [45].

- Algoritmos Genéticos: Esta é a categoria mais conhecida dos Algoritmos Evolutivos, seu fundamento está na sobrevivência dos melhores indivíduos. A partir de um conjunto de soluções, uma busca adaptativa faz com que o grau de qualidade de cada uma evolua através de um processo feito em várias etapas [20, 21, 45].

- Programação Genética: Este método automático de programação possibilita a evolução de programas computacionais, os quais de modo exato ou aproximado solucionam problemas [20, 21, 45].

3.1.1 Algoritmo genético

A proposta desta categoria trás um processo adaptativo e paralelo para a busca de soluções. O fato frequente de as soluções correntes influenciarem a busca pelas próximas faz com que tal processo seja adaptativo, já o paralelismo se deve aos vários resultados que são avaliados constantemente. Os Algoritmos Genéticos são muito bons na resolução de problemas pois conseguem explorar diversas partes do espaço de busca de soluções viáveis e evoluir gradualmente em direção àquelas de melhor qualidade. Esta metodologia surgiu, inicialmente, com um enfoque maior na evolução dos indivíduos de uma população do que na resolução de problemas de otimização. Como buscava-se construir sistemas artificiais embasados nos processos adaptativos presentes em sistemas naturais, tais processos precisavam se tornar compreensíveis. Os termos específicos da Biologia são utilizados a fim de simplificar a descrição e utilização dos algoritmos [20, 25, 45].

O comportamento de um Algoritmo Genético equivale ao que acontece na natureza com os seres de uma população. Em sua sobrevivência, há uma disputa entre estes indivíduos pelos variados recursos disponíveis em seu habitat como, por exemplo, abrigo, água e alimentação. No ambiente em que vivem, cada indivíduo se difere um do outro quanto a adaptação por causa de suas características externas, conhecidas como fenótipo. Tais particularidades fazem referência à sua constituição genética, denominada genótipo. A forma como um indivíduo se adapta influencia a sua sobrevivência o suficiente para chegue a fase de procriação. Por meio desta, dois indivíduos tem suas características genéticas combinadas e transmitidas a seus descendentes. Assim, é alta a chance de indivíduos com o necessário para uma vida mais duradoura constituírem as gerações sucessoras. O contexto relatado é conhecido como Seleção Natural [11, 20, 25, 45, 54].

A partir deste conceito grandes espaços de busca conseguem ser explorados de forma eficiente num problema. Com buscas direcionadas e inteligentes, certos problemas que possuem poucas soluções consideradas aceitáveis, alcançam um ótimo desempenho com o uso desta técnica. A otimização de funções matemáticas, o Problema do Caixeiro Viajante e o Problema de Otimização de Rota de Veículos são alguns exemplos de problemas em que os Algoritmos Genéticos podem ser aplicados para sua resolução [15, 20].

Para resolver um problema, as suas possíveis resoluções são codificadas em cadeias de caracteres que são denominadas como indivíduos. Vários destes são processados num grande conjunto conhecido como população. Cada população corresponde às soluções temporárias que estão sendo analisadas. Estas soluções se referem aos membros viventes no momento. Alguns destes indivíduos sobrevivem, alcançam a fase madura e geram filhos. Algumas características dos pais são transmitidas para os descendentes. Dado que muitas soluções são avaliadas ao mesmo tempo, a execução é mais dinâmica do que numa avaliação sequencial, feita individualmente [11, 15, 20, 25, 30].

A função objetivo utilizada neste método de otimização é denominada como função de

aptidão. A qualidade de cada elemento da população, ou seja seu fenótipo, é obtida por meio desta função, que avalia seu genótipo de acordo com o problema modelado. Da mesma maneira que na evolução natural, os indivíduos mais aptos devem ter maior chance de passar sua codificação genética para sua descendência [11, 15, 45].

A busca por soluções é direcionada com o uso de regras probabilísticas. Cada iteração do processo de busca chama-se geração. Assim, em cada um destes ciclos cria-se uma nova população a partir dos indivíduos da geração anterior. Como os indivíduos mais adaptados têm maior probabilidade de gerar descendentes para a população seguinte, um algoritmo genético tende a produzir populações cada vez melhores ao longo de sua execução [15, 25].

Diferente de outras metodologias de busca, não são utilizadas apenas informações locais durante a execução. Operadores genéticos são aplicados nos indivíduos para que diversas regiões do espaço de busca sejam exploradas, desta forma é menos provável que o algoritmo convirja para uma solução ótima local. Graças a eles, novas características podem ser estudadas, levando a uma possível evolução dos indivíduos. Assim, a população tende a convergir para a solução ótima do problema, que é uma excelente combinação de características para o que está sendo trabalhado. A evolução natural das soluções faz com que grande parte dos problemas de otimização possam ser resolvidos com a aplicação dos Algoritmos Genéticos. Uma analogia entre os Algoritmos Genéticos e a Evolução Natural é apresentada no Quadro 3.1 [11, 20, 45, 54].

Quadro 3.1: Uma analogia entre os Algoritmos Genéticos e a Natureza [20].

Evolução Natural	**Algoritmos Genéticos**
Meio ambiente	Problema
Indivíduo	Solução
Cromossomo	Representação (Palavra binária, vetor e etc)
Gene	Característica do Problema
Alelo	Valor da Característica
Loco	Posição na Representação
Genótipo	Estrutura
Fenótipo	Estrutura submetida ao problema
Reprodução	Operador de Cruzamento
Mutação	Operador de Mutação
População	Conjunto de Soluções
Gerações	Ciclos

3.1.1.1 A História

Em 1930, a publicação do livro *The Genetic Theory of Natural Selection* trouxe um modelo matemático que tentava demonstrar a teoria de Darwin. A partir deste conceito matemático, John Holland se dedicou aos estudos de processos naturais adaptáveis. Desta forma, ele formulou a ideia dos algoritmos genéticos em meados da década de 60. Posteriormente, em companhia de seus alunos e colegas de universidade , os algoritmos foram desenvolvidos e aperfeiçoados entre as décadas de 60 e 70 [20, 43, 45].

Com a criação desses algoritmos, Holland desejava estudar formalmente o fenômeno da adaptação presente na natureza. Graças a motivação de desenvolver modelos em que sistemas computacionais conseguissem lidar em sua temática com mecanismos da adaptação natural ele publicou o livro *Adaptation in Natural and Artificial Systems*, em 1975. Além desta, uma outra obra que também é tida como referência na área é a escrita por David Goldberg, em 1989, chamada *Genetic Algorithms in Search, Optimization and Machine Learning* [20, 33, 43].

3.1.1.2 O Algoritmo

Num Algoritmo Genético, um processo contínuo se repete evoluindo a cada ciclo executado, sendo um critério de parada responsável por controlar a sua execução. A dinâmica existente numa geração é ilustrada na Figura 3.1. Cada etapa será apresentada com mais detalhes posteriormente [20].

Através deste procedimento iterativo, os indivíduos passam por certos operadores para que populações descendentes sejam geradas. Como o valor máximo de aptidão possível para um indivíduo, ou seja a solução ótima para o problema, dificilmente é encontrado, é definido um limite máximo de gerações. Deste modo, garante-se que ocorra a finalização do processo de busca. Uma boa solução, muito próxima da ótima, pode ser encontrada dentro de um número fixo de iterações do algoritmo. O passo a passo da metodologia é descrito no Algoritmo 4 [15, 20, 45].

3.1.1.3 A Codificação do Problema

Os dados de entrada são codificados de forma que o algoritmo possa atuar sobre eles. A definição da codificação é de grande importância para que se tenha um bom desempenho na execução e bons resultados sejam gerados. Um alfabeto finito codifica simbolizando com elementos binários (por exemplo, verdadeiro, falso) ou numéricos (por exemplo, $0, 1, \ldots, n$). A representação de um indivíduo, também nomeado como cromossomo, é feita por meio de uma cadeia de símbolos. Cada uma destas posições é denominada gene, uma unidade de informação do domínio do problema, e a concatenação de uma série destes se refere a uma solução [11, 15, 20, 28, 45].

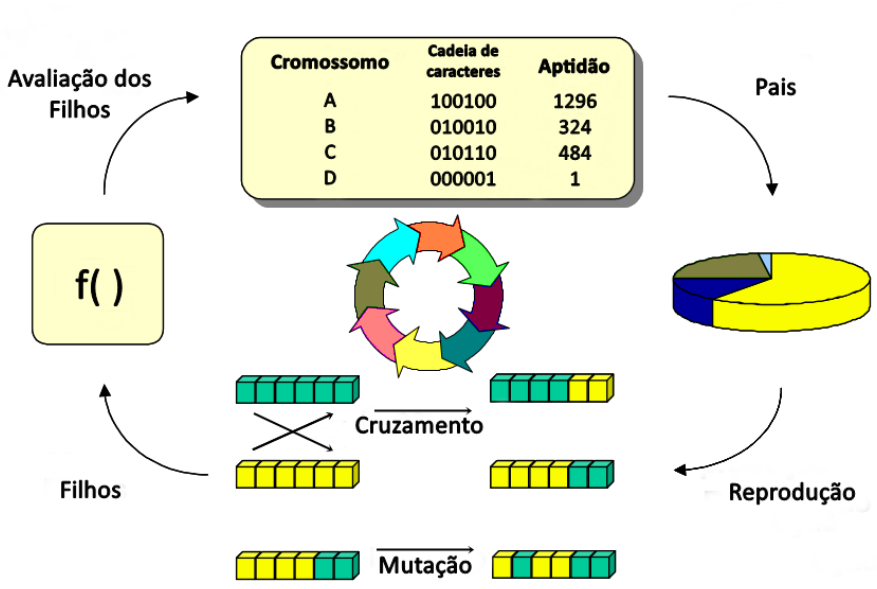


Figura 3.1: Representação do ciclo de um Algoritmo Genético [20].

Algoritmo 4: Algoritmo Genético tradicional

```
1  início
2      i= 0;
3      Gerar a população inicial P(0);
4      Avaliar a população P(0);
5      enquanto critério de parada não é satisfeito faça
6          i++;
7          Gerar a população P(i) a partir de P(i-1);
8          Aplicar os operadores Seleção e Reprodução a P(i);
9          Aplicar o operador Cruzamento a P(i);
10         Aplicar o operador Mutação a P(i);
11     fim
12     Informar a melhor solução encontrada.
13 fim
```

A estrutura para os cromossomos é definida de acordo como as soluções do espaço de busca podem ser simbolizadas. A representação binária é a mais usual, porém muitas vezes, o problema necessita que mais símbolos façam parte do alfabeto. Cada gene que constitui a solução deve ter a possibilidade de assumir todos os símbolos que estão presentes no domínio do problema, dado que o algoritmo deve ser capaz de retratar qualquer solução que o mesmo seja capaz de

atingir. Assim, o espaço de busca do Algoritmo Genético deve ser igual ao do problema em estudo, abrangendo cada uma das possíveis configurações que um indivíduo pode adotar [11, 20, 28, 45].

A codificação de indivíduos deve ser válida de modo que sejam inexistentes soluções que não pertençam ao espaço de busca do problema. Caso o contrário aconteça, o próprio algoritmo deve alterar a representação para uma configuração que seja coerente. O algoritmo deve, também, impossibilitar que diferentes representações correspondam a uma mesma solução. Tais redundâncias podem afetar a convergência da população [45].

A solução real do problema é construída a partir da decodificação do cromossomo. Dessa forma, é possível avaliá-la através da função de aptidão [20].

Podem ser apresentadas como estruturas para a codificação de uma solução de agrupamento as seguintes formas: binária, inteira e real. Ao trabalhar com o Problema de Agrupamento Automático a estrutura de codificação precisa trabalhar com uma variação na quantidade de grupos, por este ser um valor inicialmente desconhecido. Uma intervalo $[K_{min}, K_{max}]$ que pode ser utilizado tem como limites $K_{min} = 2$ e $K_{max} = \sqrt{n}$, onde n é o número de elementos do conjunto de dados. A fim de exemplificar os diferentes formatos de codificação para o agrupamento automático será considerado um conjunto de dados A de dez elementos que serão dispostos em grupos na representação, estes dados são apresentados na Figura 3.2 [28].

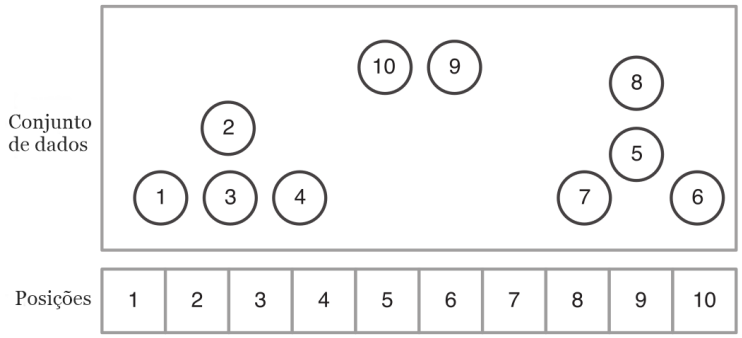


Figura 3.2: O conjunto de dados A [28].

3.1.1.3.1 Codificação Binária

Uma solução é descrita através de uma cadeia binária de tamanho igual a n, cada uma de suas posições representa um elemento do conjunto de dados. Um valor igual a “1” simboliza que tal objeto é tido como um referencial do grupo, do contrário o valor de sua referente posição equivale a "0". Na Figura 3.3 são ilustradas soluções para o conjunto de dados apresentado com o número de grupos k igual a três e a cinco [28].

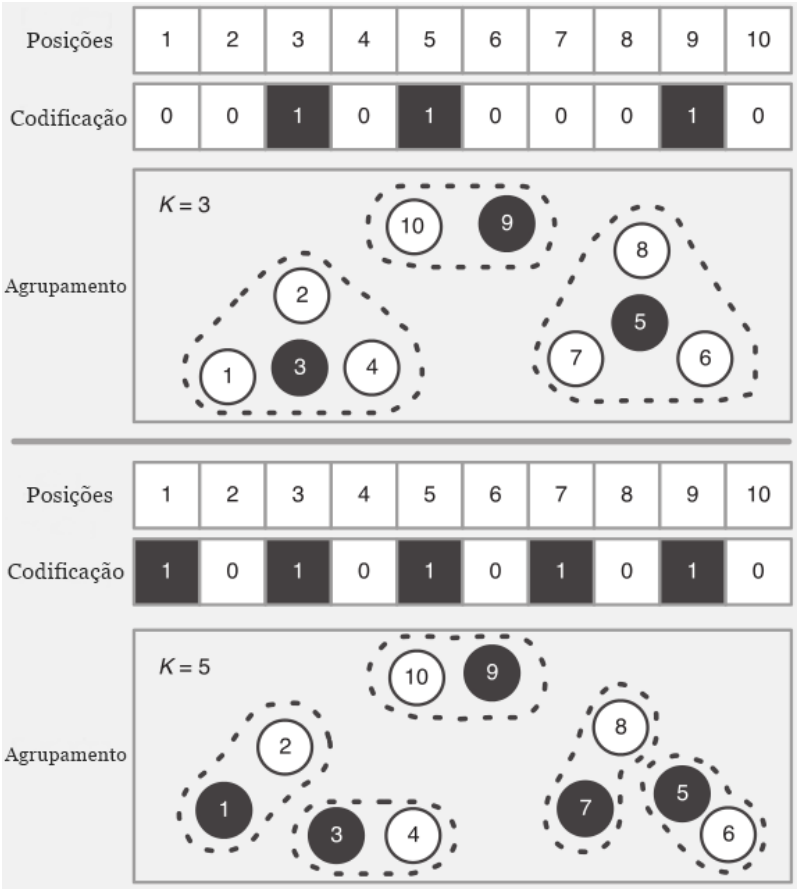


Figura 3.3: Exemplo de codificação binária [28].

3.1.1.3.2 Codificação Inteira

Semelhante ao caso anterior, esta representação também codifica numa cadeia de caracteres a associação entre elementos e grupos, porém a posição é simbolizada por um número inteiro. Existem duas formas de realizar uma codificação inteira: codificação baseada em rótulos e codificação baseadas em grafos [28].

- **Codificação baseada em rótulo:** Na cadeia de caracteres, cada elemento possui um valor inteiro referente a identificação de um grupo. Tal valor pertence ao intervalo $[1, K_{max}]$, onde K_{max} é o número máximo de grupos permitido [28].

  Na codificação inteira, um algoritmo de agrupamento pode percorrer um espaço de busca muito maior do que o espaço original das soluções devido a redundâncias, pois K_{max} representações distintas equivalem a uma mesmo resultado de agrupamento. Por exemplo, num

conjunto de dados de 5 elementos com $k = 2$, as soluções $S_1 = \{11122\}$ e $S_2 = \{22211\}$ equivalem exatamente ao mesmo agrupamento. Uma forma de solucionar esta situação seria a utilização de algum procedimento de renumeração [28].

Na Figura 3.4 é exemplificada esta forma de codificação para o mesmo conjunto A com dois casos, um com o número de grupos $k = 3$ e outro com $k = 5$ [28].

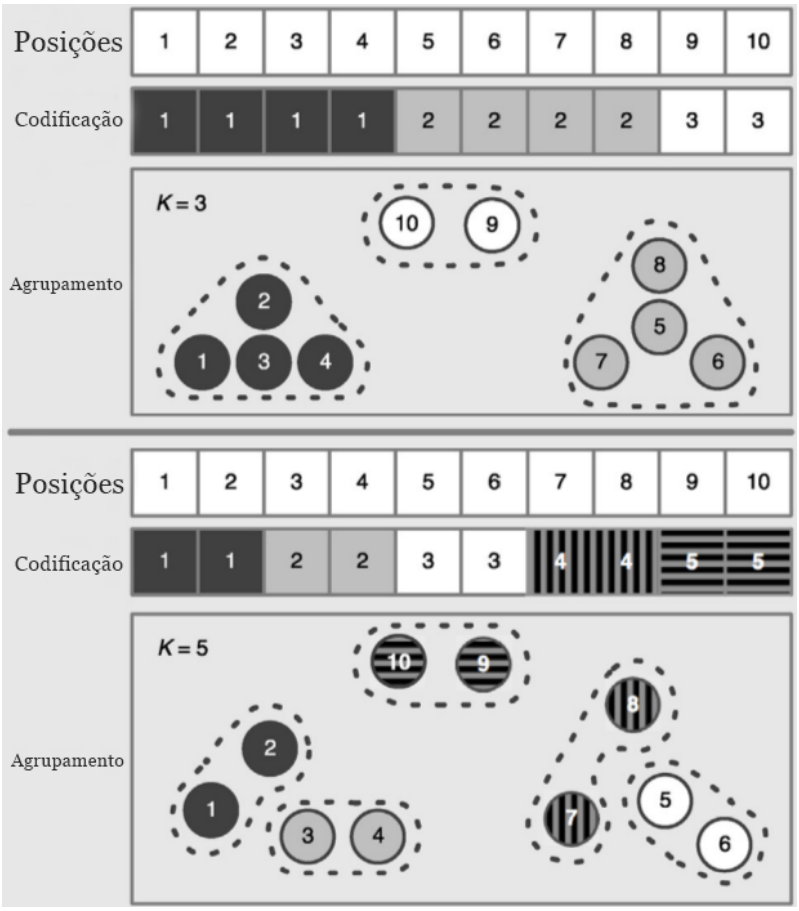


Figura 3.4: Exemplo de codificação inteira baseada em rótulo [28].

- **Codificação baseada em grafos:** Cada elemento i do conjunto de dados possui um valor j associado na cadeia, onde j pertence a $[1, n]$ e n é a quantidade total de elementos. Tal valor indica que há uma relação do elemento i com o j e ambos são alocados no mesmo grupo. O resultado de agrupamento é gerado detectando todos os componentes conectados do grafo direcionado. Os exemplos ilustrados na Figura 3.5 apresentam soluções para o conjunto de dados A com o número de grupos definido como $k = 3$ e $k = 5$ [28].

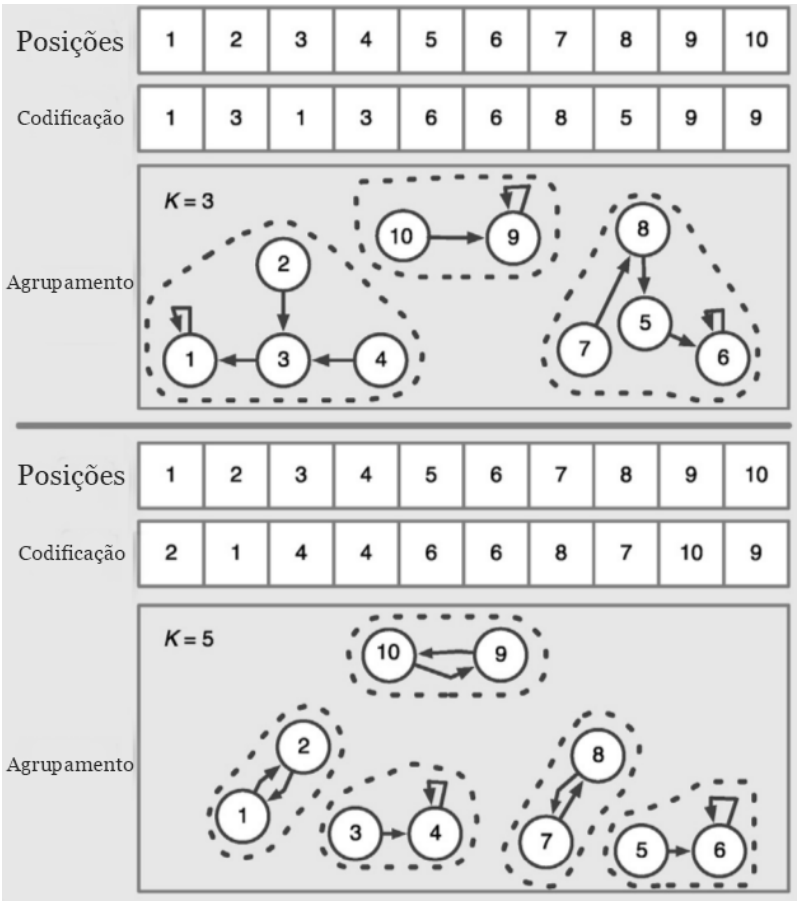


Figura 3.5: Exemplo de codificação inteira baseada em grafos [28].

3.1.1.3.3 Codificação Real

Nesta forma de codificação, um resultado de agrupamento retrata o posicionamento de cada referencial de grupo dentro de um espaço de características de dimensão m. Os k grupos que acomodarão os elementos do conjunto são representados por referenciais que são codificados numa cadeia de caracteres de números reais com tamanho $m * k$, onde m equivale ao número de características de cada elemento. Esta representação costuma ser utilizada de duas formas: codificação de comprimento de fixo ou variável [28].

- **Codificação de comprimento fixo:** Nesta codificação trabalha-se com uma quantidade máxima de referenciais K_{max} que é predefinida. Deste modo, para uma população de um algoritmo todos os indivíduos possuem o mesmo tamanho de cadeia, $m * K_{max}$, durante todo o processo de otimização. A especificação de quais referenciais estão ativos numa

iteração pode ser definida por um vetor máscara ou limites de ativação, por exemplo. Na Figura 3.6 é apresentado um exemplo desta codificação com $K_{max} = 5$ e $k = 3$ referenciais ativos [28].

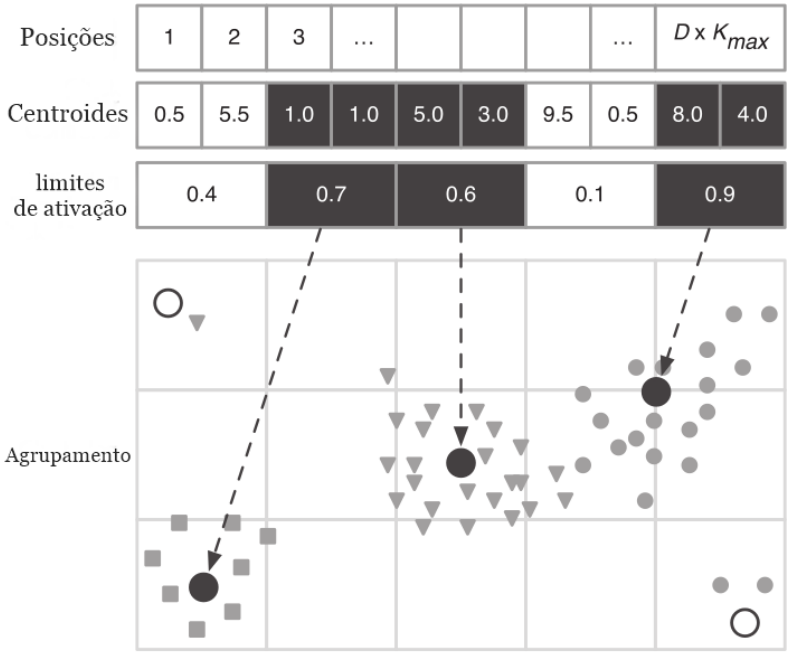


Figura 3.6: Exemplo de codificação real de comprimento fixo [28].

- **Codificação de tamanho variável:** Considerando um conjunto de soluções de agrupamento $C = \{C_i | i = 1, \ldots, p\}$, nesta codificação cada solução C_i define um número específico de referenciais K_i. Deste modo, o tamanho da cadeia de caracteres que representa um resultado não é fixo, sendo estabelecido como $m * K_i$. Assim, para uma população de um algoritmo o processamento precisaria lidar com indivíduos de tamanhos variados [28].

3.1.1.4 A Função de Aptidão (Fitness)

A qualidade de um indivíduo é definida por esta função. Em sua representação, aqueles que possuem um alto valor correspondem a uma solução melhor na avaliação do que indivíduos com um menor grau de aptidão. Para trabalhar com diferentes tipos de problemas de otimização utilizando Algoritmos Genéticos basta especificar a função de aptidão com uma que seja própria à questão abordada. Diferente de outros métodos de otimização em que apenas uma avaliação de solução é feita a cada iteração, os Algoritmos Genéticos fazem uma grande quantidade de avaliações por execução. Por isso, a função de aptidão deve ser implementada de modo que a mesma seja ágil ao processar uma população de soluções [11, 15, 33, 43, 45].

3.1.1.5 O Tamanho da População

O tamanho da população determina a quantidade de cadeias de caracteres pertencentes a mesma. Quanto maior o número de indivíduos desse conjunto, maior a possibilidade da solução ótima ser encontrada. Como Algoritmos Genéticos possuem um esforço computacional elevado ao lidar com uma população de indivíduos, a definição desse tamanho deve ser cautelosa para que o desempenho de sua execução não seja afetado. Uma população com muitos indivíduos pode necessitar de um alto custo de processamento, porém uma população pequena pode ser direcionada para ótimos locais de baixa qualidade em pouco tempo [15, 45].

3.1.1.6 Construção da População Inicial

Na inicialização da população são definidos os indivíduos que serão trabalhados na primeira geração. A construção deste conjunto é uma etapa muito importante e quando não bem trabalhada pode afetar a qualidade das soluções alcançadas, assim como o número de gerações ou de elementos pertencentes a população também podem afetar, por exemplo. A formação destes indivíduos é definida por cadeias de tamanho fixo, podendo acontecer aleatoriamente ou através de uma heurística. Apesar da criação aleatória ser muito utilizada ao empregar a segunda estratégia os resultados podem ser conduzidos a uma parte do espaço de busca que se aproxime da solução ótima [15, 20, 45].

3.1.1.7 Critério de Parada

A quantidade total de gerações e a convergência da população são duas condições de parada do algoritmo muito empregadas. Deve-se ter prudência ao estabelecer tal critério para que bons resultados possam ser obtidos. A solução ótima pode não ser alcançada se o número de gerações estipulado for insuficiente. Se o critério adotado for referente a convergência da população, pode ser estabelecido que o processamento finalizará quando um determinado número de indivíduos forem iguais, ou todos eles, sendo que este último caso não é garantido que aconteça [45].

3.1.1.8 As Gerações

A cada geração, uma nova população é gerada a partir da anterior, com a mesma quantidade n de indivíduos. A formação do novo conjunto de indivíduos é feita por meio de operadores genéticos. Ao final dos procedimentos a população recém-criada é avaliada e substitui a anterior. De um modo geral, as soluções obtidas pelo algoritmo evoluem a cada geração e no final são encontrados indivíduos de alta qualidade [15, 20, 30, 43, 45].

3.1.1.9 Os Operadores Genéticos

Os operadores genéticos são aplicados numa população anterior para que uma nova seja formada. Existem alguns operadores básicos que são executados de forma sequencial nos Algoritmos Genéticos: Seleção, Reprodução, Cruzamento e Mutação [15, 45].

3.1.1.9.1 Seleção

A etapa de seleção decide quais indivíduos irão passar pela reprodução para que seja formada a população da próxima geração. Sabe-se que, no final de cada iteração ocorre uma avaliação da população, portanto são conhecidos os indivíduos de alta qualidade. O algoritmo tende a favorecer aqueles que são mais aptos. Deste modo, o operador de seleção é responsável pela conservação dos melhores para que estes transmitam suas características para as gerações descendentes [11, 15, 20, 43, 45].

A forma como é realizada a seleção tem grande peso no desempenho do Algoritmo Genético. Esta delicada questão pode afetar a evolução da população com casos de convergência lenta ou prematura, um processo de seleção justo pode evitar ou amenizar estes problemas. A simulação do processo de seleção natural deve possibilitar que indivíduos pais de alta qualidade tenham uma maior chance de gerar descendentes porém sem impedir que indivíduos menos aptos também se reproduzam. Tal ação se justifica no fato de que, cromossomos com menor qualidade podem ter características genéticas favoráveis à concepção de uma prole que possua a melhor solução para o problema, e tais particularidades podem não ser presentes em nenhum outro membro da população. Quando a população é rapidamente marcada por características de indivíduos de alta qualidade mas que não equivalem a solução ótima ocorre o que se chama convergência prematura. Em vista disto, o problema é conduzido a um ótimo local distante do global, com uma menor qualidade. Já a convergência lenta, ao contrário da prematura, pode não obter ótimos locais de boa qualidade, ou nem até mesmo um ótimo global. Em geral, neste caso, não há variedade o bastante na população para que se evolua rumo a solução ótima. O elevado valor médio da aptidão faz com que a sua diferença com o melhor indivíduo seja baixa [33, 45].

De acordo com a aptidão são escolhidos os pares de indivíduos da população. Existem diferentes formas de realizar a seleção, as duas mais famosas serão apresentadas a seguir [15, 20, 45].

- **Seleção por roleta:** Esta técnica faz uma analogia ao jogo de roleta presente em cassinos. Membros da geração anterior são selecionados aleatoriamente através de uma roleta para compor a nova população. Todos os indivíduos possuem uma determinada porção na roleta de tamanho proporcional a qualidade de sua aptidão. Assim, cada um possui uma certa chance de ser escolhido porém os mais aptos são mais propensos à seleção. A quantidade

de vezes que se gira a roleta é relativa ao tamanho da população, um indivíduo é selecionado por rodada. Pode ocorrer que um mesmo indivíduo seja escolhido várias vezes. Este método não é indicado para casos onde um dos indivíduos possui um grau de aptidão muito superior aos demais pois assim a seleção sempre levaria a mesma escolha, resultando numa convergência prematura. Na Figura 3.7 é apresentado um exemplo da seleção de indivíduos pelo método da roleta [15, 20, 45, 54].

Ind	F_{Obj}	$1/F_{Obj}$	Probab.	Faixa
1	5.6	0.18	0.19	0 – 0.19
2	12	0.08	0.09	0.19 – 0.28
3	8	0.13	0.13	0.28 – 0.41
4	2.4	0.42	0.44	0.41 – 0.85
5	7	0.14	0.15	0.85 – 1.00
Σ	---	0.95	1.00	

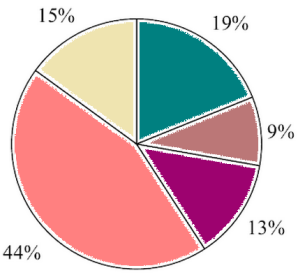


Figura 3.7: Exemplo da Seleção por roleta [54].

- **Seleção por torneio:** Existem diversas variações de implementação para esta forma de seleção. Na mais básica, para cada novo elemento que fará parte da população descendente são selecionados t indivíduos da geração anterior de modo aleatório. Quão maior for o valor de t mais rápida será a convergência da população, mas em geral o seu valor é 2. Os escolhidos disputam um com o outro, vencerá o mais apto [20, 45].

  O exemplo da Figura 3.8 ilustra o funcionamento desta forma de seleção. Considere no quadro alguns indivíduos com suas respectivas aptidões. Para assumir o lugar dos oito indivíduos da próxima geração são realizados, para cada elemento, um torneio com três componentes. Os indivíduos vencedores estão representados em destaque [20].

3.1.1.9.2 Reprodução

A substituição dos indivíduos para a população da próxima geração é especificada na fase da reprodução. Nela, o código genético dos escolhidos é copiado para fazer parte do novo conjunto de indivíduos. A seguir são apresentadas algumas técnicas [20, 45].

- **Troca de toda a População:** Em cada geração toda a população anterior é substituída. Deste modo, são selecionados $n/2$ pares de indivíduos para gerar n descendentes [20].
- **Elitismo:** Todos os indivíduos são trocados, porém há a garantia da cópia do melhor elemento da população precedente para a nova que será construída [15, 20, 45].

Indivíduo	Aptidão
x1	200
x2	100
x3	9500
x4	100
x5	100
x6	10000
x7	1
x8	40

Torneios

x1	x7	x5
x2	x3	x7
x6	x4	x8
x8	x7	x1
x3	x1	x5
x3	x4	x2
x4	x2	x6
x7	x6	x4

Figura 3.8: Exemplo da Seleção por torneio [20].

- ***Steady State*:** De um total de n indivíduos que pertencerão a nova população, p destes serão gerados para substituir os piores da população corrente, sendo $p < n$. Neste processo, alguns elementos da população descendente poderão estar repetidos [20].

- ***Steady State* sem duplicados:** Esta técnica é semelhante à *Steady State*, com a diferença de que não é admitida a duplicação de indivíduos [20].

3.1.1.9.3 Cruzamento (*Crossover*)

Este operador é aplicado logo após a seleção e reprodução. O cruzamento combina partes do código genético de pares de indivíduos selecionados, sendo estes denominados pais. A partir deles são geradas novas cadeias de caracteres com grandes chances de superar a qualidade de ambos os pais. A partir dos pontos já visitados do espaço de busca o operador utiliza o conhecimento obtido por meio deles. Seu uso, geralmente, depende de uma taxa de probabilidade, onde esta é definida de acordo com o problema trabalhado [15, 20, 43, 45].

Cada indivíduo do par é dividido segundo um determinado critério, então dois novos indivíduos são gerados trocando caracteres entre os pontos de divisão das cadeias selecionadas. Desta forma o tamanho da população é sempre mantido e os indivíduos pais criam duas novas cadeias de caracteres. A seguir são apresentados os métodos mais comuns de cruzamento [15, 33, 45].

- **Cruzamento de um ponto:** É definida uma posição inteira k, com $1 \leq k \leq l$ e l sendo o tamanho da cadeia de caracteres. Cada indivíduo do par é dividido em duas partes segundo

o ponto de corte k, esta divisão utiliza uma seleção aleatória das posições da cadeia. Os fragmentos dos pais são trocados após este ponto e os filhos são gerados como pode ser visto na Figura 3.9 [15, 20, 33, 45].

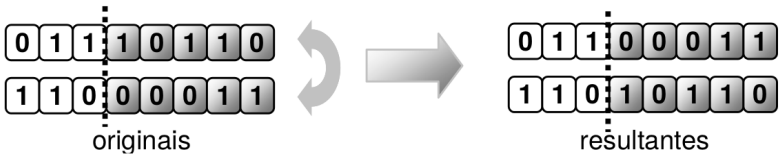


Figura 3.9: Exemplo de aplicação do Cruzamento de um ponto [45].

- **Cruzamento de múltiplos pontos:** Esta é uma forma genérica para o cruzamento de um ponto. São definidos, de forma aleatória, pontos de corte. Os intervalos de caracteres dos pais situados entre os pontos definidos são trocados, gerando os indivíduos descendentes. A Figura 3.10 exemplifica o mecanismo utilizando o método com dois pontos de corte [15, 20, 45].

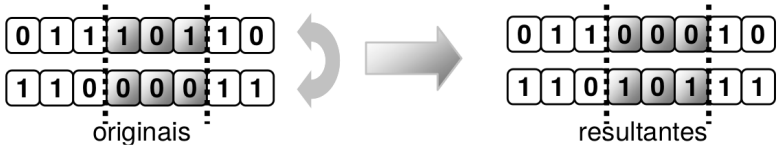


Figura 3.10: Exemplo de aplicação do Cruzamento de múltiplos pontos [45].

- **Cruzamento uniforme:** Aleatoriamente, uma máscara de dígitos binários é definida para configurar a geração dos novos indivíduos. O dígito 1 da máscara determina que, na referente posição, os caracteres dos pais devem ser trocados entre si. Já o dígito 0 define que, em tal posição, os valores originais dos pais são preservados. A Figura 3.11 apresenta um exemplo de como os filhos são gerados nessa forma de cruzamento [15, 20, 33, 45].

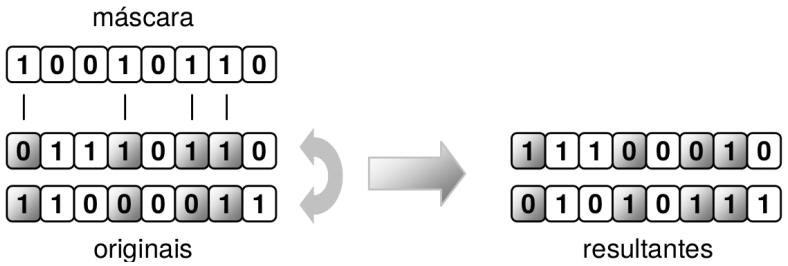


Figura 3.11: Exemplo de aplicação do Cruzamento uniforme [45].

A combinação entre a seleção e reprodução e o cruzamento não é complexa. Apesar da simplicidade, a reprodução de cadeias de alta qualidade e a troca de informações através do cruzamento estão no ponto central do Algoritmo Genético em descobrir boas soluções [15].

3.1.1.9.4 Mutação

Após os indivíduos passarem pelo cruzamento este operador é executado. A mutação efetua mudanças genéticas em alguns caracteres dos membros selecionados. Assim, permite-se que novas regiões do espaço de busca sejam exploradas por meio destes indivíduos, evitando que as soluções se prendam a uma zona de ótimo local [15, 20, 43, 45].

Assim como na etapa anterior, a mutação é aplicada de acordo com uma taxa de probabilidade fixa conforme o problema, em geral seu valor é bem pequeno, se esta for próxima a 100% a busca transforma-se em aleatória. Neste procedimento cada caractere da cadeia possui uma taxa de mutação, com valor suficiente para que se tenha diversidade entre os cromossomos da população, que determina a chance de uma alteração por outro valor aleatório do mesmo tipo e intervalo. Esta taxa não deve ser alta pois o modelo poderia ficar instável e seriam maiores as chances de um bom indivíduo ser destruído [15, 20, 33, 45].

Os operadores de seleção, reprodução e cruzamento são de grande importância para que sejam encontradas cadeias com alto valor de aptidão. Já o operador de mutação é útil para que seja possível alcançar outras cadeias de alta aptidão quando a população corrente está presa num ótimo local, convergindo apenas para ele. Dessa forma, quando utilizado com moderação, o operador consegue explorar novas regiões do espaço de busca proporcionando, assim, novas alternativas sem eliminar os indivíduos correntes de alta qualidade [15, 45].

Figura 3.12 é exemplificado o uso do operador mutação num indivíduo denotado por valores binários. Os caracteres alterados estão destacados na cadeia [45].

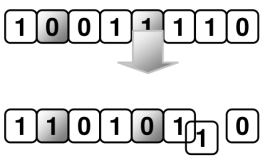


Figura 3.12: Exemplo de aplicação da Mutação [45].

4 OS PROCEDIMENTOS PROPOSTOS

Neste trabalho, a resolução do Problema de Agrupamento Automático é feita por meio da meta-heurística Algoritmo Genético. A forma de agrupamento adotada é a particional e a Distância Euclidiana determina a semelhança entre os objetos.

Os estudos que serão apresentados têm como base o método nomeado Algoritmo Evolutivo Construtivo com Busca Local1 (AECBL1), já conhecido na literatura. A heurística AECBL1 busca resolver o PCA a partir do conceito da meta-heurística Algoritmo Evolutivo. Ela possui duas fases, uma inicial que caracteriza a Etapa de Construção e outra denominada como Módulo Evolutivo. Esta última apresenta um algoritmo genético com busca local, onde noções de memória adaptativa são utilizadas para tentar aprimorar a representação de uma provável solução. Tem-se como objetivo aperfeiçoar tal algoritmo para se que sejam alcançadas soluções de melhor qualidade [11].

Durante o desenvolvimento deste trabalho, observou-se que duas questões influenciam diretamente a qualidade da solução de agrupamento e que seria necessário investigá-las a fim de melhorar as soluções finais obtidas. São elas: a geração de grupos iniciais e a função de avaliação adotada. Nas próximas seções tais pontos serão trabalhados.

4.1 Composição dos Algoritmos

Os parâmetros de entrada trabalhados são: o conjunto de dados X, o tamanho da população T_{pop}, o número de gerações G_{max} que serão executadas, a taxa de cruzamento p_c, a probabilidade de mutação p_m, o valor α utilizado na geração de grupos iniciais e os valores t e r, indicadores da frequência de aplicação das buscas locais Inversão Individual e Reconexão por Caminhos [11].

4.1.1 Representando soluções

Sejam $B = \{B_1, B_2, \ldots, B_p\}$ o conjunto de grupos iniciais produzido na fase de geração de grupos iniciais e x_i o centroide referente a cada grupo B_i, com $i = 1, 2, \ldots, p$. Uma solução é representada por meio de uma cadeia binária de tamanho p, com cada caractere simbolizando um dos respectivos grupos. Uma posição com valor 1 na cadeia indica um grupo pai, enquanto que um valor igual a 0 equivale a um grupo que é filho. A quantidade de pais definida é fixa em cada solução, deste modo para que a representação seja feita, até o fim do processamento todo grupo filho é agregado a algum pai. A junção é feita através da condição de menor distância entre centroides, em cada uma dessas recalcula-se o valor do centroide da nova formação. A Figura 4.1 traz, como exemplo, uma possível cadeia binária de 7 posições [11].

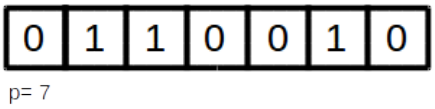


Figura 4.1: Exemplo de uma solução representada numa cadeia de 7 posições [11].

O conjunto de grupos $C = \{c_1, c_2, \ldots, c_k\}$ formado ao término do processamento é constituído pelos chamados grupos finais. A dinâmica da formação de cada solução sucede da seguinte forma [11]:

Considera-se uma cadeia binária gerada com p caracteres. A partir do conjunto B por ela retratado tem-se os subconjuntos $B^0 = \{B_1^0, B_2^0, \ldots, B_q^0\}$, que corresponde aos grupos representados pelo valor 0 na cadeia, e $B_1 = \{B_1^1, B_2^1, \ldots, B_k^1\}$, equivalente ao subconjunto de B com grupos simbolizados como 1, com $p = q + k$. Cada grupo B_j^1 equivale a um grupo final C_j, onde $j = 1, 2, \ldots, k$. Desta forma, deve-se obter um $B_r^1 \in B_1$ para cada $B_s^0 \in B^0$, com $r = 1, 2, \ldots, k$ e $s = 1, 2, \ldots, q$, tal que seus centroides cumpram a relação a seguir [11]:

$$d_e(\bar{c}_r^1\ , \bar{c}_s^0\ ) = \min\{d_e(\bar{c}_j^1\ , \bar{c}_s^0\ )\},\ j = 1, \ldots, k \tag{4.1}$$

Em cada verificação um B_s^0 é anexado a um B_r^1, recalcula-se o valor do centroide e atualiza-se o conjunto B_0. A processo encerra quando $B^0 = \emptyset$. Então, para cada $C_j \in C$ tem-se [11]:

$$C_j = \cup_{i=1}^{z} B_i, onde\ 1 \leq z \leq p \tag{4.2}$$

4.1.2 A geração de grupos iniciais

A Etapa de Construção do AECBL1 equivale a organização de um conjunto de grupos, a partir do conjunto de elementos X, que formará a solução inicial do algoritmo genético. Nesta fase, são construídos grupos provisórios a fim de diminuir a cardinalidade dos dados de entrada do problema, cada um destes grupos é considerado como um objeto pelo algoritmo. Pontos situados numa região densa dão origem a um grupo, assim ao invés de lidar com pontos unitários são manipulados conjuntos de pontos [11].

O procedimento original é composto por dois métodos que funcionam em conjunto: Gerar *Clusters* Parciais (GCP) e Junção de *Clusters* Parciais Adjacentes (JCPA). A partir de ambos foram desenvolvidas as diferentes formas de geração de grupos iniciais adotadas neste trabalho [11].

O número de grupos iniciais determina o tamanho do cromossomo que será adotado até

o fim da execução do algoritmo. O parâmetro α influencia diretamente a quantidade de grupos iniciais gerados. Quanto maior é o valor deste parâmetro menor será o número de grupos iniciais e mais rápida será a execução do algoritmo. Porém apesar da melhoria na velocidade a qualidade das soluções finais é consequentemente agravada. Quanto maior o número de grupos iniciais mais combinações são analisadas e maior é a chance de gerar boas soluções.

Desta forma, tornou-se necessário melhorar o processo de geração de grupos iniciais para que se consiga reduzir a cardinalidade dos dados de entrada do problema de forma eficiente para que não se perca em qualidade na solução final.

Foram estudadas cinco propostas de construção de grupos iniciais. A primeira adota parte da abordagem utilizada na Etapa de Construção do AECBL1. Já as propostas dois e três trazem uma nova estratégia a partir da segunda etapa desta Etapa de Construção. Alguns trabalhos da literatura consideram como fato que o número de grupos de uma solução costuma estar presente no intervalo de 2 a $\sqrt{n}$, onde n é o número de elementos da instância trabalhada. Como algumas metodologias atuam com a quantidade de grupos iniciais definida como k variando aleatoriamente entre estes dois valores ou como $k = \sqrt{n}$, o quarto e quinto procedimentos tomaram este fato como base para auxiliar em novas formas de se criar os grupos iniciais [23, 28, 29, 35, 47, 56]. Os cinco procedimentos são apresentados na sequência.

4.1.2.1 Procedimento Gerar Solução Inicial 1 (GSI1)

Neste procedimento é adotada apenas a metodologia utilizada no GCP do AECBL1. Para facilitar, este trabalho irá se referir a ele como GSI1, seu algoritmo é apresentado no Algoritmo 5 [11].

Num mesmo grupo são reunidos os elementos concentrados numa região densa. Define-se, para cada elemento a menor distância partindo dele a algum outro elemento. Então, é calculada a média de todas estas menores distâncias, que é chamada como d_{medio} [11].

A partir disto, cada elemento é definido como centro de um círculo cujo raio equivale a $r = \alpha * d_{medio}$, onde α é um parâmetro de entrada. Então, o grupo de elementos pertencentes a cada círculo $N_i = circulo(x_i; r)$ é definido. Este processo ocorre como no exemplo da Figura 4.2 [11].

Uma lista T armazena a quantidade de elementos de cada círculo. Ela é ordenada de modo decrescente de acordo com estas cardinalidades. Assim, os elementos correspondentes a cada posição de T são definidos os grupos iniciais deste procedimento, formando $B = \{B_1, B_2, \ldots, B_t\}$. Esses grupos não compartilham nenhum elemento pois, quando um círculo é escolhido os elementos que a ele pertencem não farão parte de nenhum outro [11].

Algoritmo 5: Algoritmo GSI1

Entrada: X, α

início
  para *i=1 até n* **faça**
    $d_{min}(x_i) = \min\{d_e(x_i, x_j)\}, i \neq j, 1 \leq j \leq n$;
  fim
  $d_{medio} = \frac{1}{n} \sum_{i=1}^{n} d_{min}(x_i)$;
  $r = \alpha * d_{medio}$;
  para *i=1 até n* **faça**
    $N_i = circulo(x_i, r)$;
    $T = T \cup N_i$;
  fim
  Ordenar T em ordem decrescente;
  $i = 1$;
  enquanto $T \neq \emptyset$ **faça**
    $B_i = proximo(N_j \in T)$;
    $T = T - N_j$;
    $i++$;
  fim
  Retornar $B = \{B_1, \ldots, B_t\}$, os t grupos iniciais;
fim

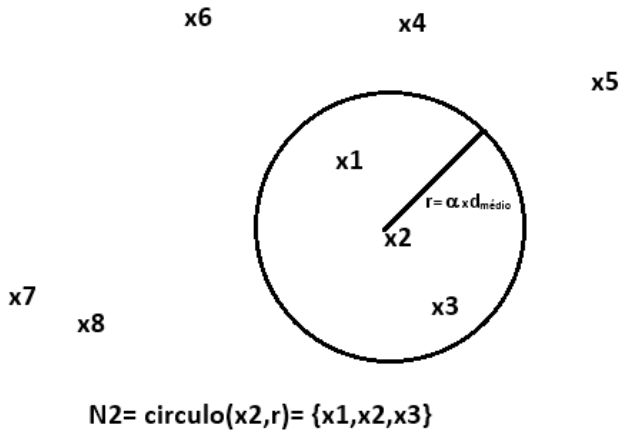


Figura 4.2: Cálculo dos elementos pertencentes ao círculo de centro x_2 e raio $r = \alpha * d_{medio}$ [11].

4.1.2.2 Procedimento Gerar Solução Inicial 2 (GSI2)

Este procedimento opera semelhante a Etapa de Construção do AECBL1. Com a cardinalidade dos dados reduzida após a execução do GSI1, o GSI2 busca diminuir ainda mais a

quantidade a ser processada. Por meio dele, são agregados os grupos parciais pequenos, com no máximo quatro elementos, próximos a outros que possuem mais elementos. O Algoritmo 6 retrata a metodologia de GSI2 [11].

Algoritmo 6: Algoritmo GSI2

Entrada: B

início
  Seja $B = \{B_1, \ldots, B_t\}$ o conjunto de grupos parciais gerado pelo GSI1;
  para *i=1 até t* **faça**
    se $cardinalidade(B_i) \leq 4$ **então**
      $B_k = menorDistanciaCentroide(B_i)$;
      $B_k = B_k \cup B_i$;
    fim
  fim
  Retornar $B = \{B_1, \ldots, B_p\}$ grupos iniciais, onde $p \leq t$;
fim

O mecanismo consiste em duas etapas. Primeiramente o GSI1 é aplicado sobre o conjunto de dados, daí é gerado um conjunto de grupos parciais. A partir destes grupos formados, são selecionados aqueles que possuem menos do que quatro componentes. Cada um destes une-se ao grupo mais próximo, onde a distância entre ambos centroides é a menor possível [11].

A partir das novas junções feitas um novo conjunto de grupos $B = \{B_1, B_2, \ldots, B_p\}$ é produzido, onde $p \leq t$. Assim, o GSI2 busca diminuir a quantidade de grupos resultantes do processo GSI1 [11].

4.1.2.3 Procedimento Gerar Solução Inicial 3 (GSI3)

A geração de grupos iniciais é bastante similar ao procedimento anterior. O agrupamento do conjunto inicial também é realizado em dois passos, sendo o primeiro deles o GSI1. A diferença na execução se deve ao fato que, no GSI3, apenas grupos com até dois elementos são selecionados para se unir a outros maiores. O método desta segunda etapa é apresentado no Algoritmo 7.

4.1.2.4 Procedimento Gerar Solução Inicial 4 (GSI4)

Esta metodologia atua de modo parecido ao GSI1, porém a criação dos grupos iniciais é limitada a exatamente $\sqrt{n}$. Dos círculos calculados são considerados um total de $\sqrt{n}$ dos que correspondem a regiões mais densas, estes dão origem aos grupos.

Algoritmo 7: Algoritmo GSI3

Entrada: B

início
  Seja $B = \{B_1, \ldots, B_t\}$ o conjunto de grupos parciais gerado pelo GSI1;
  para *i=1 até t* **faça**
    se $cardinalidade(B_i) \leq 2$ **então**
      $B_k = menorDistanciaCentroide(B_i)$;
      $B_k = B_k \cup B_i$;
    fim
  fim
  Retornar $B = \{B_1, \ldots, B_p\}$ grupos iniciais, onde $p \leq t$;
fim

Cada grupo é iniciado como unitário com um dos primeiros elementos da lista T ordenada, respectivamente. Tais elementos inicialmente são definidos como centroides de seus grupos. Em sequência, todo elemento que não pertence a nenhum destes grupos intermediários é alocado naquele que possui maior proximidade. Após todos os elementos estarem alocados num dos grupos o conjunto inicial $B = \{B_1, B_2, \ldots, B_t\}$ está formado. O procedimento é exposto no Algoritmo 8.

4.1.2.5 Procedimento Gerar Solução Inicial 5 (GSI5)

Este quinto cenário produz os grupos também em dois passos, iniciando com o procedimento GSI1. Após a execução deste, o conjunto de grupos iniciais terá a quantidade máxima de $\sqrt{n}$ grupos, de modo que os primeiros grupos construídos receberão, de acordo com a proximidade, os elementos alocados nos que foram gerados por último. A metodologia é apresentada no Algoritmo 9.

4.1.3 Algoritmo genético

A fase denominada Módulo Evolutivo no AECBL1 corresponde a noções de memória adaptativa junto a aplicação de três técnicas de busca local somadas a um Algoritmo Evolutivo, as buscas são: Inversão Individual, Troca entre Pares e Reconexão por Caminhos. O uso destas em conjunto com o algoritmo genético tradicional reforça a procura por boas soluções, possibilitando uma melhor qualidade nos resultados finais [11].

Como um dos objetivos ao se resolver um Problema de Agrupamento Automático é encontrar o número de grupos, de modo aleatório, o algoritmo genético possibilita que sejam gerados

Algoritmo 8: Algoritmo GSI4

Entrada: X, α

início
  para *i=1 até n* **faça**
    $d_{min}(x_i) = \min\{d_e(x_i, x_j)\}, i \neq j, 1 \leq j \leq n$;
  fim
  $d_{medio} = \frac{1}{n} \sum_{i=1}^{n} d_{min}(x_i)$;
  $r = \alpha * d_{medio}$;
  para *i=1 até n* **faça**
    $N_i = circulo(x_i, r)$;
    $T = T \cup N_i$;
  fim
  Ordenar T em ordem decrescente;
  $i = 1$;
  enquanto $i \leq \sqrt{n}$ **faça**
    $adicionaCentroCirculo(B_i, N_i), N_i \in T$;
    $i++$;
  fim
  $i = 1$;
  enquanto $T \neq \emptyset$ **faça**
    se $x_i \in T$ **então**
      $B_k = grupoMaisProximo(x_i)$;
      $atualiza(B_k, x_i, T)$;
    fim
    $i++$;
  fim
  Retornar $B = \{B_1, \ldots, B_t\}$, os t grupos iniciais;
fim

diferentes números de grupos pai por iteração [11].

O Algoritmo Genético adotado neste trabalho tem como base o procedimento original Módulo Evolutivo do AECBL1. Após o processamento de uma das propostas de Geração de Grupos Iniciais, o algoritmo começa a sua execução inicializando uma população. Em sequência, serão executadas G_{max} iterações referentes às gerações do Algoritmo Genético [11].

A cada geração são selecionados indivíduos para as operações de reprodução, cruzamento e mutação. Os pares selecionados são definidos da seguinte forma: o primeiro é escolhido dentre os 50% mais aptos e o segundo dentre toda a população, ambos selecionados de forma aleatória. Não há repetições na escolha dos indivíduos. O número de pares que passará pelo cruzamento

Algoritmo 9: Algoritmo GSI5

Entrada: B

início
  Seja $B = \{B_1, \ldots, B_t\}$ o conjunto de grupos parciais gerado pelo GSI1;
  se $t > \sqrt{n}$ **então**
    para $i = \sqrt{n}$ *até* t **faça**
      para $j = 1$ *até* $|B_i|$ **faça**
        $B_k = grupoMaisProximo(x_j)$;
        $atualiza(B_k, B_i, x_j)$;
      fim
    fim
  fim
  Retornar $B = \{B_1, \ldots, B_p\}$ grupos iniciais, onde $p \leq t$;
fim

é definido de acordo com uma taxa p_c, a operação é aplicada no método de dois pontos. Logo após o operador de mutação é aplicado com uma probabilidade p_m, ele efetua a troca de um dos caracteres de um dos indivíduos do par. Se algum indivíduo gerado representa uma configuração inválida, sem grupos, outro é gerado aleatoriamente para substitui-lo. Lembrando que, ao gerar algum cromossomo, seja por aplicação de um operador ou durante um processo de busca, ele é avaliado pela função de aptidão [11].

A busca local Inversão Individual é aplicada nos indivíduos mais aptos da população a cada t iterações. Já a busca local Reconexão por Caminhos é executada a cada r iterações sobre o melhor indivíduo da população e o melhor do conjunto Elite [11].

O conjunto elite é definido para um tamanho de cinco indivíduos e guarda a melhor solução em cada iteração, esta deve ser melhor do que a pior solução presente neste conjunto e diferente de todas as demais, e no final do processamento este conjunto passa pela busca Troca entre Pares [11].

Concluindo, o algoritmo retorna a melhor de todas as soluções após concluir o processamento das operações. O Algoritmo 10 apresenta o funcionamento da metodologia [11].

4.1.4 Memória adaptativa

Este conceito trabalha com as melhores soluções alcançadas no processamento, o conjunto por elas formado é atualizado no decorrer da execução do algoritmo [11].

Neste trabalho, em cada iteração é armazenada a solução de maior qualidade no denomi-

Algoritmo 10: Algoritmo AGBL

Entrada: $X, T_{pop}, G_{max}, p_c, p_m, \alpha, t, r$

início
  $G = gerarGruposIniciais(X, \alpha)$;
  $P = gerarPopulacaoInicial(G, T_{pop})$;
  para $k = 1$ *até* G_{max} **faça**
    para $i = 1$ *até* p_c **faça**
      $seleciona(p_1, p_2)$;
      $cruzamento(p_1, p_2)$;
      $mutacao(p_2, p_m)$;
      $verificarIndividuos(p_1, p_2)$;
      $avaliarSolucoes(p_1, p_2)$;
    fim
    se $k\%t = 0$ **então**
      $inversaoIndividual(P)$;
    fim
    se $k\%r = 0$ **então**
      $reconexaoCaminhos(P, elite)$;
    fim
    $atualizarConjuntoElite(P, elite)$;
  fim
  $trocaEntrePares(elite)$;
  Retornar a melhor solução S;
fim

nado conjunto Elite, as soluções presentes nele são distintas. O algoritmo faz uso deste conjunto em duas situações a fim de tentar aumentar a qualidade das soluções, são elas: no meio de sua execução, ao aplicar a Reconexão por Caminhos, e no final, no uso da Troca Entre Pares [11].

4.1.5 As buscas locais

4.1.5.1 Inversão Individual

Este método busca melhorar a solução corrente por meio de análises das soluções de sua vizinhança. Cada posição da cadeia da solução tem seu valor trocado, ou seja, um caractere representado por 1 se transforma em 0 e vice-versa. Esta permutação acontece com a troca de uma posição de cada vez, assim, a cada mudança é calculado com a função de avaliação o valor alcançado pela solução recém-gerada. O algoritmo atualiza a solução se o valor obtido pela função

for melhor do que o anterior, caso contrário a solução é mantida. Encerra-se a busca após a troca de todos os caracteres da cadeia que simboliza a solução [11].

Como uma das metas do problema é que se descubra o número ideal de grupos, a Inversão Individual contribui para que isto ocorra pois incluir ou retirar um grupo pai pode resultar numa solução corrente de melhor qualidade. O exemplo da Figura 4.3 ilustra o funcionamento desta busca [11].

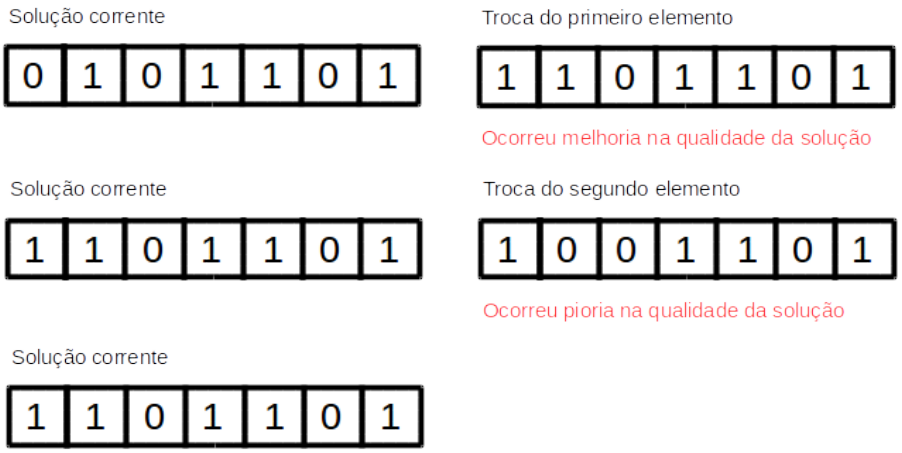


Figura 4.3: Exemplo de parte do funcionamento da busca Inversão Individual [11].

4.1.5.2 Troca Entre Pares

Nesta busca são trocados, quando diferentes, os valores de duas posições da cadeia que representa a solução. A cada troca calcula-se para a solução recém-gerada o valor obtido pela função de avaliação. Quando há melhoria na qualidade da solução ela é admitida e se torna a nova solução corrente, do contrário a anterior se mantém. Depois de investigar todas as possíveis trocas entre dois elementos o processo chega ao fim [11].

A Troca Entre Pares tem o intuito de descobrir soluções de melhor qualidade sem que a quantidade de grupos pais obtida anteriormente seja mudada. Na Figura 4.4 é exposto um exemplo de parte do funcionamento desta busca [11].

4.1.5.3 Reconexão por Caminhos

Esta é uma metodologia que busca, entre duas soluções limitantes, por soluções intermediárias de boa qualidade. A Reconexão por Caminhos se baseia na ideia que, entre duas boas soluções há uma chance de que exista uma terceira com qualidade superior que a das outras [11].

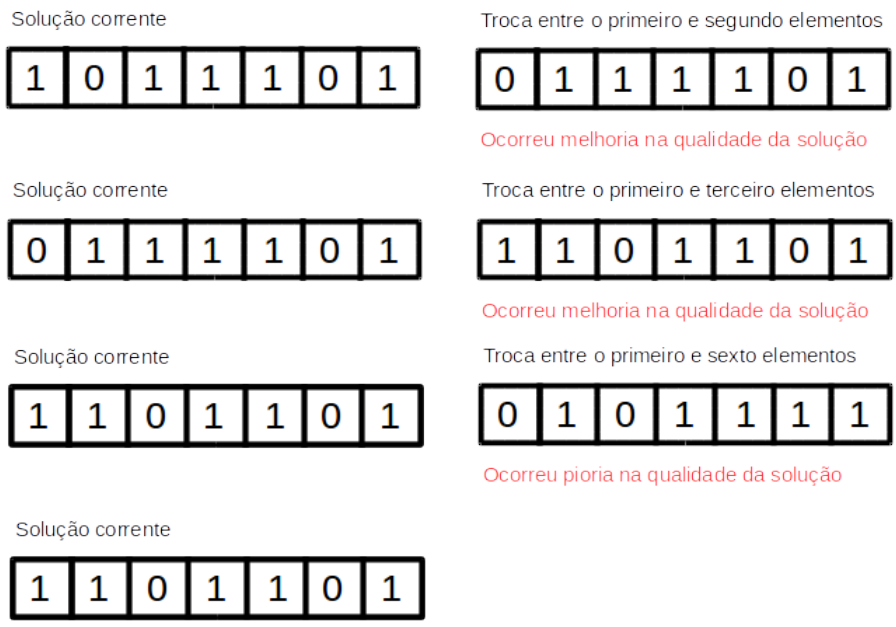


Figura 4.4: Exemplo de parte do funcionamento da busca Troca Entre Pares [11].

Um caminho é demarcado entre uma solução base e uma destino, ambas com um bom valor de aptidão, e ao percorrer o trajeto as soluções intermediárias geradas são analisadas. Busca-se encontrar soluções com qualidade melhor do que as outras duas que limitam durante o percurso. Neste trabalho a trajetória adotada considera um sentido que inicia na solução de melhor qualidade e termina na de pior. Um fragmento da solução de qualidade mais baixa é inserido na de qualidade maior. Insere-se primeiro a última posição da cadeia do pior na referente posição do melhor. Na segunda tentativa são substituídos na melhor solução a penúltima e última posições. A investigação segue do mesmo modo até a troca de todas as posições entre as soluções. Ao finalizar a execução a melhor solução alcançada ao longo do processamento é retornada [11].

Dado que achar a quantidade ideal de grupos é um dos pontos que se deseja alcançar, buscar entre duas boas soluções, com distintas quantidades de grupos pai, por outras soluções com diferentes números de pais pode ser uma boa estratégia na resolução do problema. Na Figura 4.5 é exemplificado o início do funcionamento da Reconexão por Caminhos [11].

4.2 Adotando uma Função de Avaliação

A função de avaliação adotada pela heurística AECBL1 é a função Índice Silhueta. Inicialmente o uso desta função foi mantido para realizar as verificações das diferentes abordagens para

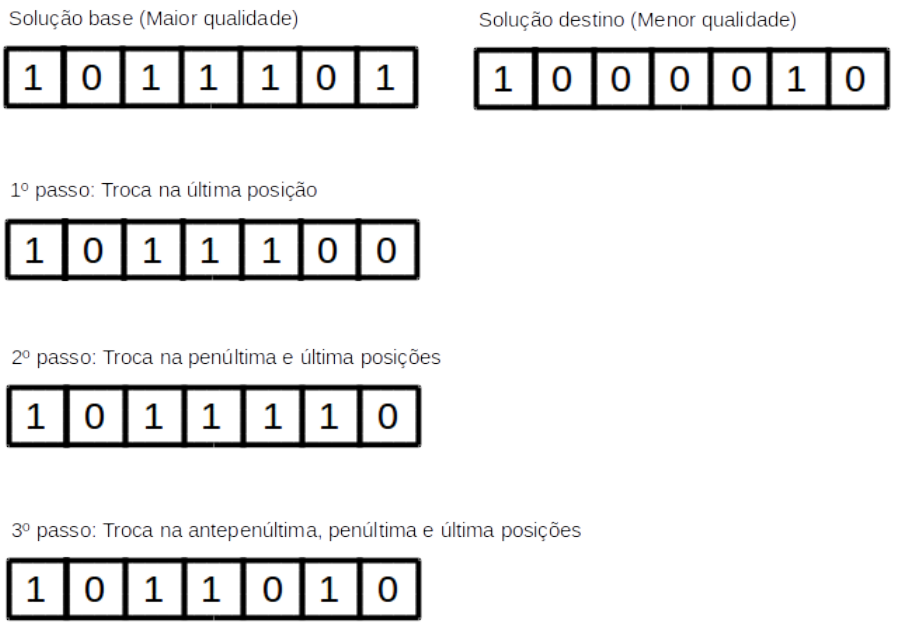


Figura 4.5: Exemplo de parte do funcionamento da busca Reconexão por Caminhos [11].

a geração de grupos iniciais. Porém sabia-se da necessidade de estudar a utilização de uma outra, pois com outro critério de avaliação a qualidade das soluções finais poderia melhorar. Assim, após analisar qual abordagem inicial obteve melhor êxito o algoritmo adota o seu uso em conjunto com uma nova função de avaliação [11].

4.2.1 Índice silhueta

Foi observado empiricamente que, a função Índice Silhueta se comporta de modo menos eficiente quando num conjunto de dados os elementos possuem uma proximidade muito alta, influenciando, assim, na solução final. Desse modo, tornou-se necessário investigar, na literatura, algumas funções de avaliação para que fosse possível ajustar os grupos de uma melhor forma.

A partir dos primeiros experimentos, que serão apresentados posteriormente, observou-se que muitas vezes com o uso desta função as soluções conseguem alcançar valores de avaliação altos porém com um número de grupos que não é coerente com a realidade.

4.2.2 Índice CH

Alguns trabalhos apresentam propostas para a resolução do Problema de Agrupamento Automático utilizando o Índice CH como função de avaliação. Afirma-se que o uso de tal índice é uma boa escolha na busca por soluções com boas formações de grupos pois, além de ser simples realizar a sua implementação o seu processamento não é muito custoso computacionalmente. De modo geral, seus resultados são robustos comparado a outros métodos de validação de grupos [23, 29].

Assim, numa segunda parte do estudo, foi adotada como avaliação a função Índice CH juntamente com a abordagem de geração de grupos iniciais que obteve melhor êxito. O resultado final continuará sendo apresentado através da função Índice Silhueta, porém ela irá apenas calcular seu valor para a solução gerada. A avaliação de qualidade dos indivíduos durante todo o processo será feita pela função CH. Para que possam ser feitas comparações, o resultado final não apresentará apenas os valores de CH devido a escassez de trabalhos que a utilizem.

5 EXPERIMENTOS COMPUTACIONAIS

Neste trabalho foram realizados experimentos com o intuito de analisar a qualidade das soluções dos algoritmos propostos. Os resultados obtidos serão apresentados neste capítulo. Nas seções seguintes serão melhor detalhados o equipamento utilizado, parâmetros de entrada, os cenários de testes, dentre outros pontos.

Primeiro, serão comparadas as propostas de Algoritmo Genético que utilizam diferentes estratégias para a geração de grupos iniciais avaliando os agrupamentos com a função Índice Silhueta. Estas comparações irão apontar qual versão da meta-heurística alcançou melhores resultados.

Na sequência a versão indicada será comparada com uma outra que fará uso da mesma metodologia de geração de grupos iniciais porém com a função de avaliação Índice CH.

Por fim, estas duas serão comparadas com os métodos dos trabalhos da literatura escolhidos, que também resolvem o Problema de Agrupamento Automático e utilizam a função Índice Silhueta para mensurar a qualidade final de cada solução. Estes algoritmos apresentam resultados com valores numéricos, o que tornou possível a realização das comparações, que incluem o número de grupos e o valor da função Silhueta de suas resoluções.

5.1 Materiais e Ambiente

Para a realização dos experimentos foi utilizado o equipamento com a especificação abaixo.

- **Laptop:**
  - Samsung ATIV Book 2 (270E4E-KD5)
  - Intel Core i3-3110M
  - 4 GB de RAM
  - Linux (Ubuntu 14.04) *kernel* 3.16.0-50

Como os experimentos para buscar melhores resultados de agrupamento se basearam na técnica AECBL1 a codificação foi feita a partir da implementação disponibilizada da referida metodologia, na linguagem de programação C++ [11]. Foi utilizado o compilador g++ na versão 4.8.4. A seguir serão descritos os valores utilizados como parâmetros de entrada no algoritmo.

- **Parâmetro α:** Valores testados variando entre $0, 5$ e 12, de acordo com as particularidades do conjunto de dados.

- T_{pop}**:** O tamanho da população foi definido como $1/3$ do número de grupos iniciais gerados, ou seja, $1/3$ do tamanho do cromossomo. Seu valor é limitado a um máximo de 30 indivíduos.
- **Parâmetro** G_{max}**:** O algoritmo teve o número de gerações estipulado como 50.
- **Parâmetro** p_c**:** O número de pares de indivíduos selecionados à operação de cruzamento equivale a 40% do tamanho da população.
- **Parâmetro** p_m**:** A chance de um indivíduo selecionado sofrer a operação de mutação foi fixada em 10%.
- **Parâmetro** t**:** Foi estipulado para que a cada cinco iterações ocorresse a sua aplicação. Deste modo, para 20% dos melhores indivíduos da população a busca é executada nas seguintes gerações: 5, 10, 15, 20, 25, 30, 35, 40, 45 e 50. O operador é aplicado de forma semelhante ao algoritmo AECBL1, tendo a mesma frequência definida.
- **Parâmetro** r**:** A aplicação da busca local Reconexão de Caminhos foi definida para as seguintes gerações: 18, 28, 38 e 48. O AECBL1 executa a busca desta forma e decidiu-se mantê-la igualmente.

Para identificar qual procedimento inicial e função objetivo o algoritmo AGBL está utilizando cada proposta é denominada de acordo com o Quadro 5.1.

Quadro 5.1: Propostas apresentadas para a resolução do PCA.

	Procedimento inicial	**Função de avaliação**
AGBL1	GSI1	Índice Silhueta
AGBL2	GSI2	Índice Silhueta
AGBL3	GSI3	Índice Silhueta
AGBL4	GSI4	Índice Silhueta
AGBL5	GSI5	Índice Silhueta
AGBL6	GSI1	Índice CH

5.1.1 Conjuntos de dados utilizados

Diversos tipos de conjuntos de dados foram utilizados, sendo alguns gerados artificialmente e outros originários de dados reais. Os dados manuseados nos testes são todos numéricos.

Posteriormente, junto aos resultados de cada categoria de experimentos serão especificadas as instâncias utilizadas, cada qual pertencendo a uma das coleções que serão apresentadas na sequência [1, 2].

- ***UCI datasets:*** As seis instâncias a seguir são bastante conhecidas na literatura e são derivadas de informações reais [1, 2].

  - **Iris:** Instância composta por 150 elementos que organizam-se em três grupos. Cada um corresponde a um tipo da planta íris e é formado por 50 elementos. Esse conjunto possui quatro dimensões [1, 2].
  - **Wine:** Este conjunto de dados é formado por 178 elementos com 13 dimensões que configuram três grupos. Os dados originam de uma análise química de vinhos que foram cultivados numa mesma região italiana, provenientes de três diferentes cultivares. Esta análise definiu as quantidades de 13 componentes encontrados em cada um dos três tipos de vinho [1, 2].
  - **Yeast:** O conjunto de dados é composto por 1484 elementos com oito dimensões gerando dez grupos. Este conjunto prediz os pontos de localização celular de proteínas [1, 2].
  - **Glass:** Com tamanho de 214 elementos, este conjunto de dados possui nove dimensões e configura sete grupos. São classificados os tipos de vidro a partir de seu teor de óxido. Este estudo foi impulsionado por investigações criminológicas [1, 2].
  - **Thyroid:** Instância de cinco dimensões formada por 215 elementos que se dispõem em três grupos. Proveniente de testes utilizados para prever o diagnóstico da tireoide de pacientes [1, 2].
  - **Breast:** Os 699 elementos desta instância possuem nove dimensões e se organizam em dois grupos. A base de dados auxilia no diagnóstico de câncer de mama [1, 2].

- ***DIM-sets (high):*** Os conjuntos de dados abaixo possuem alta dimensão, cada qual com tamanho de 1024 elementos. Foram criados artificialmente tendo como fim validar, com dados de dimensões maiores, a escalabilidade da abordagem proposta. Sabe-se que seus elementos configuram 16 grupos [1, 2, 29].

  - **Dim32:** Instância constituída por 32 dimensões [1].
  - **Dim64:** Instância constituída por 64 dimensões [1].
  - **Dim128:** Instância constituída por 128 dimensões [1].
  - **Dim256:** Instância constituída por 256 dimensões [1].

- ***A-sets:*** Estas três instâncias, geradas artificialmente, possuem duas dimensões. Sua representação gráfica é apresentada na Figura 5.1. Para cada uma destas, configura-se a existência de 150 elementos por grupo [1].

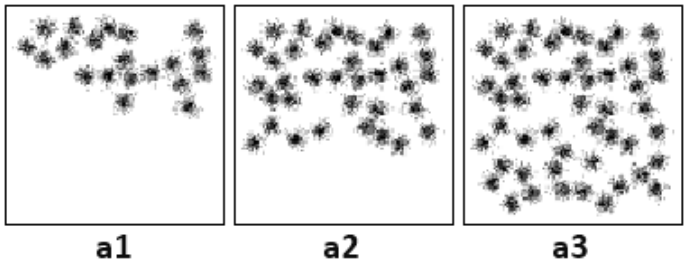


Figura 5.1: Os conjuntos de dados de A-sets [1].

  - **A1:** Instância formada por 3000 elementos. Em sua composição destacam-se 20 grupos [1].
  - **A2:** Instância formada por 5250 elementos. Em sua composição destacam-se 35 grupos [1].
  - **A3:** Instância formada por 7500 elementos. Em sua composição destacam-se 50 grupos [1].

- ***Shape sets:*** R15 e D31 são conjuntos de dados, também sintéticos, de duas dimensões que geram grupos Gaussianos semelhantes. Ambos são ilustrados na Figura 5.2 [1, 59].

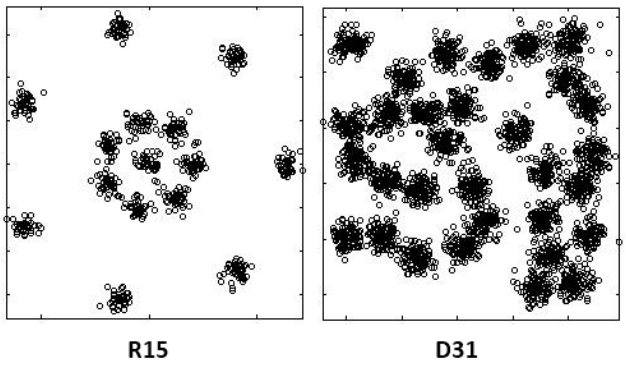


Figura 5.2: Os conjuntos de dados de Shape sets [1].

  - **R15:** Instância formada por 600 elementos. Configura-se como 15 grupos que se dispõem em anéis [1, 59].

- **D31:** Instância formada por 3100 elementos. Configura-se como 31 grupos, cada qual formado por 100 elementos [1, 59].

- **Comportadas e Não comportadas:** Estes conjuntos de dados foram construídos artificialmente através de uma ferramenta gráfica. Possuem diferentes tamanhos, variando de 100 a 2000 elementos. Todas as instâncias pertencem ao $\mathbb{R}^2$ e têm números de grupos que variam entre dois e 27. O nome de cada conjunto de dados é representado de acordo com sua quantidade de elementos e número de grupos. Existem dois tipos de instâncias que serão apresentadas a seguir [11]:

  - **Comportadas:** Em cada um destes conjuntos de dados os grupos são bem definidos e separados. São nomeados com uma letra "c" no final, uma instância chamada 300p4c, por exemplo, é formada por 300 elementos e quatro grupos [11].
    - **100p3c:** Conjunto de dados com 100 elementos que configura três grupos [11].
    - **100p7c:** Conjunto de dados com 100 elementos que configura sete grupos [11].
    - **100p10c:** Conjunto de dados com 100 elementos que configura dez grupos [11].
    - **200p4c:** Conjunto de dados com 200 elementos que configura quatro grupos [11].
    - **300p3c:** Conjunto de dados com 300 elementos que configura três grupos [11].
    - **400p3c:** Conjunto de dados com 400 elementos que configura três grupos [11].
    - **500p3c:** Conjunto de dados com 500 elementos que configura três grupos [11].
    - **600p15c:** Conjunto de dados com 600 elementos que configura 15 grupos [11].
    - **700p4c:** Conjunto de dados com 700 elementos que configura quatro grupos [11].
    - **800p23c:** Conjunto de dados com 806 elementos que configura 23 grupos [11].
    - **900p5c:** Conjunto de dados com 900 elementos que configura cinco grupos [11].
    - **900p12c:** Conjunto de dados com 900 elementos que configura 12 grupos [11].
    - **1000p6c:** Conjunto de dados com 1000 elementos que configura seis grupos [11].
    - **1000p14c:** Conjunto de dados com 1000 elementos que configura 14 grupos [11].
    - **1300p17c:** Conjunto de dados com 1300 elementos que configura 17 grupos [11].
    - **1800p22c:** Conjunto de dados com 1800 elementos que configura 22 grupos [11].
    - **2000p11c:** Conjunto de dados com 2000 elementos que configura 11 grupos [11].

– **Não comportadas:** Os conjuntos de dados deste tipo possuem muitos elementos localizados entre os grupos. Seus nomes possuem um caractere "1" no final, então uma instância chamada 100p5c1 possui 100 elementos e cinco grupos. A quantidade de grupos é aproximada devido a imprecisão na organização dos dados [11].

  * **100p2c1:** Conjunto de dados com 100 elementos que configura dois grupos [11].
  * **100p3c1:** Conjunto de dados com 100 elementos que configura três grupos [11].
  * **100p5c1:** Conjunto de dados com 110 elementos que configura cinco grupos [11].
  * **100p7c1:** Conjunto de dados com 112 elementos que configura sete grupos [11].
  * **200p2c1:** Conjunto de dados com 200 elementos que configura dois grupos [11].
  * **200p3c1:** Conjunto de dados com 200 elementos que configura três grupos [11].
  * **200p4c1:** Conjunto de dados com 200 elementos que configura quatro grupos [11].
  * **200p7c1:** Conjunto de dados com 210 elementos que configura sete grupos [11].
  * **200p12c1:** Conjunto de dados com 222 elementos que configura 12 grupos [11].
  * **300p2c1:** Conjunto de dados com 300 elementos que configura dois grupos [11].
  * **300p3c1:** Conjunto de dados com 300 elementos que configura três grupos [11]..
  * **300p4c1:** Conjunto de dados com 300 elementos que configura quatro grupos [11].
  * **300p6c1:** Conjunto de dados com 300 elementos que configura seis grupos [11].
  * **300p13c1:** Conjunto de dados com 235 elementos que configura 13 grupos [11].
  * **400p4c1:** Conjunto de dados com 400 elementos que configura quatro grupos [11].
  * **400p17c1:** Conjunto de dados com 408 elementos que configura 17 grupos [11].
  * **500p4c1:** Conjunto de dados com 500 elementos que configura quatro grupos [11].
  * **500p6c1:** Conjunto de dados com 500 elementos que configura seis grupos [11].
  * **600p3c1:** Conjunto de dados com 600 elementos que configura três grupos [11].
  * **700p15c1:** Conjunto de dados com 703 elementos que configura 15 grupos [11].
  * **800p4c1:** Conjunto de dados com 800 elementos que configura quatro grupos [11].
  * **800p10c1:** Conjunto de dados com 800 elementos que configura dez grupos [11].
  * **800p18c1:** Conjunto de dados com 800 elementos que configura 18 grupos [11].

* **1000p5c1:** Conjunto de dados com 1000 elementos que configura cinco grupos [11].
* **1000p27c1:** Conjunto de dados com 1005 elementos que configura 27 grupos [11].
* **1100p6c1:** Conjunto de dados com 1100 elementos que configura seis grupos [11].
* **1500p6c1:** Conjunto de dados com 1500 elementos que configura seis grupos [11].
* **2000p9c1:** Conjunto de dados com 2000 elementos que configura nove grupos [11].

– **Outras instâncias:**

* **Ruspini:** Conjunto de dados bidimensional com 75 elementos que configura quatro grupos [11, 29, 53].
* **Maronna:** Conjunto de dados de duas dimensões onde 200 elementos configuram um total de quatro grupos [11, 38].
* **200DATA:** Com tamanho de 200 elementos, esta instância configura quatro grupos para dados com duas dimensões [11, 18].
* **Broken ring:** Um total de 800 elementos deste conjunto de dados que possui duas dimensões se dispõe em cinco grupos [11, 60].

5.2 Resultados e Discussão

Na sequência serão apresentadas as resoluções de todas as propostas executadas neste trabalho e comparações de algumas destas com metodologias da literatura.

Durante a execução dos experimentos foi observado que, para alguns cenários a execução dos algoritmos tornava-se demasiado longa. Tal fato é acarretado pelo tamanho de certos conjuntos de dados e pelo alto número de grupos iniciais gerados. Assim, pôde ser visto empiricamente que, o algoritmo genético tem um processamento mais lento a medida que o valor de α é reduzido. Desta forma, o indivíduo que representa a solução assume um tamanho maior fazendo com que seja mais vasto o número de possíveis associações entre os elementos. Logo, cada escolha do valor deste parâmetro também levou em consideração o tempo de processamento praticado. Este é calculado em segundos quando apresentado nos resultados. Quando denotado como zero significa que o tempo médio de processamento demorou menos do que um segundo.

As soluções finais foram avaliadas através de valores Silhueta em todas as estratégias para efetuar as comparações, haja vista que muitos trabalhos na literatura apresentam a avaliação final

de seus agrupamentos através desta função. Em cada quadro os maiores resultados Silhueta estarão destacados em negrito.

5.2.1 Comparação entre os procedimentos para a geração de grupos iniciais

Os resultados que serão apresentados no Quadro 5.2 comparam as qualidades das soluções obtidas com o uso de cada um dos procedimentos propostos para a geração de grupos iniciais. A coluna "Conj. dados" indica o nome de cada conjunto de dados. As colunas "AGBL1", "AGBL2", "AGBL3", "AGBL4" e "AGBL5" se referem aos resultados finais obtidos com a utilização de cada um dos cinco procedimentos propostos. A coluna denominada "Liter." corresponde às informações conhecidas na literatura. Cada conjunto de dados tem seus resultados apresentados em três linhas, uma denominada "Núm. grupos" referente a quantidade de grupos gerada na solução final, outra "Aptidão" equivalente ao resultado alcançado pela função de avaliação Índice Silhueta e uma nomeada "Tempo" correspondente ao custo de processamento para a instância. Este quadro refere-se aos experimentos executados para conjuntos de dados apresentados das seguintes coleções: UCI *datasets*, DIM-*sets* (*high*), A-*sets*, *Shape sets* e Outras instâncias.

Quadro 5.2: Resultados obtidos com o uso de diferentes procedimentos para a geração de grupos iniciais. *(continua)*

Conj. dados		AGBL1	AGBL2	AGBL3	AGBL4	AGBL5	Liter.
ruspini	Núm. grupos	69	4	4	4	4	4
	Aptidão	**0,926141**	0,737657	0,737657	0,737657	0,737657	
	Tempo	0,4	0,2	0	0	0	
iris	Núm. grupos	143	2	2	2	2	3
	Aptidão	**0,971475**	0,686393	0,686393	0,686393	0,686393	
	Tempo	3,2	0	0	0,2	0	
wine	Núm. grupos	124	2	2	2	3	3
	Aptidão	**0,818327**	0,621128	0,621128	0,606042	0,629785	
	Tempo	13	0	0	0,4	0,2	
glass	Núm. grupos	154	2	2	4	2	7
	Aptidão	**0,709138**	0,614656	0,613216	0,577913	0,656562	
	Tempo	13,8	0,8	0,2	1	0,6	
thyroid	Núm. grupos	187	2	2	2	2	2
	Aptidão	**0,848463**	0,638445	0,645883	0,510998	0,67316	
	Tempo	9,6	0,4	0,4	0,6	0,4	
breast	Núm. grupos	243	2	2	2	2	2
	Aptidão	**0,678474**	0,596185	0,596186	0,591643	0,559964	
	Tempo	764,4	144,2	107,2	11,4	6,6	

Quadro 5.2. *Continuação*

Conj. dados		AGBL1	AGBL2	AGBL3	AGBL4	AGBL5	Liter.
yeast	Núm. grupos	2	2	2	2	2	10
	Aptidão	**0,611731**	0,591896	0,597582	0,542819	0,592413	
	Tempo	15,4	91,2	122,8	86,2	84,6	
dim32	Núm. grupos	16	16	16	6	16	16
	Aptidão	**0,945562**	**0,945562**	**0,945562**	0,356	**0,945562**	
	Tempo	92,6	41,2	41	136,6	95,8	
dim64	Núm. grupos	16	16	16	6	16	16
	Aptidão	0,966338	**0,966339**	**0,966339**	0,300176	0,966338	
	Tempo	72,8	100,6	347,2	249,4	73	
dim128	Núm. grupos	16	16	16	30	16	16
	Aptidão	**0,974642**	**0,974642**	**0,974642**	0,216036	**0,974642**	
	Tempo	181,6	141,6	134	358,8	177,2	
dim256	Núm. grupos	16	16	16	6	16	16
	Aptidão	0,982946	**0,982947**	0,982946	0,324823	0,982946	
	Tempo	277,6	2783,4	284,8	898	287,2	
R15	Núm. grupos	511	15	15	14	15	15
	Aptidão	**0,900603**	0,751731	0,750624	0,629379	0,745127	
	Tempo	166	54	2	3,6	3	
D31	Núm. grupos	31	31	31	26	28	31
	Aptidão	**0,572864**	0,568617	0,564459	0,514079	0,477901	
	Tempo	3858,2	2119,6	1344,6	391,6	399,8	
a1	Núm. grupos	20	20	20	19	20	20
	Aptidão	**0,593601**	0,592506	0,59314	0,54957	0,55823	
	Tempo	1661,2	1078,6	1138,6	340,2	377,8	
a2	Núm. grupos	35	35	35	29	2	35
	Aptidão	**0,595603**	0,594142	0,590559	0,524645	0,538552	
	Tempo	10342,4	6791,6	6097,2	1840,8	1226,8	
a3	Núm. grupos	50	50	50	31	54	50
	Aptidão	**0,593568**	0,591038	0,59184	0,43842	0,494165	
	Tempo	24514,4	9812,4	11435,8	4732	5232,6	

Nesta etapa da pesquisa foram estudadas as metodologias responsáveis pela geração dos grupos iniciais no algoritmo genético. Tem-se o objetivo de avaliar se as particularidades definidas em cada proposta foram eficazes, preservando o número correto de grupos e altos valores de aptidão para as soluções, com um tempo de processamento pouco custoso. Desta forma, as comparações entre seus resultados foram testadas em 16 instâncias, tendo em vista a maximização da função objetivo. Além de observar o número de grupos gerado foi analisada a aptidão alcançada

pela função. Este valor de avaliação originou-se por meio da própria função Índice Silhueta e os experimentos consideraram apenas os maiores valores obtidos pela função para cada instância.

No Quadro 5.2 pode ser observado que, com relação a todos os conjuntos de dados analisados, em mais de 80% dos casos pelo menos um dos algoritmos conseguiu encontrar o número correto de grupos em sua resolução. Nenhum dos procedimentos obteve o número de grupos correto nos experimentos que consideraram as seguintes instâncias: iris, glass e yeast. Apesar disto, números razoavelmente próximos foram encontrados. O AGBL1 alcançou a quantidade certa de grupos num total de oito vezes. Já os procedimentos AGBL2 e AGBL3 obtiveram 12 vezes o número de grupos correto. Somente para três conjuntos de dados o algoritmo AGBL4 conseguiu acertar o número correto de grupos. E o AGBL5 obteve para dez instâncias a quantidade certa de grupos.

Considerando o custo na computação, as propostas AGBL1 e AGBL2 obtiveram processamento igual ou melhor do que as outras num total de três vezes. Já AGBL3 atingiu velocidade equivalente ou menor do que outros procedimentos em oito situações. O processamento de AGBL4 foi igual ou melhor para apenas quatro conjuntos de dados. E AGBL5 obteve velocidade de computação igual ou superior do que os outros cinco vezes.

Apesar da metodologia de AGBL1 não ser a menos custosa dentre as cinco a sua resolução ganhou/empatou 14 vezes com relação aos valores silhueta das outras técnicas. Suas resoluções perderam apenas nos conjuntos de dados dim64 e dim256, no primeiro caso sendo 0,000001 menor em relação a AGBL2 e AGBL3, e no segundo perdendo apenas para AGBL2 também por 0,000001. Do total de 16 instâncias testadas, em nove casos a diferença entre os dois maiores valores silhueta dos procedimentos foi maior do que 0,02.

Os resultados de avaliação de AGBL2 e AGBL3 são bem similares. A maior diferença encontrada entre os dois não ultrapassa o valor 0,0075. Em geral, as silhuetas de ambos têm uma considerável proximidade das obtidas pelo AGBL1. Num total de oito situações a diferença entre o AGBL1 e um desses foi inferior a 0,01. Tal semelhança pode estar relacionada ao fato que as três propostas possuem uma etapa em comum no procedimento de geração de grupos iniciais. No que refere a AGBL2 e AGBL3 a proximidade de valores é ainda maior visto que toda a metodologia de formação dos grupos iniciais é igual com exceção pela definição de quais grupos parciais pequenos que são agregados em cada proposta.

No que se refere a AGBL4, seus resultados silhueta foram pouco satisfatórios. Em geral, foi grande a diferença entre as aptidões do AGBL4 quanto aos outros algoritmos, ele não conseguiu superar às resoluções do AGBL1 em nenhum dos conjuntos de dados. As silhuetas obtidas pelo procedimento empataram com AGBL2, AGBL3 e AGBL5 apenas 2 vezes. A proposta na qual AGBL4 alcança alguma proximidade é a AGBL5, conseguindo obter um melhor resultado para duas instâncias: breast, com uma diferença de 0,031679, e D31 alcançando 0,036178 a mais na

função. O valor silhueta mais próximo obtido foi para a instância a1, onde o algoritmo alcança uma diferença inferior a 0,01.

Apesar do bom desempenho em tempo de processamento do AGBL5 os valores silhueta por ele obtidos não tiveram grande destaque. Os melhores resultados conseguiram empatar com os de AGBL1 apenas em quatro casos, perdendo em todos os outros. Comparando com AGBL2 e AGBL3, a proposta alcançou valores próximos em 9 instâncias, empatando quatro e cinco vezes respectivamente. Para quatro instâncias o AGBL5 obteve melhores valores silhueta do que o AGBL2, em duas delas com uma diferença superior a 0,01. Quanto a AGBL3, os resultados de AGBL5 conseguiram ganhar em três situações, sendo que os valores obtidos por duas delas alcançaram uma diferença maior do que 0,01.

Ao tentar encontrar alternativas para a formação dos grupos iniciais tal processo pode ter se tornado demasiado restrito. Nas tentativas de obter agrupamentos de qualidade com um menor custo computacional os valores de aptidão alcançados tiveram seus valores reduzidos. Assim, observando os valores atingidos pela função objetivo nas propostas AGBL2, AGBL3, AGBL4 e AGBL5 pode ser notado que, de diferentes formas a qualidade do valor Índice Silhueta foi influenciada de forma negativa. Por possuir menos aspectos limitadores em sua metodologia de formação de grupos iniciais o AGBL1 conseguiu maiores valores de avaliação em suas resoluções. Porém ao analisar o número de grupos gerado pelos algoritmos pode ser observado que apesar das elevadas aptidões de AGBL1 muitas de suas soluções configuraram agrupamentos inadequados. Sabe-se que um agrupamento de qualidade busca o equilíbrio entre a relação de homogeneidade e separação de grupos. Nesse caso foram gerados resultados com altos números de grupos constituídos por poucos elementos. Considerando as resoluções para instância iris, por exemplo, enquanto as outras propostas obtiveram um número de grupos menor, mais próximo do conhecido na literatura, o algoritmo AGBL1 gerou 143 grupos com um maior valor Silhueta. Estes grupos podem ter uma boa coesão interna mas uma separação entre grupos mal definida, possivelmente o primeiro critério teve um maior impacto na avaliação de qualidade. Como esta técnica não obteve o processamento mais ágil nem o número esperado de grupos em diferentes casos decidiu-se verificar a sua execução com a aplicação de outra função de avaliação para melhorar as configurações de grupos geradas com bons valores de aptidão. Dando prosseguimento ao trabalho foi adotada a metodologia de AGBL1. O seu comportamento será analisado nas próximas seções comparando o uso da função já utilizada, a Índice Silhueta, e uma nova, a Índice CH.

5.2.2 Comparação silhueta x CH

Como descrito anteriormente, um dos pontos que este trabalho se propôs a analisar é referente à função adotada para a avaliação de qualidade das soluções. Uma comparação entre as funções Índice Silhueta e Índice CH empregadas é exibida no Quadro 5.3. Como também já

citado, os resultados finais apresentam valores Silhueta, porém um dos processos deste quadro aplica as avaliações com o uso da função CH. Logo, para este caso também serão apresentados os valores obtidos pela sua respectiva função de avaliação.

Na coluna "Conj. dados" são retratados os nomes dos conjuntos de dados. A coluna "AGBL1" apresenta os resultados com o uso do método de geração de grupos iniciais GSI1 e da função Índice Silhueta como função de avaliação. A coluna "AGBL6" inclui os resultados obtidos também adotando o procedimento GSI1 porém com a utilização da função Índice CH como forma de avaliação. A coluna denominada "Liter." corresponde às informações conhecidas na literatura. Cada conjunto de dados tem seus resultados apresentados em quatro linhas. A primeira, "Núm. grupos", contém o número de grupos obtido na solução final. A segunda, "Valor Silhueta", inclui os valores alcançados pela função Índice Silhueta para a configuração final gerada. A terceira, "Valor CH", apresenta os valores obtidos pela respectiva função quando adotada como forma de avaliação. Por fim, a quarta nomeada "Tempo" equivale ao tempo de computação para a instância. Os experimentos foram realizados para um total de 48 conjuntos de dados das seguintes coleções: Comportadas, Não comportadas e Outras instâncias.

Quadro 5.3: Resultados obtidos com o uso de diferentes funções de avaliação. *(continua)*

Conj. dados		**AGBL1**	**AGBL6**	**Liter.**
Maronna	Núm. grupos	169	2	4
	Valor Silhueta	**0,910095**	0,562172	
	Valor CH		274,996	
	Tempo	5	0	
200DATA	Núm. grupos	159	3	4
	Valor Silhueta	**0,84846**	0,823151	
	Valor CH		530,28	
	Tempo	5	0	
Broken ring	Núm. grupos	2	2	5
	Valor Silhueta	**0,980893**	0,687451	
	Valor CH		159,964	
	Tempo	1,4	1	
100p3c	Núm. grupos	85	3	3
	Valor Silhueta	**0,873481**	0,785802	
	Valor CH		198,116	
	Tempo	0,4	0	
100p7c	Núm. grupos	96	7	7
	Valor Silhueta	**0,981499**	0,833863	
	Valor CH		186,113	
	Tempo	0,8	0	

Quadro 5.3. *Continuação*

Conj. dados		**AGBL1**	**AGBL6**	**Liter.**
100p10c	Núm. grupos	98	10	10
	Valor Silhueta	**0,983481**	0,833613	
	Valor CH		140,566	
	Tempo	0,8	0	
100p2c1	Núm. grupos	92	2	2
	Valor Silhueta	**0,938813**	0,74274	
	Valor CH		257,271	
	Tempo	0,6	0	
100p3c1	Núm. grupos	96	3	3
	Valor Silhueta	**0,971191**	0,580263	
	Valor CH		108,262	
	Tempo	0,8	0,2	
100p5c1	Núm. grupos	83	5	5
	Valor Silhueta	**0,780534**	0,684301	
	Valor CH		116,551	
	Tempo	0,6	0,2	
100p7c1	Núm. grupos	107	2	7
	Valor Silhueta	**0,959163**	0,473892	
	Valor CH		135,312	
	Tempo	1	0	
200p4c	Núm. grupos	187	4	4
	Valor Silhueta	**0,956433**	0,772547	
	Valor CH		298,394	
	Tempo	5,2	0,4	
200p2c1	Núm. grupos	145	2	2
	Valor Silhueta	**0,802865**	0,76417	
	Valor CH		636,475	
	Tempo	4,6	2,4	
200p3c1	Núm. grupos	181	2	3
	Valor Silhueta	**0,930249**	0,648382	
	Valor CH		361,114	
	Tempo	5,2	0	
200p4c1	Núm. grupos	165	4	4
	Valor Silhueta	**0,854091**	0,744936	
	Valor CH		280,164	
	Tempo	4,8	1	
200p7c1	Núm. grupos	195	2	7
	Valor Silhueta	**0,948975**	0,53906	
	Valor CH		301,242	
	Tempo	6	0	

Quadro 5.3. *Continuação*

Conj. dados		**AGBL1**	**AGBL6**	**Liter.**
200p12c1	Núm. grupos	201	2	12
	Valor Silhueta	**0,925712**	0,524953	
	Valor CH		298,195	
	Tempo	7,2	0	
300p3c	Núm. grupos	252	3	3
	Valor Silhueta	**0,893164**	0,766375	
	Valor CH		634,296	
	Tempo	16,8	0,2	
300p2c1	Núm. grupos	244	2	2
	Valor Silhueta	**0,860352**	0,77669	
	Valor CH		805,498	
	Tempo	16,6	0,8	
300p3c1	Núm. grupos	270	2	3
	Valor Silhueta	**0,926111**	0,63254	
	Valor CH		489,531	
	Tempo	17,4	0,2	
300p4c1	Núm. grupos	283	2	4
	Valor Silhueta	**0,958464**	0,503781	
	Valor CH		350,574	
	Tempo	18,8	0	
300p6c1	Núm, grupos	263	2	6
	Valor Silhueta	**0,904417**	0,548274	
	Valor CH		386,583	
	Tempo	16,4	0,8	
300p13c1	Núm. grupos	213	2	13
	Valor Silhueta	**0,924156**	0,545101	
	Valor CH		342,347	
	Tempo	8	0,2	
400p3c	Núm. grupos	341	3	3
	Valor Silhueta	**0,891211**	0,798579	
	Valor CH		919,626	
	Tempo	46	0,4	
400p4c1	Núm. grupos	322	2	4
	Valor Silhueta	**0,839479**	0,541096	
	Valor CH		513,395	
	Tempo	36,4	15,6	
400p17c1	Núm. grupos	368	2	17
	Valor Silhueta	**0,928069**	0,513234	
	Valor CH		573,552	
	Tempo	44,2	1	

Quadro 5.3. *Continuação*

Conj. dados		**AGBL1**	**AGBL6**	**Liter.**
500p3c	Núm. grupos	465	3	3
	Valor Silhueta	**0,943565**	0,824936	
	Valor CH		1454,37	
	Tempo	89,2	19,8	
500p4c1	Núm. grupos	471	2	4
	Valor Silhueta	**0,950054**	0,63033	
	Valor CH		896,475	
	Tempo	91,4	16,4	
500p6c1	Núm. grupos	402	2	6
	Valor Silhueta	**0,837209**	0,429675	
	Valor CH		505,718	
	Tempo	74,6	1	
600p15c	Núm. grupos	547	16	15
	Valor Silhueta	**0,932803**	0,747162	
	Valor CH		437,05	
	Tempo	193	155,8	
600p3c1	Núm. grupos	488	3	3
	Valor Silhueta	**0,827741**	0,721168	
	Valor CH		1146,35	
	Tempo	163,2	9	
700p4c	Núm. grupos	621	4	4
	Valor Silhueta	**0,921318**	0,796956	
	Valor CH		1181,22	
	Tempo	335,4	5,4	
700p15c1	Núm. grupos	550	2	15
	Valor Silhueta	**0,820419**	0,387273	
	Valor CH		652,843	
	Tempo	338,4	5,4	
800p23c	Núm. grupos	728	24	23
	Valor Silhueta	**0,930737**	0,763279	
	Valor CH		492,824	
	Tempo	553,4	299,6	
800p4c1	Núm. grupos	630	4	4
	Valor Silhueta	**0,83645**	0,70434	
	Valor CH		1003,7	
	Tempo	546,2	161,8	
800p10c1	Núm. grupos	688	2	10
	Valor Silhueta	**0,894152**	0,467051	
	Valor CH		878,184	
	Tempo	501	155,4	

Quadro 5.3. *Continuação*

Conj. dados		**AGBL1**	**AGBL6**	**Liter.**
800p18c1	Núm. grupos	629	2	18
	Valor Silhueta	**0,825029**	0,417692	
	Valor CH		811,169	
	Tempo	477,6	6	
900p5c	Núm. grupos	824	5	5
	Valor Silhueta	**0,94248**	0,716048	
	Valor CH		928,315	
	Tempo	781,6	4,6	
900p12c	Núm. grupos	825	12	12
	Valor Silhueta	**0,94567**	0,840808	
	Valor CH		1016,1	
	Tempo	846,8	76,8	
1000p6c	Núm. grupos	906	6	6
	Valor Silhueta	**0,936172**	0,73567	
	Valor CH		1172,81	
	Tempo	1078,8	582,6	
1000p14c	Núm. grupos	908	14	14
	Valor Silhueta	**0,936088**	0,830566	
	Valor CH		1015,62	
	Tempo	1187,6	206	
1000p5c1	Núm. grupos	859	4	5
	Valor Silhueta	**0,892347**	0,614515	
	Valor CH		853,24	
	Tempo	1092,4	17,8	
1000p27c1	Núm. grupos	852	2	27
	Valor Silhueta	**0,884936**	0,477777	
	Valor CH		1211,12	
	Tempo	1152,2	99,4	
1100p6c1	Núm. grupos	944	6	6
	Valor Silhueta	**0,891761**	0,673319	
	Valor CH		1063,59	
	Tempo	1328,2	441	
1300p17c	Núm. grupos	17	2	17
	Valor Silhueta	**0,822918**	0,442087	
	Valor CH		1299,85	
	Tempo	23,8	14,6	
1500p6c1	Núm. grupos	6	6	6
	Valor Silhueta	**0,645995**	**0,645448**	
	Valor CH		1343,79	
	Tempo	807,4	1266,6	

Quadro 5.3. *Continuação*

Conj. dados		AGBL1	AGBL6	Liter.
1800p22c	Núm. grupos	22	23	22
	Valor Silhueta	**0,803624**	0,772767	
	Valor CH		1329,69	
	Tempo	234,2	391	
2000p11c	Núm. grupos	11	2	11
	Valor Silhueta	**0,712972**	0,516363	
	Valor CH		2603,8	
	Tempo	79,8	46,2	
2000p9c1	Núm. grupos	9	2	9
	Valor Silhueta	**0,626429**	0,501528	
	Valor CH		2532,81	
	Tempo	634,2	730,6	

Observando o número de grupos encontrado em cada estratégia pode ser observado que, enquanto o AGBL1 identificou o número correto de grupos para cinco instâncias, o AGBL6 conseguiu obter tal valor para 22 destas. Com relação a Outras instâncias, nenhum dos dois algoritmos conseguiu alcançar a quantidade exata de grupos. Sobre os conjuntos de Comportadas, apenas para três casos o AGBL1 obteve o valor correto. Já o AGBL6 atingiu este número para 12 instâncias, ou seja, aproximadamente 85% de acertos alcançados neste cenário. Em Não comportadas, o AGBL1 atingiu o número de grupos certo somente em dois casos. Em contrapartida, no AGBL6 foi obtido dez vezes o número correto. Mais uma vez, considerando conjuntos de dados diferentes dos adotados nos experimentos da etapa anterior, o AGBL1 gera soluções com uma configuração insatisfatória de grupos.

Também foram analisados os resultados considerando resoluções que se aproximam do número correto de grupos em até duas unidades. Na atuação de AGBL1, não foi identificada nenhuma solução que conseguisse tal proximidade de valores. Com o AGBL6 foram constatados 11 resultados que não obtiveram esta quantidade exata mas alcançaram resoluções com um certo grau de semelhança para o número de grupos. Destes, dois deles correspondem a conjuntos de dados de Outras instâncias, outros três fazem parte de Comportadas e, por fim, seis integram a coleção Não comportadas.

Ao analisar os valores Índice Silhueta observa-se claramente o destaque das soluções derivadas do AGBL1. Para todos os conjuntos de dados tal algoritmo possui os maiores resultados comparado ao AGBL6. A diferença média entre as silhuetas de AGBL1 e AGBL6 foi de, aproximadamente, 0,230894. O maior afastamento entre os valores da função chegou a 0,485271.

A maior proximidade entre os resultados destes procedimentos atingiu um valor de 0,000547, na instância 1500p6c1, onde ambos algoritmos conseguiram atingir o número certo de grupos e sendo a única vez em que a diferença foi inferior a 0,02. A média de diferença entre os resultados das instâncias de Comportadas ficou em torno de 0,146122. Para os conjuntos de dados de Não comportadas a diferença média foi verificada por volta do valor 0,274194.

Sobre o tempo de processamento pode ser verificado que, o uso da função Índice CH originou uma significativa melhora em desempenho. Ocorreu, em média, uma redução de tempo de 63% ao empregar o AGBL6 ao invés do AGBL1. Somente para três instâncias o AGBL6 apresentou um pior desempenho, sendo estas: 1500p6c1, 1800p22c e 2000p9c1.

Apesar da redução nos valores silhueta as soluções do AGBL6 obtiveram bons resultados de agrupamento, encontrando muitas vezes o número correto de grupos ou valores próximos a ele. Além disto, um melhor desempenho foi alcançado em termos de tempo de processamento. A função Índice CH originou uma expressiva otimização no custo computacional acertando em mais de 45% no número de grupos das soluções.

5.2.3 Comparação com outros trabalhos da literatura

Os próximos quadros apresentam uma análise dos resultados obtidos neste trabalho com algumas propostas da literatura, sendo estas: GADE, AECBL1, MRDBSCAN, AK-means e ACO [11, 29, 31, 55, 46]. Para a comparação serão considerados os algoritmos AGBL1 e AGBL6, que utilizam a mesma função geradora de grupos iniciais e adotam como forma de avaliação o Índice Silhueta e Índice CH, respectivamente. Valores Índice Silhueta simbolizam a qualidade da solução final nos quadros que serão apresentados. Serão considerados como resultados equivalentes os valores silhueta com até 0,02 de diferença, a fim de desconsiderar possíveis arredondamentos das estratégias dos outros trabalhos.

5.2.3.1 Comparação com AECBL1, MRDBSCAN e ACO

O Quadro 5.4 apresenta uma comparação entre os resultados obtidos com o uso de cada uma desses procedimentos e o de outros trabalhos da literatura. A coluna “Conj. dados” indica o nome de cada conjunto de dados. A coluna “AGBL1” apresenta os resultados com o uso da função Índice Silhueta como função de avaliação. A coluna “AGBL6” inclui os resultados obtidos utilizando a função Índice CH como forma de avaliação. As colunas “MRDBSCAN”, “AECBL1” e “ACO” contêm os resultados dos respectivos algoritmos da literatura: MRDBSCAN, AECBL1 e ACO. A coluna denominada “Liter.” corresponde às informações conhecidas na literatura. Cada conjunto de dados tem seus resultados apresentados em duas linhas. A primeira, “Núm. grupos”, contém o número de grupos obtido na solução final. A segunda, “Valor Silhueta”, inclui os valores

alcançados pela função Índice Silhueta para a solução final gerada. Os resultados apresentados neste quadro fazem uso dos mesmos conjuntos de dados do Quadro 5.3.

Quadro 5.4: Comparação dos resultados obtidos nas propostas deste trabalho, AGBL1 e AGBL6, com os algoritmos AECBL1, MRDBSCAN e ACO. *(continua)*

Conj. dados		**AGBL1**	**AGBL6**	**MRDBSCAN**	**AECBL1**	**ACO**	**Liter.**
Maronna	Núm. grupos	169	2	2	4	2	4
	Valor Silhueta	**0,910095**	0,562172	0,562	0,5745	0,562	
200DATA	Núm. grupos	159	3	3	3	3	4
	Valor Silhueta	**0,84846**	0,823151	0,823	0,8231	0,823	
Broken ring	Núm. grupos	2	2		5		5
	Valor Silhueta	**0,980893**	0,687451		0,4995		
100p3c	Núm. grupos	85	3	3	3	3	3
	Valor Silhueta	**0,873481**	0,785802	0,786	0,7858	0,786	
100p7c	Núm. grupos	96	7	7	7	7	7
	Valor Silhueta	**0,981499**	0,833863	0,834	0,8338	0,834	
100p10c	Núm. grupos	98	10	8	10	10	10
	Valor Silhueta	**0,983481**	0,833613	0,692	0,8336	0,834	
100p2c1	Núm. grupos	92	2	2	2		2
	Valor Silhueta	**0,938813**	0,74274	0,743	0,7427		
100p3c1	Núm. grupos	96	3	5	3	4	3
	Valor Silhueta	**0,971191**	0,580263	0,104	0,5802	0,133	
100p5c1	Núm. grupos	83	5	2	7	17	5
	Valor Silhueta	**0,780534**	0,684301	0,423	0,6958	0,729	
100p7c1	Núm. grupos	107	2	2	7	23	7
	Valor Silhueta	**0,959163**	0,473892	-0,013	0,4911	0,326	
200p4c	Núm. grupos	187	4	4	4	4	4
	Valor Silhueta	**0,956433**	0,772547	0,773	0,7725	0,773	
200p2c1	Núm. grupos	145	2	6	2	2	2
	Valor Silhueta	**0,802865**	0,76417	0,625	0,7642	0,749	
200p3c1	Núm. grupos	181	2	2	3	2	3
	Valor Silhueta	**0,930249**	0,648382	0,648	0,6797	0,648	
200p4c1	Núm. grupos	165	4	3	4		4
	Valor Silhueta	**0,854091**	0,744936	0,623	0,7449		
200p7c1	Núm. grupos	195	2	3	13	8	7
	Valor Silhueta	**0,948975**	0,53906	0,392	0,5759	0,310	
200p12c1	Núm. grupos	201	2	3	13	12	12
	Valor Silhueta	**0,925712**	0,524953	0,403	0,5753	0,321	
300p3c	Núm. grupos	252	3	3	3	3	3
	Valor Silhueta	**0,893164**	0,766375	0,766	0,7663	0,766	

Quadro 5.4. *Continuação*

Conj. dados		AGBL1	AGBL6	MRDBSCAN	AECBL1	ACO	Liter.
300p2c1	Núm. grupos	244	2	4	2	2	2
	Valor Silhueta	**0,860352**	0,77669	0,621	0,7764	0,758	
300p3c1	Núm. grupos	270	2	2	3	2	3
	Valor Silhueta	**0,926111**	0,63254	0,640	0,6768	0,690	
300p4c1	Núm. grupos	283	2	3	2		4
	Valor Silhueta	**0,958464**	0,503781	0,269	0,5910		
300p6c1	Núm. grupos	263	2	2	8	11	6
	Valor Silhueta	**0,904417**	0,548274	0,549	0,6636	0,577	
300p13c1	Núm. grupos	213	2	3	13	5	13
	Valor Silhueta	**0,924156**	0,545101	0,404	0,5644	0,449	
400p3c	Núm. grupos	341	3	3	3	3	3
	Valor Silhueta	**0,891211**	0,798579	0,799	0,7985	0,799	
400p4c1	Núm. grupos	322	2	2	4	2	4
	Valor Silhueta	**0,839479**	0,541096	0,379	0,5989	0,382	
400p17c1	Núm. grupos	368	2	14	2	24	17
	Valor Silhueta	**0,928069**	0,513234	0,183	0,5138	0,193	
500p3c	Núm. grupos	465	3	3	3	3	3
	Valor Silhueta	**0,943565**	0,824936	0,825	0,8249	0,825	
500p4c1	Núm. grupos	471	2	2	5		4
	Valor Silhueta	**0,950054**	0,63033	0,305	0,6595		
500p6c1	Núm. grupos	402	2	12	6	20	6
	Valor Silhueta	**0,837209**	0,429675	0,495	0,6287	0,557	
600p15c	Núm. grupos	547	16	15	15	15	15
	Valor Silhueta	**0,932803**	0,747162	0,781	0,7812	0,781	
600p3c1	Núm. grupos	488	3	2	3	3	3
	Valor Silhueta	**0,827741**	0,721168	0,687	0,7209	0,661	
700p4c	Núm. grupos	621	4	4	4	4	4
	Valor Silhueta	**0,921318**	0,796956	0,797	0,7969	0,797	
700p15c1	Núm. grupos	550	2	2	15		15
	Valor Silhueta	**0,820419**	0,387273	0,123	0,6804		
800p23c	Núm. grupos	728	24	23	23	20	23
	Valor Silhueta	**0,930737**	0,763279	0,787	0,7873	0,724	
800p4c1	Núm. grupos	630	4	2	4		4
	Valor Silhueta	**0,83645**	0,70434	0,509	0,7021		
800p10c1	Núm. grupos	688	2	2	2	30	10
	Valor Silhueta	**0,894152**	0,467051	0,079	0,4681	0,092	
800p18c1	Núm. grupos	629	2	24	19	16	18
	Valor Silhueta	**0,825029**	0,417692	0,266	0,6914	0,628	
900p5c	Núm. grupos	824	5	5	5	5	5
	Valor Silhueta	**0,94248**	0,716048	0,716	0,7160	0,716	

Quadro 5.4. *Continuação*

Conj. dados		AGBL1	AGBL6	MRDBSCAN	AECBL1	ACO	Liter.
900p12c	Núm. grupos	825	12	12	12	11	12
	Valor Silhueta	**0,94567**	0,840808	0,841	0,8408	0,818	
1000p6c	Núm. grupos	906	6	6	6	5	6
	Valor Silhueta	**0,936172**	0,73567	0,736	0,7356	0,709	
1000p14c	Núm. grupos	908	14	15	14	15	14
	Valor Silhueta	**0,936088**	0,830566	0,808	0,8306	0,808	
1000p5c1	Núm. grupos	859	4	2	5	11	5
	Valor Silhueta	**0,892347**	0,614515	0,164	0,6391	0,586	
1000p27c1	Núm. grupos	852	2	3	25	35	27
	Valor Silhueta	**0,884936**	0,477777	-0,293	0,5186	0,313	
1100p6c1	Núm. grupos	944	6	5	6	12	6
	Valor Silhueta	**0,891761**	0,673319	0,369	0,6717	0,618	
1300p17c	Núm. grupos	17	2	18	17		17
	Valor Silhueta	**0,822918**	0,442087	0,806	**0,8229**		
1500p6c1	Núm. grupos	6	6	18	6	10	6
	Valor Silhueta	**0,645995**	**0,645448**	0,123	**0,6436**	**0,630**	
1800p22c	Núm. grupos	22	23	23	22		22
	Valor Silhueta	**0,803624**	0,772767	0,791	**0,8036**		
2000p11c	Núm. grupos	11	2	11	11	11	11
	Valor Silhueta	**0,712972**	0,516363	**0,713**	**0,7129**	**0,713**	
2000p9c1	Núm. grupos	9	2	2	9	15	9
	Valor Silhueta	**0,626429**	0,501528	0,164	**0,6230**	0,572	

Os melhores resultados silhueta foram obtidos pelo AGBL1, que empatou/ganhou em todas as instâncias comparado aos trabalhos da literatura analisados. A metodologia AECBL1 empata cinco vezes com os valores encontrados por AGBL1. Já os valores de MRDBSCAN empatam em três conjuntos de dados. Em duas situações houveram empates com as silhuetas do ACO. Tais resultados iguais entre os algoritmos foram para as seguintes instâncias: 1300p17c, 1500p6c1, 1800p22c, 2000p11c e 2000p9c1. Especialmente, estes cinco casos foram os únicos onde o AGBL1 conseguiu obter a quantidade correta de grupos.

Considerando o AGBL6, a técnica alcançou bons resultados, muitas vezes empatando seus valores silhueta com esses trabalhos da literatura. Comparado ao AECBL1, foram alcançados resultados análogos em 28 instâncias. Foram obtidos valores inferiores em apenas 19 casos, e ganhou-se uma vez no conjunto de dados broken ring. Apesar do AGBL6 perder em aproximadamente 39% dos casos, foram atingidos resultados equivalentes aos do AECBL1 em cerca de 58% dos conjuntos de dados. Observando o algoritmo MRDBSCAN, conseguiu-se empatar os valores

silhueta em 17 instâncias, ganhou-se em outras 25 e foram obtidos resultados inferiores apenas cinco vezes. Assim, o algoritmo AGBL6 obteve valores silhueta equivalentes ou maiores do que os do MRDBSCAN em pelo menos 89% dos casos. Com relação ao ACO, foi alcançada equivalência nos valores em 15 conjuntos de dados. Valores superiores foram atingidos em 16 casos, e oito vezes foram obtidos resultados inferiores. Deste modo, os resultados do AGBL6 empataram ou ganharam dos gerados pelo ACO no mínimo em 79% das instâncias.

No que se refere à quantidade de grupos, o AGBL6 acerta este valor para 22 instâncias. Dos trabalhos confrontados, o AECBL1 consegue atingir o valor correto mais vezes, para um total de 37 conjuntos de dados. Os outros dois, MRDBSCAN e ACO, encontram ao todo em 14 e 15 situações, respectivamente. Considerando, também, a descoberta de valores próximos, o AGBL6 alcançou um certo grau de semelhança para 11 instâncias, assim como o ACO. Já o MRDBSCAN obteve valores próximos 17 vezes, e o AECBL1 em oito situações.

5.2.3.2 Comparação com GADE e ACO

Os métodos GADE e ACO serão comparados aos resultados deste trabalho no Quadro 5.5. A coluna "Conj. dados" apresenta o nome de cada conjunto de dados. A coluna "AGBL1" contém os resultados obtidos por meio do algoritmo AGBL1. Na coluna "AGBL6" estão presentes os resultados alcançados com o algoritmo AGBL6. A coluna "GADE" apresenta os resultados obtidos pelos autores do algoritmo GADE. A coluna "ACO" corresponde aos resultados gerados pelo respectivo algoritmo ACO. Cada conjunto de dados tem seus resultados apresentados em duas linhas. A primeira, "Núm. grupos", contém o número de grupos gerado pela solução. A segunda, "Valor Silhueta", inclui os valores alcançados pela função Índice Silhueta para a solução resultante. Foram realizadas comparações para os resultados referentes aos seguintes conjuntos de dados: ruspini, iris, wine e breast.

Quadro 5.5: Comparação dos resultados obtidos nas propostas deste trabalho, AGBL1 e AGBL6, com os métodos GADE e ACO.

Conj. dados		**AGBL1**	**AGBL6**	**GADE**	**ACO**
ruspini	Núm. grupos	69	4		4
	Valor Silhueta	**0,926141**	0,737657		0,738
iris	Núm. grupos	143	2	2.31	2
	Valor Silhueta	**0,971475**	0,686393	0,429655	0,686
wine	Núm. grupos	124	2	3.16	
	Valor Silhueta	**0,818327**	0,61704	0,582197	
breast	Núm. grupos	243	2	2.08	
	Valor Silhueta	**0,678474**	0,593488	0,648297	

Os valores silhueta alcançados pelo AGBL1 ultrapassam tanto o AGBL6 quanto as outras metodologias da literatura, porém para nenhum dos conjuntos de dados é encontrado o número correto de grupos nem valores próximos. O AGBL6 e ACO empataram nos valores silhueta para ruspini e iris. Já o GADE, apesar de superar a silhueta de AGBL6 uma vez em breast, obteve valores inferiores para as outras duas instâncias.

Para o conjunto de dados iris, embora o AGBL6 não tenha encontrado seu número certo de grupos um valor próximo a este foi obtido, assim como nos algoritmos da literatura. Do mesmo modo que o ACO, o AGBL6 conseguiu obter a quantidade correta de grupos para ruspini. O mesmo aconteceu com a instância breast, foi encontrado o valor certo tal qual o GADE. Sobre o conjunto de dados wine, diferentemente de GADE, o número correto não foi encontrado, porém uma quantidade próxima foi alcançada por AGBL6.

5.2.3.3 Comparação com AK-means

A metodologia AK-means será comparada no Quadro 5.6. O nome de cada conjunto de dados apresentado na coluna "Conj. dados". Os resultados alcançados com a função de avaliação Índice Silhueta estão presentes na coluna "AGBL1". A coluna "AGBL6" apresenta os resultados gerados a partir da avaliação da função Índice CH. Os resultados obtidos pelos autores do respectivo algoritmo em comparação são apresentados na coluna "AK-means". Cada conjunto de dados tem seus resultados apresentados em três linhas. A primeira, "Núm. grupos", contém a quantidade de grupos gerada pela solução. A segunda, "Valor Silhueta", exibe os valores alcançados pela função Índice Silhueta para a solução resultante. A última, "Valor CH", indicas os valores obtidos pela respectiva função quando adotada como forma de avaliação. Neste quadro são exibidas as comparações para os resultados referentes a 16 conjuntos de dados das seguintes coleções: UCI *datasets*, DIM-*sets* (*high*), A-*sets*, *Shape sets* e Outras instâncias.

Quadro 5.6: Comparação dos resultados obtidos nas propostas deste trabalho, AGBL1 e AGBL6, com o algoritmo AK-means. *(continua)*

Conj. dados		**AGBL1**	**AGBL6**	**AK-means**
ruspini	Núm. grupos	69	4	4
	Valor Silhueta	**0,926141**	0,737657	0,9086
	Valor CH		112,015	
iris	Núm. grupos	143	2	3
	Valor Silhueta	**0,971475**	0,686393	0,7786
	Valor CH		305,805	

Quadro 5.6. *Continuação*

Conj. dados		**AGBL1**	**AGBL6**	**AK-means**
wine	Núm. grupos	124	2	3
	Valor Silhueta	**0,818327**	0,61704	0,5043
	Valor CH		282,443	
glass	Núm. grupos	154	2	15
	Valor Silhueta	**0,709138**	0,634285	0,6514
	Valor CH		14,1576	
thyroid	Núm. grupos	187	2	3
	Valor Silhueta	**0,848463**	0,602425	0,7773
	Valor CH		66,4384	
breast	Núm. grupos	243	2	2
	Valor Silhueta	0,678474	0,593488	**0,7542**
	Valor CH		984,485	
yeast	Núm. grupos	2	2	2
	Valor Silhueta	**0,611731**	0,561033	0,4102
	Valor CH		87,0216	
dim32	Núm. grupos	16	16	16
	Valor Silhueta	0,945562	0,945562	**0,9962**
	Valor CH		1661,72	
dim64	Núm. grupos	16	16	16
	Valor Silhueta	0,966338	0,966338	**0,9985**
	Valor CH		2379,18	
dim128	Núm. grupos	16	16	16
	Valor Silhueta	0,974642	0,974642	**0,9991**
	Valor CH		3046,76	
dim256	Núm. grupos	16	16	16
	Valor Silhueta	0,982946	0,982946	**0,9996**
	Valor CH		4356,83	
R15	Núm. grupos	511	15	15
	Valor Silhueta	0,900603	0,752739	**0,9361**
	Valor CH		439,674	
D31	Núm. grupos	31	2	31
	Valor Silhueta	0,572864	0,393086	**0,9222**
	Valor CH		2901,52	
a1	Núm. grupos	20	2	20
	Valor Silhueta	0,593601	0,502558	**0,7892**
	Valor CH		4131,56	
a2	Núm. grupos	35	2	35
	Valor Silhueta	0,595603	0,460388	**0,7911**
	Valor CH		6061,87	

Quadro 5.6. *Continuação*

Conj. dados		AGBL1	AGBL6	AK-means
a3	Núm. grupos	50	2	50
	Valor Silhueta	0,593568	0,377202	**0,7949**
	Valor CH		6526,36	

Analisando os valores silhueta de AGBL1, são alcançadas aptidões equivalentes ou superiores às de AGBL6 para todos os conjuntos de dados. Porém, em geral, as soluções deste último configuram números de grupos mais coerentes do que os de AGBL1. Ao confrontar os resultados silhueta obtidos por AGBL1 e AK-means observa-se que, o primeiro algoritmo conseguiu empatar ou ganhar para sete instâncias, perdendo do AK-means em outras nove. Já o AGBL6, obteve valores inferiores em 12 casos, e empatou ou ganhou em quatro.

No que diz respeito à quantidade de grupos obtida nas soluções, ambos AGBL1 e AGBL6 geraram o número correto para metade das instâncias, sendo estas diferentes para cada um destes. O AK-means atingiu o valor certo em 13 casos. Considerando o alcance de valores próximos ao correto, com uma diferença de até duas unidades, o AGBL6 gerou tal proximidade para dois conjuntos de dados, o AK-means em um caso e o AGBL1 não conseguiu em nenhuma situação.

5.2.4 Análise gráfica de soluções

Certos resultados obtidos pelo AGBL6 alcançaram valores silhueta pouco satisfatórios, sendo inferiores ao de outros algoritmos durante as comparações. Considerando AGBL1 e AK-means foi observado que, algumas de suas soluções atingem valores silhueta muito maiores do que os obtidos por AGBL6. Porém o primeiro algoritmo, além de poucas vezes acertar o número de grupos tem um custo computacional consideravelmente maior como visto no Quadro 5.3. Sobre o AK-means, para alguns conjuntos de dados, apesar do resultado silhueta do AGBL6 ser inferior o número correto de grupos é encontrado. Considerando tais fatos, decidiu-se efetuar uma análise mais criteriosa. Algumas das resoluções onde a quantidade correta de grupos é encontrada pelo AGBL6, e as dimensões da instância permitem ilustração, serão examinadas graficamente. As quatro figuras a seguir exibirão as soluções de conjuntos de dados do $\mathbb{R}^2$, de diferentes tipos e tamanhos, sendo estes: ruspini, R15, 300p2c1 e 1000p6c. Cada figura foi esboçada através do programa gnuplot, via linha de comando, onde a ilustração representa a organização dos grupos gerados para cada instância.

Observando graficamente, percebe-se que a configuração de grupos dos quatro conjuntos é adequada, apresentando certa homogeneidade em cada grupo e boas demarcações entre os grupos.

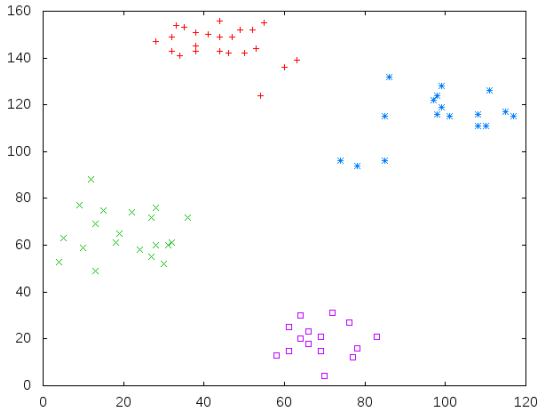


Figura 5.3: Solução gerada para ruspini com o algoritmo AGBL6.

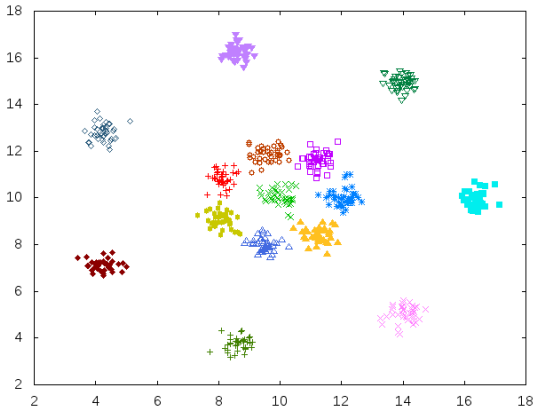


Figura 5.4: Solução gerada para R15 com o algoritmo AGBL6.

Apesar da função Silhueta não atingir valores maiores de modo visual é constatado que, com a quantidade correta de grupos encontrada as soluções geradas podem ser consideradas ótimas, suas partições estão corretas. O algoritmo AGBL6 é sim uma estratégia válida porém não 100% precisa. Logo, esta pode ser aprimorada para ajustar os casos onde não é encontrado o número correto de grupos.

Em uma comparação com soluções obtidas em outros trabalhos, ilustradas anteriormente na Figura 2.1, é visto que o agrupamento obtido por AGBL6 para a instância 300p2c1 possui uma considerável semelhança à resolução de AECBL1. Na Figura 5.7 é visto que, diferente da técnica CLUES, ambos geram o mesmo número de grupos sendo este o mesmo valor estimado na literatura, como pode ser verificado no Quadro 5.4.

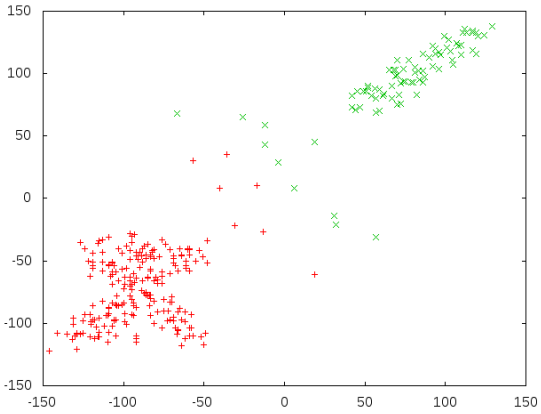


Figura 5.5: Solução gerada para 300p2c1 com o algoritmo AGBL6.

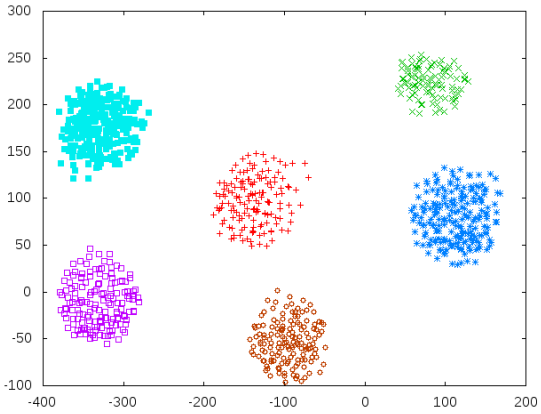


Figura 5.6: Solução gerada para 1000p6c com o algoritmo AGBL6.

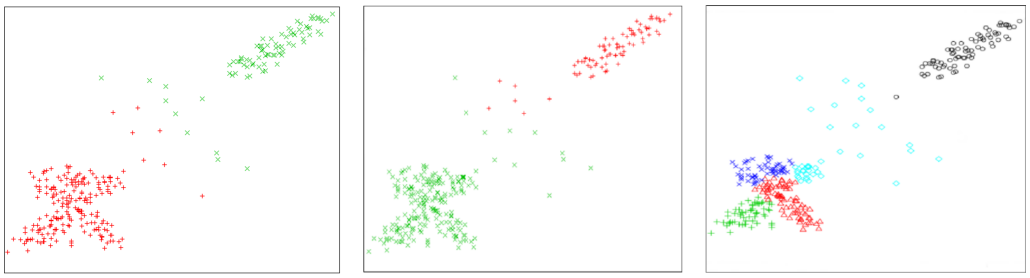


Figura 5.7: Comparação entre os agrupamentos obtidos por AGBL6, AECBL1 e CLUES para a instância 300p2c1 [11].

6 CONSIDERAÇÕES FINAIS

Este trabalho apresentou as principais características da análise de grupos, tendo ênfase no Problema de Agrupamento Automático na forma particional. Experimentos foram realizados para a resolução do PCA tendo como base a meta-heurística Algoritmo Genético. Foi observado que a qualidade de um agrupamento é fortemente influenciada pela estratégia de geração de grupos iniciais e pela função de avaliação empregada. Considerando tais pontos, procedimentos foram desenvolvidos a partir de diferentes técnicas para produzir a solução inicial, além de dois métodos distintos de avaliação. Os resultados obtidos foram comparados com outros trabalhos da literatura.

6.1 Conclusões

Os experimentos efetuados no presente trabalho tiveram como base dois pontos que são capazes de impactar a qualidade da solução de um agrupamento, sendo estes o modo como são formados os grupos iniciais e a função objetivo responsável pela avaliação de qualidade. Com fundamento nisto, a partir de um Algoritmo Genético já existente, a primeira etapa produziu cinco propostas de formação inicial de grupos. A partir da proposta com maiores valores de aptidão foi concebida uma nova estratégia adotando uma função de avaliação diferente da utilizada na etapa de testes anterior. Após isto foram realizadas comparações com as soluções geradas por outros trabalhos da literatura.

Dentre as técnicas iniciais a que obteve maior destaque foi a de AGBL1. Este algoritmo registrou, num maior número de casos, os melhores valores de aptidão com relação aos outros quatro. Comparada a outras estratégias da literatura que também utilizam a função Silhueta como forma de avaliação mostrou-se que AGBL1 também conseguiu, muitas vezes, alcançar valores mais altos. Entretanto, para algumas instâncias o número correto de grupos não foi encontrado, frequentemente configurando uma numerosa quantidade de grupos com poucos elementos, além de seu custo de processamento ter sido pouco satisfatório. Com esses resultados fica claro o mau comportamento desta função, que consegue atingir valores aptidão de boa qualidade em meio a configurações de agrupamento insatisfatórias. Com o objetivo de manter o alto grau de aptidão das soluções o Algoritmo Genético adotou este procedimento inicial GSI1 na etapa seguinte do trabalho, utilizando outra forma de avaliação, tendo em vista a identificação da quantidade esperada de grupos com um menor custo computacional.

O algoritmo AGBL6 utilizou a melhor estratégia inicial citada acima com a função Índice CH como forma de avaliação. Os resultados deste procedimento atingiram menores valores de aptidão, porém ele conseguiu alcançar mais vezes o número correto de grupos ou valores próximos. De fato, com uma função de menor complexidade houve uma considerável melhora no tempo de computação, com uma média de 71% de redução em relação ao AGBL1.

Na comparação com outros trabalhos da literatura foram considerados os resultados das propostas AGBL1 e AGBL6. Apesar das soluções obtidas por AGBL6 serem melhores, AGBL1 também foi observado para indicar que neste trabalho conseguiu-se alcançar resultados com altos valores Silhueta entretanto com configurações inadequadas de agrupamento. Em geral, o AGBL1 atingiu melhores aptidões do que os outros algoritmos, porém poucas vezes encontrando a quantidade correta de grupos. Já o AGBL6 conseguiu alcançar valores silhueta equivalentes ou maiores aos de ACO, no primeiro cenário analisado, e MRDBSCAN em 79% e 89% dos casos, respectivamente. Também houveram empates com o AECBL1 para 58% dos conjuntos de dados. No que se refere à quantidade de grupos, na primeira comparação com trabalhos da literatura, o AGBL6 acerta este valor para 22 instâncias de um total de 48, perdendo apenas para o AECBL1. Foram semelhantes os resultados obtidos pelos algoritmos na análise que considera AGBL1, AGBL6, GADE e o ACO, com exceção pelo AGBL1.

A partir da comparação com o AK-means foi visto que, muitas vezes seus valores silhueta são superiores aos resultados obtidos pelo AGBL6, mesmo quando o número correto de grupos foi encontrado por esta proposta. De um total de 16 conjuntos de dados em dez vezes o AGBL6 conseguiu obter o número certo de grupos ou valores próximos a ele. Diante disto, foi realizada uma análise mais minuciosa observando os resultados das soluções graficamente. Constatou-se que os particionamentos são adequados, sendo constituídos por grupos de formação apropriada. Assim, pode-se concluir que a proposta AGBL6 é justificável, no entanto aperfeiçoamentos devem ser estudados para corrigir os casos onde não é alcançada a quantidade correta de grupos.

6.2 Trabalhos Futuros

Uma análise mais profunda pode ser realizada nas propostas para a geração de grupos iniciais. Como observado no Capítulo 5, alguns destes geradores estimularam a construção de certas resoluções de qualidade insatisfatória. Assim, pode-se tentar aprimorar tais estratégias para que se possa trabalhar com menos grupos inicialmente, a fim de otimizar o tempo de processamento, porém com configurações melhores.

Sobre a avaliação dos resultados de agrupamento também pode ser definido em atividades futuras um método de otimização multiobjetivo. Como nem sempre o valor de ambas funções, índice Silhueta e índice CH, melhora em conjunto pensar numa forma de utilizar as duas ao mesmo tempo como critério de avaliação seria uma estratégia promissora na busca de melhores soluções.

Outra possibilidade que pode ser explorada seria a de adotar distintas funções de avaliação em diferentes etapas do algoritmo. Uma função seria empregada apenas para a fase inicial de formação de grupos e outra para a continuidade do processamento. Além de experimentar o uso das funções já adotadas neste trabalho o uso de outras também pode ser estudado.

REFERÊNCIAS

[1] Clustering basic benchmark. Disponível em: `http://cs.joensuu.fi/sipu/datasets/`; Acessado em 04 de Abril de 2018.

[2] Machine learning repository. Disponível em: `http://archive.ics.uci.edu/ml/index.php`; Acessado em 13 de Julho de 2017.

[3] ANARI, B., TORKESTANI, J. A., E RAHMANI, A. Automatic data clustering using continuous action-set learning automata and its application in segmentation of images. *Applied Soft Computing 51* (2017), 253–265.

[4] ARENALES, M., MORABITO, R., ARMENTANO, V., E YANASSE, H. *Pesquisa operacional: Para cursos de engenharia*. Elsevier Brasil, 2015.

[5] BECKER, J. L. *Estatística básica: transformando dados em informação*. Bookman Editora, 2015.

[6] BERKHIN, P. A survey of clustering data mining techniques. In *Grouping multidimensional data*. Springer, 2006, pp. 25–71.

[7] BRITO, J. D. M., E MONTENEGRO, F. M. T. *Aplicações de Técnicas de Pesquisa Operacional em Problemas de Agrupamento do IBGE*. População, espaço e sustentabilidade: contribuições para o desenvolvimento do Brasil, 01 2015, pp. 15–33.

[8] CALIŃSKI, T., E HARABASZ, J. A dendrite method for cluster analysis. *Communications in Statistics-theory and Methods 3*, 1 (1974), 1–27.

[9] CHARU, C. A., E CHANDAN, K. R. *Data clustering: algorithms and applications*. Chapman and Hall/CRC Boca Raton, 2013.

[10] CHOU, C.-H., SU, M.-C., E LAI, E. A new cluster validity measure and its application to image compression. *Pattern Analysis and Applications 7*, 2 (2004), 205–220.

[11] CRUZ, M. D. *O problema de clusterização automática*. PhD thesis, COPPE/UFRJ, Rio de Janeiro, 2010.

[12] DAS, S., ABRAHAM, A., E KONAR, A. Automatic clustering using an improved differential evolution algorithm. *IEEE Transactions on systems, man, and cybernetics-Part A: Systems and Humans 38*, 1 (2008), 218–237.

[13] DAS, S., CHOWDHURY, A., E ABRAHAM, A. A bacterial evolutionary algorithm for automatic data clustering. In *2009 IEEE Congress on Evolutionary Computation* (2009), IEEE, pp. 2403–2410.

[14] DAVIES, D. L., E BOULDIN, D. W. A cluster separation measure. *IEEE transactions on pattern analysis and machine intelligence*, 2 (1979), 224–227.

[15] DOVAL, D., MANCORIDIS, S., E MITCHELL, B. S. Automatic clustering of software systems using a genetic algorithm. In *STEP'99. Proceedings Ninth International Workshop Software Technology and Engineering Practice* (1999), IEEE, pp. 73–81.

[16] DUNN, J. C. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. *Journal of Cybernetics* (1973).

[17] FERRARI, D. G., E SILVA, L. N. D. C. *Introdução a mineração de dados*. Editora Saraiva, 2017.

[18] FISHER, R. A. The use of multiple measurements in taxonomic problems. *Annals of eugenics 7*, 2 (1936), 179–188.

[19] GAN, G., MA, C., E WU, J. *Data clustering: theory, algorithms, and applications*, vol. 20. Siam, 2007.

[20] GOLDSCHMIDT, R. R. *Uma Introdução à Inteligência Computacional: fundamentos, ferramentas e aplicações*. IST-Rio, 2010.

[21] GOLDSCHMIDT, R. R. *Tópicos Especiais em Inteligência Computacional*. IST-Rio, 2011.

[22] HANSEN, P., E JAUMARD, B. Cluster analysis and mathematical programming. *Mathematical programming 79*, 1-3 (1997), 191–215.

[23] HARSH, A., E BALL, J. E. Automatic k-expectation maximization (a k-em) algorithm for data mining applications. *Journal of Computations & Modelling 6*, 3 (2016), 43–85.

[24] HE, H., E TAN, Y. A two-stage genetic algorithm for automatic clustering. *Neurocomputing 81* (2012), 49–59.

[25] HILLIER, F. S., E LIEBERMAN, G. J. *Introdução à pesquisa operacional*. McGraw Hill Brasil, 2013.

[26] JAIN, A. K., DUBES, R. C., E OTHERS. *Algorithms for clustering data*, vol. 6. Prentice hall Englewood Cliffs, 1988.

[27] JAIN, A. K., MURTY, M. N., E FLYNN, P. J. Data clustering: a review. *ACM computing surveys (CSUR) 31*, 3 (1999), 264–323.

[28] JOSÉ-GARCÍA, A., E GÓMEZ-FLORES, W. Automatic clustering using nature-inspired metaheuristics: A survey. *Applied Soft Computing 41* (2016), 192–213.

[29] KETTANI, O., RAMDANI, F., E TADILI, B. Ak-means: an automatic clustering algorithm based on k-means. *Journal of Advanced Computer Science & Technology 4*, 2 (2015), 231.

[30] KUDOVA, P. Clustering genetic algorithm. In *18th International Workshop on Database and Expert Systems Applications (DEXA 2007)* (2007), IEEE, pp. 138–142.

[31] KUNDU, D., SURESH, K., GHOSH, S., DAS, S., ABRAHAM, A., E BADR, Y. Automatic clustering using a synergy of genetic algorithm and multi-objective differential evolution. In *International Conference on Hybrid Artificial Intelligence Systems* (2009), Springer, pp. 177–186.

[32] KUO, R., HUANG, Y., LIN, C.-C., WU, Y.-H., E ZULVIA, F. E. Automatic kernel clustering with bee colony optimization algorithm. *Information Sciences 283* (2014), 107–122.

[33] LINDEN, R. *Algoritmos genéticos (2a ediçao)*. Brasport, 2008.

[34] LINDEN, R. Técnicas de agrupamento. *Revista de Sistemas de Informação da FSMA 4* (2009), 18–36.

[35] LIU, Y., WU, X., E SHEN, Y. Automatic clustering using genetic algorithms. *Applied mathematics and computation 218*, 4 (2011), 1267–1279.

[36] LOESCH, C., E HEIN, N. *Pesquisa operacional*. Editora Saraiva, 2009.

[37] LOESCH, C., E HOELTGEBAUM, M. *Métodos estatísticos multivariados*. Editora Saraiva, 2017.

[38] MARONNA, R., E JACOVKIS, P. M. Multivariate clustering procedures with variable metrics. *Biometrics* (1974), 499–505.

[39] MARTINEZ, J. M., E SANTOS, S. A. Métodos computacionais de otimização. In *XX Colóquio Brasileiro de Matemática, IMPA* (1995).

[40] MAULIK, U., E BANDYOPADHYAY, S. Genetic algorithm-based clustering technique. *Pattern recognition 33*, 9 (2000), 1455–1465.

[41] MAULIK, U., E BANDYOPADHYAY, S. Performance evaluation of some clustering algorithms and validity indices. *IEEE Transactions on pattern analysis and machine intelligence 24*, 12 (2002), 1650–1654.

[42] MISHRA, S., SAHA, S., E MONDAL, S. A multiobjective optimization based entity matching technique for bibliographic databases. *Expert Systems with Applications 65* (2016), 100–115.

[43] MITCHELL, M. *An introduction to genetic algorithms*. MIT press, 1998.

[44] NANDA, S. J., E PANDA, G. A survey on nature inspired metaheuristic algorithms for partitional clustering. *Swarm and Evolutionary computation 16* (2014), 1–18.

[45] OCHI, L. S., DIAS, C. R., E SOARES, S. S. F. Clusterização em mineração de dados. *Instituto de Computação-Universidade Federal Fluminense-Niterói* (2004), 26.

[46] PACHECO, T. M., BRUGIOLO, L., STRÖELE, V., E SÃ, S. Metaheurística inspirada no comportamento das formigas aplicada ao problema de agrupamento. In *XIII Congresso Brasileiro de Inteligência Computacional* (2017).

[47] PAL, N. R., E BEZDEK, J. C. On cluster validity for the fuzzy c-means model. *IEEE Transactions on Fuzzy systems 3*, 3 (1995), 370–379.

[48] QUEIROGA, E. V., SUBRAMANIAN, A., E CABRAL, L. D. A. F. Abordagem baseada em continuous grasp para clusterização de dados. In *XLVIII SBPO* (2016).

[49] RAPOSO, C., ANTUNES, C. H., E BARRETO, J. P. Automatic clustering using a genetic algorithm with new solution encoding and operators. In *International Conference on Computational Science and Its Applications* (2014), Springer, pp. 92–103.

[50] REZENDE, S. O., MARCACINI, R. M., E MOURA, M. F. O uso da mineração de textos para extração e organização não supervisionada de conhecimento. *Embrapa Informática Agropecuária-Artigo em periódico indexado (ALICE)* (2011).

[51] RODRIGUES, W. C., FADEL, A. C., SEMAAN, G. S., E BRITO, J. A. D. M. Um novo método baseado em grade e densidade com tratamento de ruídos para a identificação do número ideal de grupos. In *XLVIII SBPO* (2016).

[52] ROUSSEEUW, P. J. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics 20* (1987), 53–65.

[53] RUSPINI, E. H. Numerical methods for fuzzy clustering. *Information Sciences 2*, 3 (1970), 319–350.

[54] SCHWAAB, M. *Análise de Dados Experimentais: I. Fundamentos de Estatística e Estimação de Parâmetros*. Editora E-papers, 2007.

[55] SEMAAN, G. S., CRUZ, M. D., BRITO, G. D. M., E OCHI, L. S. Proposta de um método de classificação baseado em densidade para a determinação do número ideal de grupos em problemas de clusterização. *Journal of the Brazilian Computational Intelligence Society 10*, 4 (2012), 242–262.

[56] SEMAAN, G. S., FADEL, A. C., DE MOURA BRITO, J. A., E OCHI, L. S. Uma heurística baseada em densidade para o problema de agrupamento automático. *Blucher Marine Engineering Proceedings 2*, 1 (2016), 100–112.

[57] THEODORIDIS, S., E KOUTROUMBAS, K. *Pattern recognition. Second Edition*. Elsevier, 2003.

[58] TSENG, L. Y., E YANG, S. B. A genetic approach to the automatic clustering problem. *Pattern recognition 34*, 2 (2001), 415–424.

[59] VEENMAN, C. J., REINDERS, M. J. T., E BACKER, E. A maximum variance cluster algorithm. *IEEE Transactions on pattern analysis and machine intelligence 24*, 9 (2002), 1273–1280.

[60] WANG, X., QIU, W., E ZAMAR, R. H. Clues: A non-parametric clustering method based on local shrinking. *Computational Statistics & Data Analysis 52*, 1 (2007), 286–298.